New Advances in Machine Learning

New Advances in Machine Learning

Edited by

Yagang Zhang

In-Tech
intechweb.org

Published by In-Teh

In-Teh
Olajnica 19/2, 32000 Vukovar, Croatia

www.intechweb.org
Additional copies can be obtained from:
publication@intechweb.org

First published February 2010
Printed in India

Technical Editor: Sonja Mujacic
Cover designed by Dino Smrekar

New Advances in Machine Learning,
Edited by Yagang Zhang

p. cm.
ISBN 978-953-307-034-6

Preface

The purpose of this book is to provide an up-to-data and systematical introduction to the principles and algorithms of machine learning. The definition of learning is broad enough to include most tasks that we commonly call "Learning" tasks, as we use the word in daily life. It is also broad enough to encompass computer that improve from experience in quite straight forward ways.

Machine learning addresses the question of how to build computer programs that improve their performance at some task through experience. It attempts to automate the estimation process by building machine learners based upon empirical data. Machine learning algorithms have been proven to be of great practical value in a variety application domain, such as, data mining problems where large databases may contain valuable implicit regularities that can be discovered automatically; poorly understood domains where humans might not have the knowledge needed to develop effective algorithms; domains where the program must dynamically adapt to changing conditions.

Machine learning is inherently a multidisciplinary field. It draws on results from artificial intelligence, probability and statistics, computational complexity theory, control theory, information theory, philosophy, psychology, neurobiology, and other fields. The goal of this book is to present the important advances in the theory and algorithm that from the foundations of machine learning.

Large amount of knowledge about machine learning has been presented in this book, mainly include: classification, support vector machine, discriminant analysis, multi-agent system, image recognition, ant colony optimization, and so on.

The book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners. The wide scope of the book provides them with a good introduction to many approaches of machine learning, and it is also the source of useful bibliographical information.

Editor:

Yagang Zhang

Contents

1

Introduction to Machine Learning

Taiwo Oladipupo Ayodele
University of Portsmouth
United Kingdom

1. Introduction

In present times, giving a computer to carry out any task requires a set of specific instructions or the implementation of an algorithm that defines the rules that need to be followed. The present day computer system has no ability to learn from past experiences and hence cannot readily improve on the basis of past mistakes. So, giving a computer or instructing a computer controlled programme to perform a task requires one to define a complete and correct algorithm for task and then programme the algorithm into the computer. Such activities involve tedious and time consuming effort by specially trained teacher or person. Jaime et al (Jaime G. Carbonell, 1983) also explained that the present day computer systems cannot truly learn to perform a task through examples or through previous solved task and they cannot improve on the basis of past mistakes or acquire new abilities by observing and imitating experts. Machine Learning research endeavours to open the possibility of instruction the computer in such a new way and thereby promise to ease the burden of hand writing programmes and growing problems of complex information that get complicated in the computer.

When approaching a task-oriented acquisition task, one must be aware that the resultant computer system must interact with human and therefore should closely match human abilities. So, learning machine or programme on the other hand will have to interact with computer users who make use of them and consequently the concept and skills they acquire- if not necessarily their internal mechanism must be understandable to humans. Also Alpaydin (Alpaydin, 2004) stated that with advances in computer technology, we currently have the ability to store and process large amount of data, as well as access it from physically distant locations over computer network. Most data acquisition devices are digital now and record reliable data. For example, a supermarket chain that has hundreds of stores all over the country selling thousands of goods to millions of customers. The point of sale terminals record the details of each transaction: date, customer identification code, goods bought and their amount, total money spent and so forth, This typically amounts to gigabytes of data every day. This store data becomes useful only when it is analysed and tuned into information that can be used or be predicted.

We do not know exactly which people are likely to buy a particular product or which author to suggest to people who enjoy reading Hemingway. If we knew, we would not need any analysis of the data; we would just go ahead and write down code. But because we do not, we can only collect data and hope to extract the answers to these and similar question from data. We can construct a good and useful approximation. That approximation may not explain everything, but may still be able to account for some part of data. We believe that identifying the complete process may not be possible, we can still detect certain patterns or regularities. This is the niche of machine learning. Such patterns may help us understand the process, or we can use those patterns to make predictions: Assuming that the future, at least the near future, will not be much different from the past when the sample data was collected, the future predictions can be expected to be right.

Machine learning is not just a database problem, it is a part of artificial intelligence. To be intelligent, a system that is in a changing environment should have the ability to learn. If the system can learn and adapt to such changes, the system designer need not foresee and provide solutions for all possible situations. Machine learning also help us find solutions to may problems in vision, speech recognition and robotics. Lets take the example of recognising of faces: This is a task we do effortlessly; we recognise family members and friends by looking their faces or from their photographs, despite differences in pose, lighting, hair, style and so forth. But we do consciously and are able to explain how we do it. Because we are not able to explain our expertise, we cannot write the computer program. At the same time, we know that a face image is not just a random collection of pixel: a face has structure, it is symmetric. There are the eyes, the nose, the mouth, located in certain places on the face. Each person's face is a pattern that composed of a particular combination of these. By analysing sample face images of person, a learning program captures the pattern specific to that person and then recognises by checking for the pattern in a given image. This is one example of pattern recognition.

Machine learning is programming computers to optimise a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimise the parameter of the model using the training data or past experience. The model may be *predictive* to make predictions in the future, or *descriptive* to gain knowledge from data, or both. Machine learning uses the theory of statistics in building mathematical models, because the core task is making inference from sample. The role of learning is twofold: First, in training, we need efficient algorithms to solve the optimised problem, as well as to store and process the massive amount of data we generally have. Second, once a model is learned, its representation and algorithmic solution for inference needs to be efficient as well. In certain applications, the efficiency of the learning or inference algorithm, namely, its space and time complexity may be as important as its predictive accuracy.

1.1 History of Machine Learning

Over the years, Jaime et al (Jaime G. Carbonell, 1983) elaborated that research in machine learning has been pursued with varying degrees of intensity, using different approaches and placing emphasis on different, aspects and goals. Within the relatively short history of this discipline, one may distinguish three major periods, each centred on a different concept:

- neural modelling and decision-theoretic techniques
- symbolic concept-oriented learning
- knowledge-intensive approaches combining various learning strategies

1.1.1 The Neural Modelling (Self Organised System)

The distinguishing feature of the first concept was the interest in building general purpose learning systems that start with little or no initial structure or task-oriented knowledge. The major thrust of research based on this approach involved constructing a variety of neural model-based machines, with random or partially random initial structure. These systems were generally referred to as neural networks or self-organizing systems. Learning in such systems consisted of incremental changes in the probabilities that neuron-like elements (typically threshold logic units) would transmit a signal. Due to the early computer technology, most of the research under this neural network model was either theoretical or involved the construction of special purpose experimental hardware systems, such as perceptrons (Forsyth, 1990), (Ryszard S. Michalski, 1955), (Rosenblatt, 1958) pandemonium (Selfridge, 1959), and (Widrow, 2007). The groundwork for this paradigm was laid in the forties by Rashevsky in the area of mathematical biophysics (Rashevsky, 1948), and by McCulloch (McCulloch, 1943), who discovered the applicability of symbolic logic to modelling nervous system activities. Among the large number of research efforts in this area, one may mention many works such as (Rosenblatt, 1958), (Block H, 1961), (Ashby, 1960), (Widrow, 2007). Related research involved the simulation of evolutionary processes, that through random mutation and “natural” selection might create a system capable of some intelligent, behaviour (for example, (Friedberg, 1958), (Holland, 1980).

Experience in the above areas spawned the new discipline of pattern recognition and led to the development of a decision-theoretic approach to machine learning. In this approach, learning is equated with the acquisition of linear, polynomial, or related discriminant functions from a given set of training examples Example include, (Nilsson, 1982). One of the best known successful learning systems utilizing such techniques (as well as some original new ideas involving non-linear transformations) was Samuel’s checkers program, (Ryszard S. Michalski J. G., 1955). Through repeated training, this program acquired master-level performance somewhat, different, but closely related, techniques utilized methods of statistical decision theory for learning pattern recognition rules.

1.1.2 The Symbolic Concept Acquisition Paradigm

A second major paradigm started to emerge in the early sixties stemming from the work of psychologist and early AI researchers on models of human learning by Hunt (Hunt, 1966). The paradigm utilized logic or graph structure representations rather than numerical or statistical methods Systems learned symbolic descriptions representing higher level knowledge and made strong structural assumptions about the concepts to he acquired. Examples of work in this paradigm include research on human concept acquisition (Hunt, 1966) and various applied pattern recognition systems. Some researchers constructed task-oriented specialized systems that, would acquire knowledge in the context of a practical problem. Ryszard (Ryszard S. Michalski J. G., 1955), learning system was an influential development in this paradigm. In parallel with Winston’s work, different approaches to learning structural concepts from examples emerged, including a family of logic-based inductive learning programs.

1.1.3 The Modern Knowledge-Intensive Paradigm

The third paradigm represented the most recent period of research starting in the mid-seventies. Researchers have broadened their interest beyond learning isolated concepts from examples, and have begun investigating a wide spectrum of learning methods, most based upon knowledge-rich systems specifically, this paradigm can be characterizing by several new trends, including:

1. **Knowledge-Intensive Approaches:** Researchers are strongly emphasizing the use of task-oriented knowledge and the constraints it provides in guiding the learning process One lesson from the failures of earlier knowledge and poor learning systems that is acquire and to acquire new knowledge a system must already possess a great deal of initial knowledge

2. **Exploration of alternative methods of learning:** In addition to the earlier research emphasis on learning from examples, researchers are now investigating a wider variety of learning methods such as learning from instruction, (e.g (Mostow, 1983), learning by analogy and discovery of concepts and classifications (R. S. Michalski, 1983).

In contrast to previous efforts, a number of current systems are incorporating abilities to generate and select tasks and also incorporate heuristics to control their focus of attention by generating learning tasks, proposing experiments to gather training data, and choosing concepts to acquire (e g., Mitchell et al (Mitchell, 2006).

1.2. Importance of Machine Learning

These are benefits of machine learning and these are why research in machine learning is now what could not be avoided or neglected. Using machine learning techniques make life easier for computer users. These are the importance of machine learning. They are:

- Some tasks cannot be defined well except by example; that is we might be able to specify input and output pairs but not a concise relationship between inputs and desired outputs. We would like machines to be able to adjust their internal structure to produce correct outputs for a large number of sample inputs and thus suitably constrain their input and output function to approximate the relationship implicit in the examples.

- It is possible that hidden among large piles of data are important relationships and correlations. Machine learning methods can often be used to extract these relationships (data mining).

- Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might not be completely known at design time. Machine learning methods can be used for on the job improvement of existing machine designs.

- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down.

- Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign. New knowledge about tasks is constantly being discovered by humans. Vocabulary changes. There is a constant stream of new events in the world. Continuing redesign of AI systems to conform to new knowledge is impractical. But machine learning methods might be able to track much of it.

1.3 Machine Learning Varieties

Research in machine learning is now converging from several sources and from artificial intelligent field. These different traditions each bring different methods and different vocabulary which are now being assimilated into a more united discipline. Here is a brief listing of some of the separate disciplines that have contributed to machine learning (Nilsson, 1982).

- **Statistics**: A long-standing problem in statistics is how best to use samples drawn from unknown probability distributions to help decide from which distribution some new sample is drawn. A related problem is how to estimate the value of an unknown function at a new point given the values of this function at a set of sample points. Statistical methods for dealing with these problems can be considered instances of machine learning because the decision and estimation rules depend on a corpus of samples drawn from the problem environment. We will explore some of the statistical methods later in the book. Details about the statistical theory underlying these methods can be found in Orlitsky (Orlitsky, Santhanam, Viswanathan, & Zhang, 2005).

- **Brian Models:** Non linear elements with weighted inputs have been suggested as simple models of biological neurons. Networks of these elements have been studied by several researchers including (Rajesh P. N. Rao, 2002). Brain modelers are interested in how closely these networks approximate the learning phenomena of living brain. We shall see that several important machine learning techniques are based on networks of nonlinear elements often called neural networks. Work inspired by this school is some times called connectionism, brain-style computation or sub-symbolic processing.

- **Adaptive Control Theory**: Control theorists study the problem of controlling a process having unknown parameters which must be estimated during operation. Often, the parameters change during operation and the control process must track these changes. Some aspects of controlling a robot based on sensory inputs represent instances of this sort of problem.

- **Psychological Models:** Psychologists have studied the performance of humans in various learning tasks. An early example is the EPAM network for storing and retrieving one member of a pair of words when given another (Friedberg, 1958). Related work led to a number of early decision tree, (Hunt, 1966) and semantic network, (Anderson, 1995) methods. More recent work of this sort has been influenced by activities in artificial intelligence which we will be presenting. Some of the work in reinforcement learning can be traced to efforts to model how reward stimuli influence the learning of goal seeking behaviour in animals, (Richard S. Sutton, 1998). Reinforcement learning is an important theme in machine learning research.

- **Artificial Intelligence** From the beginning, AI research has been concerned with machine learning. Samuel developed a prominent early program that learned parameters of a function for evaluating board positions in the game of checkers. AI researchers have also explored the role of analogies in learning and how future actions and decisions can be based on previous exemplary cases. Recent work has been directed at discovering rules for expert systems using decision tree methods and inductive logic programming Another theme has been saving and generalizing the results of problem solving using explanation based learning, (Mooney, 2000) ,(Y. Chali, 2009).

- **Evolutionary Models**
  In nature, not only do individual animals learn to perform better, but species evolve to be better fit in their individual niches. Since the distinction between evolving and learning can be blurred in computer systems, techniques that model certain aspects of biological evolution have been proposed as learning methods to improve the performance of computer programs. Genetic algorithms and genetic programming (Oltean, 2005) are the most prominent computational techniques for evolution.

2. References

Allix, N. M. (2003, April). Epistemology And Knowledge Management Concepts And Practices. *Journal of Knowledge Management Practice* .

Alpaydin, E. (2004). *Introduction to Machine Learning.* Massachusetts, USA: MIT Press.

Anderson, J. R. (1995). *Learning and Memory.* Wiley, New York, USA.

Anil Mathur, G. P. (1999). Socialization influences on preparation for later life. *Journal of Marketing Practice: Applied Marketing Science , 5* (6,7,8), 163 - 176.

Ashby, W. R. (1960). *Design of a Brain, The Origin of Adaptive Behaviour.* John Wiley and Son.

Batista, G. &. (2003). An Analysis of Four Missing Data Treatment Methods for Suppervised Learning. *Applied Artificial Intelligence , 17*, 519-533.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford, England: Oxford University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics).* New York, New York: Springer Science and Business Media.

Block H, D. (1961). The Perceptron: A Model of Brian Functioning. *34* (1), 123-135.

Carling, A. (1992). *Introducing Neural Networks* . Wilmslow, UK: Sigma Press.

D. Michie, D. J. (1994). *Machine Learning, Neural and Statistical Classification.* Prentice Hall Inc.

Fausett, L. (19994). *Fundamentals of Neural Networks.* New York: Prentice Hall.

Forsyth, R. S. (1990). The strange story of the Perceptron. *Artificial Intelligence Review* , *4* (2), 147-155.

Friedberg, R. M. (1958). A learning machine: Part, 1. *IBM Journal* , 2-13.

Ghahramani, Z. (2008). Unsupervised learning algorithms are designed to extract structure from data. *178*, pp. 1-8. IOS Press.

Gillies, D. (1996). *Artificial Intelligence and Scientific Method.* OUP Oxford.

Haykin, S. (19994). *Neural Networks: A Comprehensive Foundation.* New York: Macmillan Publishing.

Hodge, V. A. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review* , *22* (2), 85-126.

Holland, J. (1980). Adaptive Algorithms for Discovering and Using General Patterns in Growing Knowledge Bases Policy Analysis and Information Systems. *4* (3).

Hunt, E. B. (1966). Experiment in Induction.

Ian H. Witten, E. F. (2005). *Data Mining Practical Machine Learning and Techniques* (Second edition ed.). Morgan Kaufmann.

Jaime G. Carbonell, R. S. (1983). Machine Learning: A Historical and Methodological Analysis. *Association for the Advancement of Artificial Intelligence* , *4* (3), 1-10.

Kohonen, T. (1997). Self-Organizating Maps.

Luis Gonz, l. A. (2005). Unified dual for bi-class SVM approaches. *Pattern Recognition* , *38* (10), 1772-1774.

McCulloch, W. S. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophysics* , 115-133.

Mitchell, T. M. (2006). *The Discipline of Machine Learning.* Machine Learning Department technical report CMU-ML-06-108, Carnegie Mellon University.

Mooney, R. J. (2000). Learning Language in Logic. In L. N. Science, *Learning for Semantic Interpretation: Scaling Up without Dumbing Down* (pp. 219-234). Springer Berlin / Heidelberg.

Mostow, D. (1983). *Transforming declarative advice into effective procedures: a heuristic search cxamplc In I?. S. Michalski,.* Tioga Press.

Nilsson, N. J. (1982). *Principles of Artificial Intelligence (Symbolic Computation / Artificial Intelligence).* Springer.

Oltean, M. (2005). Evolving Evolutionary Algorithms Using Linear Genetic Programming. *13* (3), 387 - 410 .

Orlitsky, A., Santhanam, N., Viswanathan, K., & Zhang, J. (2005). Convergence of profile based estimators. *Proceedings of International Symposium on Information Theory. Proceedings. International Symposium on*, pp. 1843 - 1847. Adelaide, Australia: IEEE.

Patterson, D. (19996). *Artificial Neural Networks.* Singapore: Prentice Hall.

R. S. Michalski, T. J. (1983). *Learning from Observation: Conceptual Clustering.* TIOGA Publishing Co.

Rajesh P. N. Rao, B. A. (2002). *Probabilistic Models of the Brain.* MIT Press.

Rashevsky, N. (1948). *Mathematical Biophysics:Physico-Mathematical Foundations of Biology.* Chicago: Univ. of Chicago Press.

Richard O. Duda, P. E. (2000). *Pattern Classification* (2nd Edition ed.).

Richard S. Sutton, A. G. (1998). *Reinforcement Learning.* MIT Press.

Ripley, B. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain . *Psychological Review , 65* (6), 386-408.

Russell, S. J. (2003). *Artificial Intelligence: A Modern Approach* (2nd Edition ed.). Upper Saddle River, NJ, NJ, USA: Prentice Hall.

Ryszard S. Michalski, J. G. (1955). *Machine Learning: An Artificial Intelligence Approach (Volume I).* Morgan Kaufmann .

Ryszard S. Michalski, J. G. (1955). *Machine Learning: An Artificial Intelligence Approach.*

Selfridge, O. G. (1959). Pandemonium: a paradigm for learning. *In The mechanisation of thought processes. H.M.S.O., London.* London.

Sleeman, D. H. (1983). *Inferring Student Models for Intelligent CAI. Machine Learning.* Tioga Press.

Tapas Kanungo, D. M. (2002). A local search approximation algorithm for k-means clustering. *Proceedings of the eighteenth annual symposium on Computational geometry* (pp. 10-18). Barcelona, Spain : ACM Press.

Timothy Jason Shepard, P. J. (1998). Decision Fusion Using a Multi-Linear Classifier . *In Proceedings of the International Conference on Multisource-Multisensor Information Fusion.*

Tom, M. (1997). *Machibe Learning.* Machine Learning, Tom Mitchell, McGraw Hill, 1997: McGraw Hill.

Trevor Hastie, R. T. (2001). *The Elements of Statistical Learning.* New york, NY, USA: Springer Science and Business Media.

Widrow, B. W. (2007). *Adaptive Inverse Control: A Signal Processing Approach.* Wiley-IEEE Press.

Y. Chali, S. R. (2009). Complex Question Answering: Unsupervised Learning Approaches and Experiments. *Journal of Artificial Intelligent Research* , 1-47.

Yu, L. L. (2004, October). Efficient feature Selection via Analysis of Relevance and Redundacy. *JMLR* , 1205-1224.

Zhang, S. Z. (2002). Data Preparation for Data Mining. Applied Artificial Intelligence. *17*, 375 - 381.

Machine Learning Overview

Taiwo Oladipupo Ayodele
University of Portsmouth
United Kingdom

1. Machine Learning Overview

Machine Learning according to Michie et al (D. Michie, 1994) is generally taken to encompass automatic computing procedures based on logical or binary operations that learn a task from a series of examples. Here we are just concerned with classification, and it is arguable what should come under the Machine Learning umbrella. Attention has focussed on decision-tree approaches, in which classification results from a sequence of logical steps. These are capable of representing the most complex problem given sufficient data (but this may mean an enormous amount!). Other techniques, such as genetic algorithms and inductive logic procedures (ILP), are currently under active development and in principle would allow us to deal with more general types of data, including cases where the number and type of attributes may vary, and where additional layers of learning are superimposed, with hierarchical structure of attributes and classes and so on. Machine Learning aims to generate classifying expressions simple enough to be understood easily by the human. They must mimic human reasoning sufficiently to provide insight into the decision process. Like statistical approaches, background knowledge may be exploited in development, but operation is assumed without human intervention.
To learn is:

- to gain knowledge, comprehension, or mastery of through experience or study or to gain knowledge (of something) or acquire skill in (some art or practice)
- to acquire experience of or an ability or a skill in
- to memorize (something), to gain by experience, example, or practice.

Machine Learning can be defines as a process of building computer systems that automatically improve with experience, and implement a learning process. Machine Learning can still be defined as learning the theory automatically from the data, through a process of inference, model fitting, or learning from examples:

- Automated extraction of useful information from a body of data by building good probabilistic models.
- Ideally suited for areas with lots of data in the absence of a general theory.

A major focus of machine learning research is to automatically produce models and a model is a pattern, plan, representation, or description designed to show the main working of a system, or concept, such as rules determinate rule for performing a mathematical operation and obtaining a certain result, a function from sets of formulae to formulae, and patterns (model which can be used to generate things or parts of a thing from data.

Learning is a MANY-FACETED PHENOMENON as described by Jaime et al (Jaime G. Carbonell, 1983) and also stated that Learning processes include the acquisition of new declarative knowledge, the development of motor and cognitive skills through instruction or practice, the organization of new knowledge into general, effective representations, and the discovery of new facts and theories through observation and experimentation. The study and computer modelling of learning processes in their multiple manifestations constitutes the subject matter of machine learning. Although machine learning has been a central concern in artificial intelligence since the early days when the idea of "*self-organizing systems*" was popular, the limitations inherent in the early neural network approaches led to a temporary decline in research volume. More recently, new symbolic methods and knowledge-intensive techniques have yielded promising results and these in turn have led to the current, revival in machine learning research. This book examines some basic methodological issues, existing techniques, proposes a classification of machine learning techniques, and provides a historical review of the major research directions.

Machine Learning according to Michie et al (D. Michie, 1994) is generally taken to encompass automatic computing procedures based on logical or binary operations that learn a task from a series of examples. Here we are just concerned with classification, and it is arguable what should come under the Machine Learning umbrella. Attention has focussed on decision-tree approaches, in which classification results from a sequence of logical steps. These are capable of representing the most complex problem given sufficient data (but this may mean an enormous amount!). Other techniques, such as genetic algorithms and inductive logic procedures (ILP), are currently under active development and in principle would allow us to deal with more general types of data, including cases where the number and type of attributes may vary, and where additional layers of learning are superimposed, with hierarchical structure of attributes and classes and so on. Machine Learning aims to generate classifying expressions simple enough to be understood easily by the human. They must mimic human reasoning sufficiently to provide insight into the decision process. Like statistical approaches, background knowledge may be exploited in development, but operation is assumed without human intervention. Machine learning has always been an integral part of artificial intelligence according to Jaime et al (Jaime G. Carbonell, 1983), and its methodology has evolved in concert, with the major concerns of the field. In response to the difficulties of encoding ever increasing volumes of knowledge in model AI systems, many researchers have recently turned their attention to machine learning as a means to overcome the knowledge acquisition bottleneck. This book presents a taxonomic analysis of machine learning organized primarily by learning strategies and secondarily by knowledge representation and application areas. A historical survey out lining the development of various approaches to machine learning is presented from early neural networks to present knowledge-intensive techniques.

1.1 The Aim of Machine Learning

The field of machine learning can be organized around three primary research Areas:

- **Task-Oriented Studies**: The development and analysis of learning systems oriented toward solving a predetermined set, of tasks (also known as the "engineering approach").
- **Cognitive Simulation:** The investigation and computer simulation of human learning processes (also known as the "cognitive modelling approach")
- **Theoretical Analysis:** the theoretical exploration of the space of possible learning methods and algorithms independent application domain.

Although many research efforts strive primarily towards one of these objectives, progress in on objective often lends to progress in another. For example, in order to investigate the space of possible learning methods, a reasonable starting point may be to consider the only known example of robust learning behaviour, namely humans (and perhaps other biological systems) Similarly, psychological investigations of human learning may held by theoretical analysis that may suggest various possible learning models. The need to acquire a particular form of knowledge in stone task-oriented study may itself spawn new theoretical analysis or pose the question: "how do humans acquire this specific skill (or knowledge)?" The existence of these mutually supportive objectives reflects the entire field of artificial intelligence where expert system research, cognitive simulation, and theoretical studies provide some (cross-fertilization of problems and ideas (Jaime G. Carbonell, 1983).

1.1.1 Applied Learning Systems

At, present, instructing a computer or a computer-controlled robot, to perform a task requires one to define a complete and correct, algorithm for that task, and then laboriously program the algorithm into a computer. These activities typically involve a tedious and time-consuming effort by specially trained personnel. Present-day computer systems cannot truly learn to perform a task through examples or by analogy to a similar, previous-solved task. Nor can they improve significantly on the basis of past, mistakes or acquire new abilities by observing and imitating experts. Machine learning research strives to open the possibility of instructing computers in such new ways, and thereby promises to ease the burden of hand-programming growing volumes of increasingly complex information into the computers of tomorrow. The rapid expansion of application and availability of computers today makes this possibility even more attractive and desirable.

1.1.2 Knowledge acquisition

When approaching a task-oriented knowledge acquisition task, one must be aware that, the resultant computer system must interact with humans, and therefore should closely parallel human abilities. The traditional argument that an engineering approach need not reflect human or biological performance and is not, truly applicable to machine learning. Since airplane, a successful result on an almost pure engineering approach, better little resemblance to their biological counterparts, one may argue that applied knowledge acquisition systems could be equally divorced from any consideration of human capabilities. This argument does not apply

here because airplanes need not interact, with or understand birds Learning machines, on the other hand, will have to interact, with the people who make use of them, and consequently the concept and skills they acquire- if not necessarily their internal mechanism and must be understandable to human.

1.2 Machine Learning as a Science

The clear contender for a cognitive invariant in human is the learning mechanism which is the ability facts, skills and more abstractive concepts. Therefore understanding human learning well enough to reproduce aspect of that learning behaviour in a computer system is, in itself, a worthy scientific goal. Moreover, the computer can render substantial assistance to cognitive psychology, in that it may be used to test the consistency and completeness of learning theories and enforce a commitment to the fine-structure process-level detail that precludes meaningless tautological or untestable theories (Bishop, 2006).

The study of human learning processes is also of considerable practical significance. Gaining insights into the principles underlying human learning abilities is likely to lead to more effective educational techniques. Machine learning research is all about developing intelligent computer assistant or a computer tutoring systems and many of these goals are shared within the machine learning fields. According to Jaime et al (Jaime G. Carbonell, 1983) who stated computer tutoring are starting to incorporate abilities to infer models of student competence from observed performance. Inferring the scope of a student's knowledge and skills in a particular area allows much more effective and individualized tutoring of the student (Sleeman, 1983).

The basic scientific objective of machine learning is the exploration of possible learning mechanisms, including the discovery of different induction algorithms, the scope of theoretical limitations of certain method seems to be the information that must be available to the learner, the issue of coping with imperfect training data and the creation of general techniques applicable in many task domains. There is not reason to believe that human learning methods are the only possible mean of acquiring knowledge and skills. In fact, common sense suggests that human learning represents just one point in an uncharted space of possible learning methods- a point that through the evolutionary process is particularly well suited to cope with the general physical environment in which we exist. Most theoretical work in machine learning are centred on creation, characterization and analysis of general learning methods, with the major emphasis on analyzing generality and performance rather than psychological plausibility.

Whereas theoretical analysis provides a means of exploring the space of possible learning methods, the task-oriented approach provides a vehicle to test and improve the performance of functional learning systems and testing applied learning systems, one can determine the cost-effectiveness trade-offs and limitations of particular approaches to learning. In this way, individual data points in the space possible learning systems are explored and the space itself becomes better understood.

Knowledge Acquisition and Skill Refinement: There are two basic form of learning:

1) Knowledge Acquisition
2) Skill refinement

When it is said that someone learned mathematics, it means that this person acquired concepts of mathematics, understood the meaning and their relationship to each other as well as to the world. The importance of learning in this case is acquisition of *knowledge,* including the description and models of physical systems and their behaviours, incorporating a variety of representations from simple intrusive mental model models, examples and images to completely test mathematical equations and physical laws. A person is said to have learned more if this knowledge explains a broader scope of situations, is more accurate, and is better able to predict the behaviour of the typical world (Allix, 2003). This form of learning is typically to a large variety of situations and is generally learned knowledge acquisition. Therefore, knowledge acquisition is defined as learning a new task coupled with the ability to apply the information in the effective manner.
The second form of learning is the gradual improvement of motor and cognitive skills through practice- *Learning by practice*. Learning such as:

- ✓ Learning to drive a car
- ✓ Learning to play keyboard
- ✓ Learning to ride a bicycle
- ✓ Learning to play piano

If one acquire all textbook knowledge on how to perform these aforementioned activities, this represent the initial phase in developing the required skills. So, the major part of the learning process consists of taming the acquired skill, and improving the mental or motor coordination or learning coordination by repeated practice and correction of deviations from desired behaviour. This form of learning often called skill taming. This differs in many ways from knowledge acquisition. Where knowledge acquisition may be a conscious process whose result is the creation of new representative knowledge structures and mental models, and skill taming is learning from example or learning from repeated practice without concerted conscious effort. Jamie (Jaime G. Carbonell, 1983) explained that most human learning appears to be a mixture of both activities, with intellectual endeavours favouring the former and motor coordination tasks favouring the latter. Present machine learning research focuses on the knowledge acquisition aspect, although some investigations, specifically those concerned with learning in problem-solving and transforming declarative instructions into effective actions, touch on aspects of both types of learning. Whereas knowledge acquisition clearly belongs in the realm of artificial intelligence research, a case could be made that skill refinement comes closer to non-symbolic processes such as those studied in adaptative control system. Hence, perhaps both forms of learning- (knowledge based and refinement learning) can be captured in artificial intelligence models.

1.3 Classification of Machine Learning

There are several areas of machine learning that could be exploited to solve the problems of email management and our approach implemented *unsupervised machine learning method*. **Uunsupervised learning** is a method of machine learning whereby the algorithm is presented with examples from the input space only and a model is fit to these observations. For example, a clustering algorithm would be a form of unsupervised learning. "Unsupervised learning is a method of machine learning where a model is fit to observations. It is distinguished from supervised learning by the fact that there is no *a priori* output. In unsupervised learning, a data set of input objects is gathered. Unsupervised learning then typically treats input objects as a set of random variables. A joint density model is then built for the data set. The problem of **unsupervised learning** involved learning patterns in the input when no specific output values are supplied" according to Russell (Russell, 2003).

In the *unsupervised learning problem*, we observe only the features and have no measurements of the outcome. Our task is rather to describe how the data are organized or clustered". Hastie (Trevor Hastie, 2001) explained that "In *unsupervised learning* or *clustering* there is no explicit teacher, and the system forms clusters or "natural groupings" of the input patterns. "Natural" is always defined explicitly or implicitly in the clustering system itself; and given a particular set of patterns or cost function, different clustering algorithms lead to different clusters. Often the user will set the hypothesized number of different clusters ahead of time, but how should this be done? How do we avoid inappropriate representations?" according to Duda (Richard O. Duda, 2000).

There are various categories in the field of artificial intelligence. The classifications of machine learning systems are:

- **Supervised Machine Learning:** Supervised learning is a machine learning technique for learning a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output). To achieve this, the learner has to generalize from the presented data to unseen situations in a "reasonable" way (see inductive bias). (Compare with unsupervised learning.)

  Supervised learning is a machine learning technique whereby the algorithm is first presented with training data which consists of examples which include both the inputs and the desired outputs; thus enabling it to learn a function. The learner should then be able to generalize from the presented data to unseen examples." by Mitchell (Mitchell, 2006). Supervised learning also implies we are given a training set of (X, Y) pairs by a "teacher". We know (sometimes only approximately) the values of *f* for the m samples in the training set, ≡ we assume that if we can find a hypothesis, *h*, that closely agrees with *f* for the members of ≡ then this hypothesis will be a good guess for *f* especially if ≡ is large. Curvefitting is a simple example of supervised learning of a function. Suppose we are given the values of a two-dimensional function. *f*, at the four sample points shown by the solid circles in Figure 9. We want to fit these four points with a function, *h*, drawn from the set, *H*, of second-degree functions. We show there a two-dimensional parabolic surface above the x_1. x_2, plane that fits the points. This parabolic function, *h*, is our hypothesis about the function *f*, which produced the four samples. In this case, $h\ = $ f at the four samples, but we need not have required exact matches. Read more in section 3.1.

- **Unsupervised Machine Learning:** Unsupervised learning[1] is a type of machine learning where manual labels of inputs are not used. It is distinguished from supervised learning approaches which learn how to perform a task, such as classification or regression, using a set of human prepared examples. **.**Unsupervised learning means we are only given the Xs and some (ultimate) feedback function on our performance. We simply have a training set of vectors without function values of them. The problem in this case, typically, is to partition the training set into subsets, $\equiv_1 \ldots . \equiv_R$, in some appropriate way.

1.3.1 Classification of Machine Learning

Classification of machine learning system could be implemented along many different dimensions and we have chosen these two dimensions:

- **Inference Learning:** This is a form of classification on the basis of underlying strategy that is involved. These strategies will depend on the amount of inference the learning system performs on the information provided to the system.
  Now learning strategies are distinguished by the amount of inference the learner performs on the information provided. So, if a computer system performs email classification for example, it knowledge increases but this may not perform any inference on the new information, this means all cognitive efforts is on the part of the analyst or programmer. But if the machine learning classifier independently discovers new theories or adopt new concepts, this will perform a very substantial inference. This is what is called *deriving knowledge from example or experiments or by observation.* An example is: When a student wants to solve statistical problems in a text book – this is a process that involves inference but the solution is not to discover a brand new formula without guidance from a teacher or text book. So, as the amount of inference that the learner is capable of performing increases, the burdens placed on the teacher or on external environ decreases. According to Jaime (Jaime G. Carbonell, 1983) , (Anil Mathur, 1999) who stated that it is much more difficult to teach a person by explaining each steps in a complex task than by showing that person the way that similar tasks are usually done. It more difficult yet to programme a computer to perform a complex task than to instruct a person to perform the task; as programming requires explicit specification of all prerequisite details, whereas a person receiving instruction can use prior knowledge and common sense to fill in most mundane details.
- **Knowledge Representation:** This is a form of skill acquire by the learner on the basis of the type of representation of the knowledge.

1.3.2 Existing Learning Systems

There are many other existing learning systems that employ multiple strategies and knowledge representations and some have been applied to more than one. In the knowledge based machine learning method, no inference is used but the learner display the transformation of knowledge in varieties of ways:

[1] http://en.wikipedia.org/wiki/Unsupervised_learning

- **Learning by being programmed:** When an algorithm or code is written to perform specific task. E.g. a code is written as a guessing game for the type of animal. Such a programme could be modified by external entity.
- **Learning by memorisation:** This is by memorising given facts or data with no inference drawn from the incoming information or data.
- **Learning from examples:** This is a special case of inductive learning. Given a set of examples and counterexamples of a concept, the learner induces a general concept description that describes all of the positive examples and none of the counterexamples. Learning from examples is a method has been heavily investigated in artificial intelligence field. The amount of inference perform by the learner is much greater than in learning from instructions, (Anil Mathur, 1999), (Jaime G. Carbonell, 1983).
- **Learning from Observation:** This is an unsupervised learning approach and is a very general form of inductive learning that includes discovery systems, theory formation tasks, the creation of classification criteria to form taxonomic hierarchies and similar task to be performed without benefit of an external teacher. Unsupervised learning requires the learner to perform more inference than any approach as previously explained. The learner is not provided with a set if data or instance of a particular concept. The above classification of learning strategies should help one to compare various learning systems in terms of their underlying mechanisms, in terms of the available external source of information and in terms of the degree to which they reply on pre-organised knowledge. Read more in section 3.2.

1.4 Machine Learning Applications

The other aspect for classifying learning systems is the area of application which gives a new dimension for machine learning. Below are areas to which various existing learning systems have been applied. They are:

1) Computer Programming
2) Game playing (chess, poker, and so on)
3) Image recognition, Speech recognition
4) Medical diagnosis
5) Agriculture, Physics
6) Email management, Robotics
7) Music
8) Mathematics
9) Natural Language Processing and many more.

2. References

Allix, N. M. (2003, April). Epistemology And Knowledge Management Concepts And Practices. *Journal of Knowledge Management Practice* .

Alpaydin, E. (2004). *Introduction to Machine Learning*. Massachusetts, USA: MIT Press.

Anderson, J. R. (1995). *Learning and Memory.* Wiley, New York, USA.

Anil Mathur, G. P. (1999). Socialization influences on preparation for later life. *Journal of Marketing Practice: Applied Marketing Science , 5* (6,7,8), 163 - 176.

Ashby, W. R. (1960). *Design of a Brain, The Origin of Adaptive Behaviour.* John Wiley and Son.

Batista, G. &. (2003). An Analysis of Four Missing Data Treatment Methods for Suppervised Learning. *Applied Artificial Intelligence , 17*, 519-533.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford, England: Oxford University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics).* New York, New York: Springer Science and Business Media.

Block H, D. (1961). The Perceptron: A Model of Brian Functioning. *34* (1), 123-135.

Carling, A. (1992). *Introducing Neural Networks* . Wilmslow, UK: Sigma Press.

D. Michie, D. J. (1994). *Machine Learning, Neural and Statistical Classification.* Prentice Hall Inc.

Fausett, L. (19994). *Fundamentals of Neural Networks.* New York: Prentice Hall.

Forsyth, R. S. (1990). The strange story of the Perceptron. *Artificial Intelligence Review , 4* (2), 147-155.

Friedberg, R. M. (1958). A learning machine: Part, 1. *IBM Journal* , 2-13.

Ghahramani, Z. (2008). Unsupervised learning algorithms are designed to extract structure from data. *178*, pp. 1-8. IOS Press.

Gillies, D. (1996). *Artificial Intelligence and Scientific Method.* OUP Oxford.

Haykin, S. (19994). *Neural Networks: A Comprehensive Foundation.* New York: Macmillan Publishing.

Hodge, V. A. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review, 22* (2), 85-126.

Holland, J. (1980). Adaptive Algorithms for Discovering and Using General Patterns in Growing Knowledge Bases Policy Analysis and Information Systems. *4* (3).

Hunt, E. B. (1966). Experiment in Induction.

Ian H. Witten, E. F. (2005). *Data Mining Practical Machine Learning and Techniques* (Second edition ed.). Morgan Kaufmann.

Jaime G. Carbonell, R. S. (1983). Machine Learning: A Historical and Methodological Analysis. *Association for the Advancement of Artificial Intelligence , 4* (3), 1-10.

Kohonen, T. (1997). Self-Organizating Maps.

Luis Gonz, l. A. (2005). Unified dual for bi-class SVM approaches. *Pattern Recognition , 38* (10), 1772-1774.

McCulloch, W. S. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophysics* , 115-133.

Mitchell, T. M. (2006). *The Discipline of Machine Learning.* Machine Learning Department technical report CMU-ML-06-108, Carnegie Mellon University.

Mooney, R. J. (2000). Learning Language in Logic. In L. N. Science, *Learning for Semantic Interpretation: Scaling Up without Dumbing Down* (pp. 219-234). Springer Berlin / Heidelberg.

Mostow, D. (1983). *Transforming declarative advice into effective procedures: a heuristic search cxamplc In I?. S. Michalski,.* Tioga Press.

Nilsson, N. J. (1982). *Principles of Artificial Intelligence (Symbolic Computation / Artificial Intelligence).* Springer.

Oltean, M. (2005). Evolving Evolutionary Algorithms Using Linear Genetic Programming. *13* (3), 387 - 410 .

Orlitsky, A., Santhanam, N., Viswanathan, K., & Zhang, J. (2005). Convergence of profile based estimators. *Proceedings of International Symposium on Information Theory. Proceedings. International Symposium on*, pp. 1843 - 1847. Adelaide, Australia: IEEE.

Patterson, D. (19996). *Artificial Neural Networks.* Singapore: Prentice Hall.

R. S. Michalski, T. J. (1983). *Learning from Observation: Conceptual Clustering.* TIOGA Publishing Co.

Rajesh P. N. Rao, B. A. (2002). *Probabilistic Models of the Brain.* MIT Press.

Rashevsky, N. (1948). *Mathematical Biophysics:Physico-Mathematical Foundations of Biology.* Chicago: Univ. of Chicago Press.

Richard O. Duda, P. E. (2000). *Pattern Classification* (2nd Edition ed.).

Richard S. Sutton, A. G. (1998). *Reinforcement Learning.* MIT Press.

Ripley, B. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain . *Psychological Review , 65* (6), 386-408.

Russell, S. J. (2003). *Artificial Intelligence: A Modern Approach* (2nd Edition ed.). Upper Saddle River, NJ, NJ, USA: Prentice Hall.

Ryszard S. Michalski, J. G. (1955). *Machine Learning: An Artificial Intelligence Approach (Volume I).* Morgan Kaufmann .

Ryszard S. Michalski, J. G. (1955). *Machine Learning: An Artificial Intelligence Approach.*

Selfridge, O. G. (1959). Pandemonium: a paradigm for learning. *In The mechanisation of thought processes. H.M.S.O., London.* London.

Sleeman, D. H. (1983). *Inferring Student Models for Intelligent CAI. Machine Learning.* Tioga Press.

Tapas Kanungo, D. M. (2002). A local search approximation algorithm for k-means clustering. *Proceedings of the eighteenth annual symposium on Computational geometry* (pp. 10-18). Barcelona, Spain : ACM Press.

Timothy Jason Shepard, P. J. (1998). Decision Fusion Using a Multi-Linear Classifier . *In Proceedings of the International Conference on Multisource-Multisensor Information Fusion.*

Tom, M. (1997). *Machibe Learning.* Machine Learning, Tom Mitchell, McGraw Hill, 1997: McGraw Hill.

Trevor Hastie, R. T. (2001). *The Elements of Statistical Learning.* New york, NY, USA: Springer Science and Business Media.

Widrow, B. W. (2007). *Adaptive Inverse Control: A Signal Processing Approach.* Wiley-IEEE Press.

Y. Chali, S. R. (2009). Complex Question Answering: Unsupervised Learning Approaches and Experiments. *Journal of Artificial Intelligent Research* , 1-47.

Yu, L. L. (2004, October). Efficient feature Selection via Analysis of Relevance and Redundacy. *JMLR* , 1205-1224.

Zhang, S. Z. (2002). Data Preparation for Data Mining. Applied Artificial Intelligence. *17*, 375 - 381.

3

Types of Machine Learning Algorithms

Taiwo Oladipupo Ayodele
University of Portsmouth
United Kingdom

1. Machine Learning: Algorithms Types

Machine learning algorithms are organized into taxonomy, based on the desired outcome of the algorithm. Common algorithm types include:

- Supervised learning --- where the algorithm generates a function that maps inputs to desired outputs. One standard formulation of the supervised learning task is the classification problem: the learner is required to learn (to approximate the behavior of) a function which maps a vector into one of several classes by looking at several input-output examples of the function.
- Unsupervised learning --- which models a set of inputs: labeled examples are not available.
- Semi-supervised learning --- which combines both labeled and unlabeled examples to generate an appropriate function or classifier.
- Reinforcement learning --- where the algorithm learns a policy of how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback that guides the learning algorithm.
- Transduction --- similar to supervised learning, but does not explicitly construct a function: instead, tries to predict new outputs based on training inputs, training outputs, and new inputs.
- Learning to learn --- where the algorithm learns its own inductive bias based on previous experience.

The performance and computational analysis of machine learning algorithms is a branch of statistics known as computational learning theory.
Machine learning is about designing algorithms that allow a computer to learn. Learning is not necessarily involves consciousness but learning is a matter of finding statistical regularities or other patterns in the data. Thus, many machine learning algorithms will barely resemble how human might approach a learning task. However, learning algorithms can give insight into the relative difficulty of learning in different environments.

1.1 Supervised Learning Approach

Supervised learning[1] is fairly common in classification problems because the goal is often to get the computer to learn a classification system that we have created. Digit recognition, once again, is a common example of classification learning. More generally, classification learning is appropriate for any problem where deducing a classification is useful and the classification is easy to determine. In some cases, it might not even be necessary to give pre-determined classifications to every instance of a problem if the agent can work out the classifications for itself. This would be an example of unsupervised learning in a classification context.

Supervised learning[2] often leaves the probability for inputs undefined. This model is not needed as long as the inputs are available, but if some of the input values are missing, it is not possible to infer anything about the outputs. Unsupervised learning, all the observations are assumed to be caused by latent variables, that is, the observations is assumed to be at the end of the causal chain. Examples of supervised learning and unsupervised learning are shown in the figure 1 below:

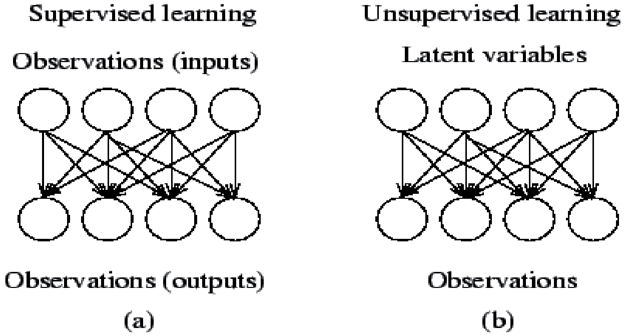


Fig. 1. Examples of Supervised and Unsupervised Learning

Supervised learning[3] is the most common technique for training neural networks and decision trees. Both of these techniques are highly dependent on the information given by the pre-determined classifications. In the case of neural networks, the classification is used to determine the error of the network and then adjust the network to minimize it, and in decision trees, the classifications are used to determine what attributes provide the most information that can be used to solve the classification puzzle. We'll look at both of these in more detail, but for now, it should be sufficient to know that both of these examples thrive on having some "supervision" in the form of pre-determined classifications.

Inductive machine learning is the process of learning a set of rules from instances (examples in a training set), or more generally speaking, creating a classifier that can be used to generalize from new instances. The process of applying supervised ML to a real-world problem is described in Figure F. The first step is collecting the dataset. If a requisite expert is available, then s/he could suggest which fields (attributes, features) are the most

[1] http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.htm

[2] http://www.cis.hut.fi/harri/thesis/valpola_thesis/node34.html

[3] http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.htm

informative. If not, then the simplest method is that of “brute-force,” which means measuring everything available in the hope that the right (informative, relevant) features can be isolated. However, a dataset collected by the “brute-force” method is not directly suitable for induction. It contains in most cases noise and missing feature values, and therefore requires significant pre-processing according to Zhang et al (Zhang, 2002).

The second step is the data preparation and data pre-processing. Depending on the circumstances, researchers have a number of methods to choose from to handle missing data (Batista, 2003). Hodge et al (Hodge, 2004) , have recently introduced a survey of contemporary techniques for outlier (noise) detection. These researchers have identified the techniques’ advantages and disadvantages. Instance selection is not only used to handle noise but to cope with the infeasibility of learning from very large datasets. Instance selection in these datasets is an optimization problem that attempts to maintain the mining quality while minimizing the sample size. It reduces data and enables a data mining algorithm to function and work effectively with very large datasets. There is a variety of procedures for sampling instances from a large dataset. See figure 2 below.

Feature subset selection is the process of identifying and removing as many irrelevant and redundant features as possible (Yu, 2004) . This reduces the dimensionality of the data and enables data mining algorithms to operate faster and more effectively. The fact that many features depend on one another often unduly influences the accuracy of supervised ML classification models. This problem can be addressed by constructing new features from the basic feature set. This technique is called feature construction/transformation. These newly generated features may lead to the creation of more concise and accurate classifiers. In addition, the discovery of meaningful features contributes to better comprehensibility of the produced classifier, and a better understanding of the learned concept.Speech recognition using hidden Markov models and Bayesian networks relies on some elements of supervision as well in order to adjust parameters to, as usual, minimize the error on the given inputs.Notice something important here: in the classification problem, the goal of the learning algorithm is to minimize the error with respect to the given inputs. These inputs, often called the "training set", are the examples from which the agent tries to learn. But learning the training set well is not necessarily the best thing to do. For instance, if I tried to teach you exclusive-or, but only showed you combinations consisting of one true and one false, but never both false or both true, you might learn the rule that the answer is always true. Similarly, with machine learning algorithms, a common problem is over-fitting the data and essentially memorizing the training set rather than learning a more general classification technique. As you might imagine, not all training sets have the inputs classified correctly. This can lead to problems if the algorithm used is powerful enough to memorize even the apparently "special cases" that don't fit the more general principles. This, too, can lead to over fitting, and it is a challenge to find algorithms that are both powerful enough to learn complex functions and robust enough to produce generalisable results.

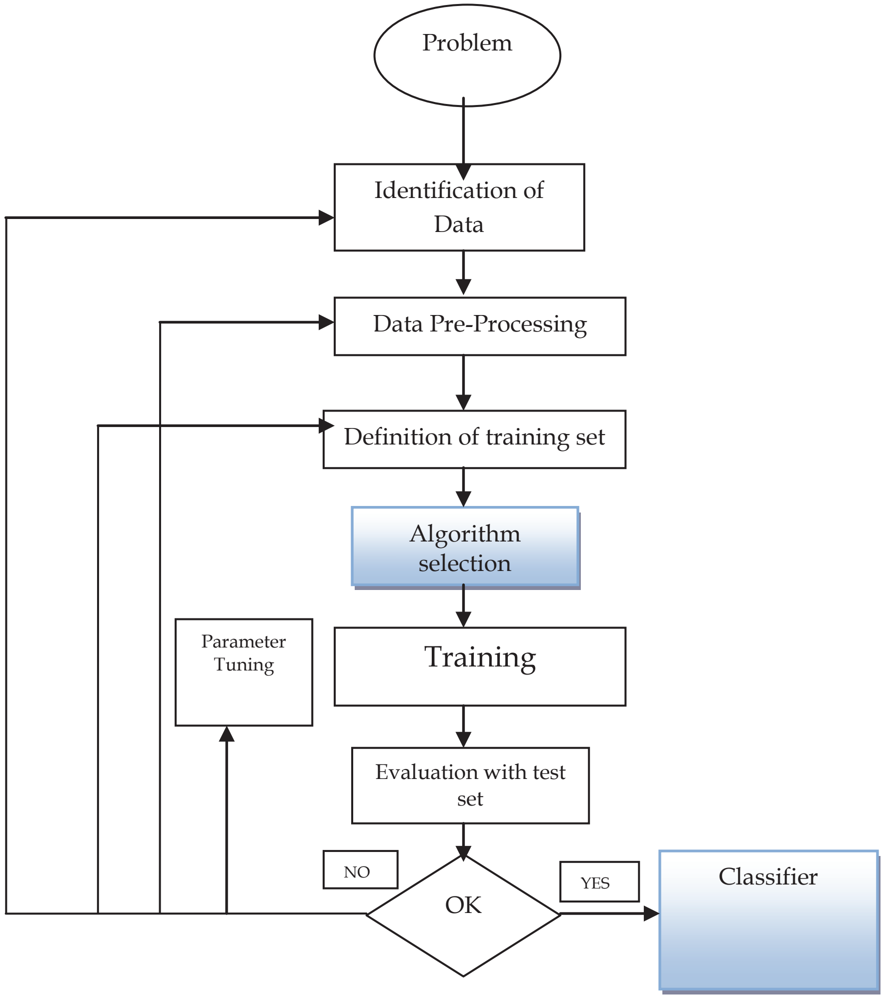


Fig. 2. Machine Learning Supervise Process

1.2 Unsupervised learning

Unsupervised learning[4] seems much harder: the goal is to have the computer learn how to do something that we don't tell it how to do! There are actually two approaches to unsupervised learning. The first approach is to teach the agent not by giving explicit categorizations, but by using some sort of reward system to indicate success. Note that this type of training will generally fit into the decision problem framework because the goal is not to produce a classification but to make decisions that maximize rewards. This approach nicely generalizes to the real world, where agents might be rewarded for doing certain

[4] http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.htm

actions and punished for doing others. Often, a form of reinforcement learning can be used for unsupervised learning, where the agent bases its actions on the previous rewards and punishments without necessarily even learning any information about the exact ways that its actions affect the world. In a way, all of this information is unnecessary because by learning a reward function, the agent simply knows what to do without any processing because it knows the exact reward it expects to achieve for each action it could take. This can be extremely beneficial in cases where calculating every possibility is very time consuming (even if all of the transition probabilities between world states were known). On the other hand, it can be very time consuming to learn by, essentially, trial and error. But this kind of learning can be powerful because it assumes no pre-discovered classification of examples. In some cases, for example, our classifications may not be the best possible. One striking exmaple is that the conventional wisdom about the game of backgammon was turned on its head when a series of computer programs (neuro-gammon and TD-gammon) that learned through unsupervised learning became stronger than the best human chess players merely by playing themselves over and over. These programs discovered some principles that surprised the backgammon experts and performed better than backgammon programs trained on pre-classified examples. A second type of unsupervised learning is called clustering. In this type of learning, the goal is not to maximize a utility function, but simply to find similarities in the training data. The assumption is often that the clusters discovered will match reasonably well with an intuitive classification. For instance, clustering individuals based on demographics might result in a clustering of the wealthy in one group and the poor in another. Although the algorithm won't have names to assign to these clusters, it can produce them and then use those clusters to assign new examples into one or the other of the clusters. This is a data-driven approach that can work well when there is sufficient data; for instance, social information filtering algorithms, such as those that Amazon.com use to recommend books, are based on the principle of finding similar groups of people and then assigning new users to groups. In some cases, such as with social information filtering, the information about other members of a cluster (such as what books they read) can be sufficient for the algorithm to produce meaningful results. In other cases, it may be the case that the clusters are merely a useful tool for a human analyst. Unfortunately, even unsupervised learning suffers from the problem of overfitting the training data. There's no silver bullet to avoiding the problem because any algorithm that can learn from its inputs needs to be quite powerful.

Unsupervised learning algorithms according to Ghahramani (Ghahramani, 2008) are designed to extract structure from data samples. The quality of a structure is measured by a cost function which is usually minimized to infer optimal parameters characterizing the hidden structure in the data. Reliable and robust inference requires a guarantee that extracted structures are typical for the data source, i.e., similar structures have to be extracted from a second sample set of the same data source. Lack of robustness is known as over fitting from the statistics and the machine learning literature. In this talk I characterize the over fitting phenomenon for a class of histogram clustering models which play a prominent role in information retrieval, linguistic and computer vision applications. Learning algorithms with robustness to sample fluctuations are derived from large deviation results and the maximum entropy principle for the learning process.

Unsupervised learning has produced many successes, such as world-champion calibre backgammon programs and even machines capable of driving cars! It can be a powerful technique when there is an easy way to assign values to actions. Clustering can be useful when there is enough data to form clusters (though this turns out to be difficult at times) and especially when additional data about members of a cluster can be used to produce further results due to dependencies in the data. Classification learning is powerful when the classifications are known to be correct (for instance, when dealing with diseases, it's generally straight-forward to determine the design after the fact by an autopsy), or when the classifications are simply arbitrary things that we would like the computer to be able to recognize for us. Classification learning is often necessary when the decisions made by the algorithm will be required as input somewhere else. Otherwise, it wouldn't be easy for whoever requires that input to figure out what it means. Both techniques can be valuable and which one you choose should depend on the circumstances--what kind of problem is being solved, how much time is allotted to solving it (supervised learning or clustering is often faster than reinforcement learning techniques), and whether supervised learning is even possible.

1.3 Algorithm Types

In the area of supervised learning which deals much with classification. These are the algorithms types:

- Linear Classifiers
    - Logical Regression
    - Naïve Bayes Classifier
    - Perceptron
    - Support Vector Machine
- Quadratic Classifiers
- K-Means Clustering
- Boosting
- Decision Tree
    - Random Forest
- Neural networks
- Bayesian Networks

Linear Classifiers: In machine learning, the goal of classification is to group items that have similar feature values, into groups. Timothy et al (Timothy Jason Shepard, 1998) stated that a linear classifier achieves this by making a classification decision based on the value of the linear combination of the features. If the input feature vector to the classifier is a real vector $\vec{x}$, then the output score is

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right),$$

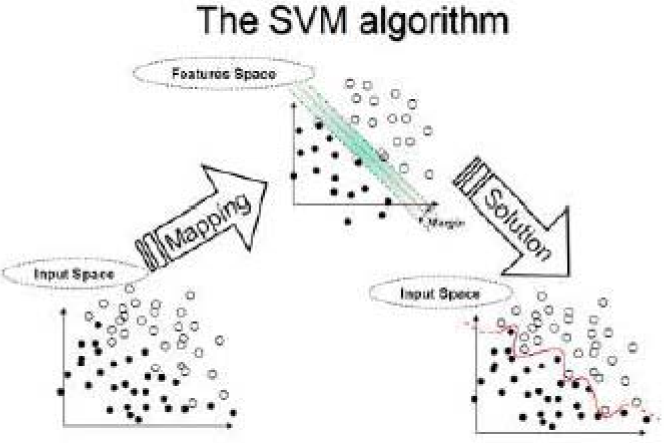
where $\vec{w}$ is a real vector of weights and *f* is a function that converts the dot product of the two vectors into the desired output. The weight vector $\vec{w}$ is learned from a set of labelled training samples. Often *f* is a simple function that maps all values above a certain threshold to the first class and all other values to the second class. A more complex *f* might give the probability that an item belongs to a certain class.

For a two-class classification problem, one can visualize the operation of a linear classifier as splitting a high-dimensional input space with a hyperplane: all points on one side of the hyper plane are classified as "yes", while the others are classified as "no". A linear classifier is often used in situations where the speed of classification is an issue, since it is often the fastest classifier, especially when $\vec{x}$ is sparse. However, decision trees can be faster. Also, linear classifiers often work very well when the number of dimensions in $\vec{x}$ is large, as in document classification, where each element in $\vec{x}$ is typically the number of counts of a word in a document (see document-term matrix). In such cases, the classifier should be well-regularized.

- **Support Vector Machine:** A Support Vector Machine as stated by Luis et al (Luis Gonz, 2005) (SVM) performs classification by constructing an *N*-dimensional hyper plane that optimally separates the data into two categories. SVM models are closely related to neural networks. In fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

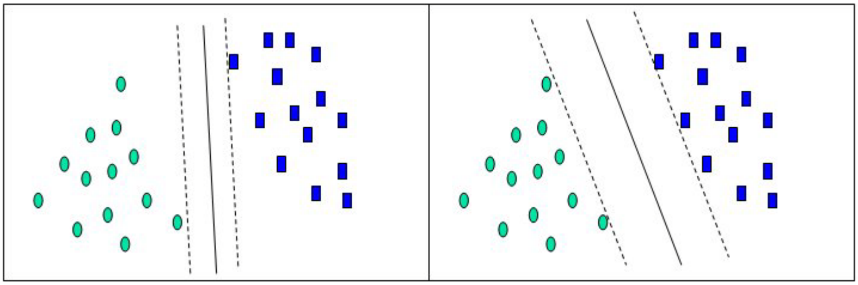
  Support Vector Machine (SVM) models are a close cousin to classical multilayer perceptron neural networks. Using a kernel function, SVM's are an alternative training method for polynomial, radial basis function and multi-layer perceptron classifiers in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.

  In the parlance of SVM literature, a predictor variable is called an *attribute*, and a transformed attribute that is used to define the hyper plane is called a *feature*. The task of choosing the most suitable representation is known as *feature selection*. A set of features that describes one case (i.e., a row of predictor values) is called a *vector*. So the goal of SVM modelling is to find the optimal hyper plane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyper plane are the *support vectors*. The figure below presents an overview of the SVM process.



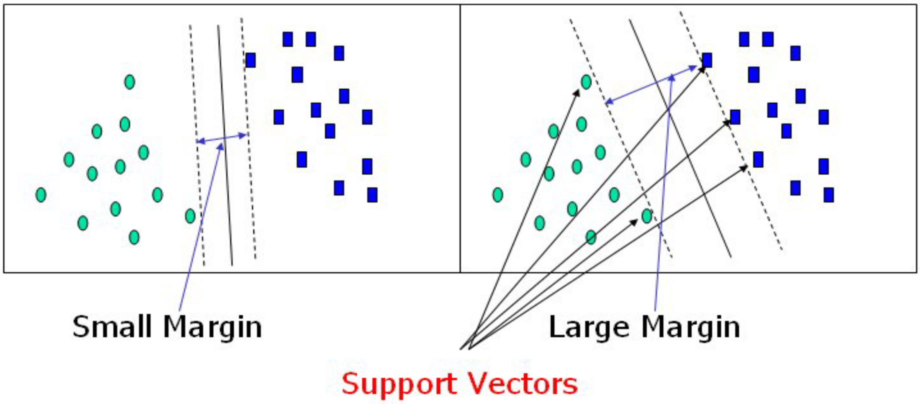
A Two-Dimensional Example

Before considering *N*-dimensional hyper planes, let's look at a simple 2-dimensional example. Assume we wish to perform a classification, and our data has a categorical target variable with two categories. Also assume that there are two predictor variables with continuous values. If we plot the data points using the value of one predictor on the X axis and the other on the Y axis we might end up with an image such as shown below. One category of the target variable is represented by rectangles while the other category is represented by ovals.



In this idealized example, the cases with one category are in the lower left corner and the cases with the other category are in the upper right corner; the cases are completely separated. The SVM analysis attempts to find a 1-dimensional hyper plane (i.e. a line) that separates the cases based on their target categories. There are an infinite number of possible lines; two candidate lines are shown above. The question is which line is better, and how do we define the optimal line.

The dashed lines drawn parallel to the separating line mark the distance between the dividing line and the closest vectors to the line. The distance between the dashed lines is called the *margin*. The vectors (points) that constrain the width of the margin are the *support vectors*. The following figure illustrates this.



An SVM analysis (Luis Gonz, 2005) finds the line (or, in general, hyper plane) that is oriented so that the margin between the support vectors is maximized. In the figure above, the line in the right panel is superior to the line in the left panel.

If all analyses consisted of two-category target variables with two predictor variables, and the cluster of points could be divided by a straight line, life would be easy. Unfortunately, this is not generally the case, so SVM must deal with (a) more than two predictor variables, (b) separating the points with non-linear curves, (c) handling the cases where clusters cannot be completely separated, and (d) handling classifications with more than two categories.

In this chapter, we shall explain three main machine learning techniques with their examples and how they perform in reality. These are:

- K-Means Clustering
- Neural Network
- Self Organised Map

1.3.1 K-Means Clustering

The basic step of k-means clustering is uncomplicated. In the beginning we determine number of cluster K and we assume the centre of these clusters. We can take any random objects as the initial centre or the first K objects in sequence can also serve as the initial centre. Then the K means algorithm will do the three steps below until convergence.

Iterate until *stable* (= no object move group):

1. Determine the centre coordinate
2. Determine the distance of each object to the centre
3. Group the object based on minimum distance

The Figure 3 shows a K- means flow diagram

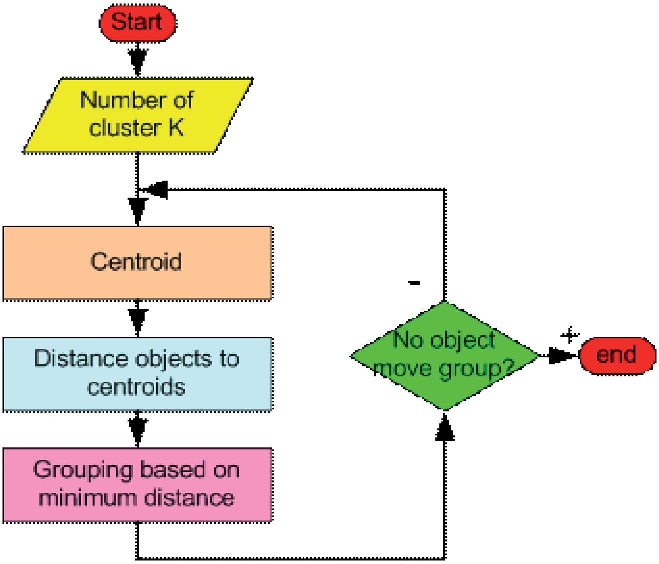

Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Fig. 3. K-means iteration

K-means (Bishop C. M., 1995) and (Tapas Kanungo, 2002) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids shoud be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 ,$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the *n* data points from their respective cluster centres.
The algorithm in figure 4 is composed of the following steps:

1. *Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *Assign each object to the group that has the closest centroid.*
3. *When all objects have been assigned, recalculate the positions of the K centroids.*
4. *Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.*

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centres. The k-means algorithm can be run multiple times to reduce this effect. K-means is a simple algorithm that has been adapted to many problem domains. As we are going to see, it is a good candidate for extension to work with fuzzy feature vectors.

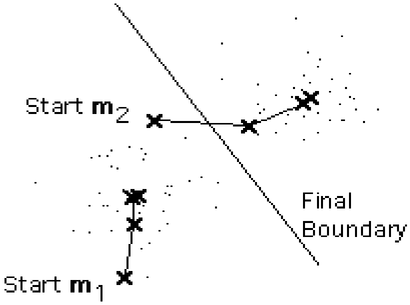
An example

Suppose that we have n sample feature vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ all from the same class, and we know that they fall into k compact clusters, k < n. Let $\mathbf{m}_i$ be the mean of the vectors in cluster i. If the clusters are well separated, we can use a minimum-distance classifier to separate them. That is, we can say that **x** is in cluster i if $|| \mathbf{x} - \mathbf{m}_i ||$ is the minimum of all the k distances. This suggests the following procedure for finding the k means:

- Make initial guesses for the means $\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_k$
- Until there are no changes in any mean
- Use the estimated means to classify the samples into clusters
- For i from 1 to k

- Replace $\mathbf{m}_i$ with the mean of all of the samples for cluster i
- end_for
- end_until

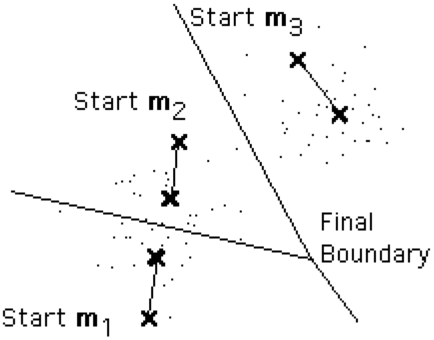
Here is an example showing how the means $\mathbf{m}_1$ and $\mathbf{m}_2$ move into the centers of two clusters.



This is a simple version of the k-means procedure. It can be viewed as a greedy algorithm for partitioning the n samples into k clusters so as to minimize the sum of the squared distances to the cluster centers. It does have some weaknesses:

- The way to initialize the means was not specified. One popular way to start is to randomly choose k of the samples.
- The results produced depend on the initial values for the means, and it frequently happens that suboptimal partitions are found. The standard solution is to try a number of different starting points.
- It can happen that the set of samples closest to $\mathbf{m}_i$ is empty, so that $\mathbf{m}_i$ cannot be updated. This is an annoyance that must be handled in an implementation, but that we shall ignore.
- The results depend on the metric used to measure $|| \mathbf{x} - \mathbf{m}_i ||$. A popular solution is to normalize each variable by its standard deviation, though this is not always desirable.
- The results depend on the value of k.

This last problem is particularly troublesome, since we often have no way of knowing how many clusters exist. In the example shown above, the same algorithm applied to the same data produces the following 3-means clustering. Is it better or worse than the 2-means clustering?



Unfortunately there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k classes and choose the best one according to a given criterion

1.3.2 Neural Network

Neural networks (Bishop C. M., 1995) can actually perform a number of regression and/or classification tasks at once, although commonly each network performs only one. In the vast majority of cases, therefore, the network will have a single output variable, although in the case of many-state classification problems, this may correspond to a number of output units (the post-processing stage takes care of the mapping from output units to output variables). If you do define a single network with multiple output variables, it may suffer from cross-talk (the hidden neurons experience difficulty learning, as they are attempting to model at least two functions at once). The best solution is usually to train separate networks for each output, then to combine them into an ensemble so that they can be run as a unit. Neural methods are:

- **Multilayer Perceptrons:** This is perhaps the most popular network architecture in use today, due originally to Rumelhart and McClelland (1986) and discussed at length in most neural network textbooks (e.g., Bishop, 1995). This is the type of network discussed briefly in previous sections: the units each perform a biased weighted sum of their inputs and pass this activation level through a transfer function to produce their output, and the units are arranged in a layered feed forward topology. The network thus has a simple interpretation as a form of input-output model, with the weights and thresholds (biases) the free parameters of the model. Such networks can model functions of almost arbitrary complexity, with the number of layers, and the number of units in each layer, determining the function complexity. Important issues in Multilayer Perceptrons (MLP) design include specification of the number of hidden layers and the number of units in these layers (Bishop C. M., 1995), (D. Michie, 1994).

  The number of input and output units is defined by the problem (there may be some uncertainty about precisely which inputs to use, a point to which we will return later. However, for the moment we will assume that the input variables are intuitively selected and are all meaningful). The number of hidden units to use is far from clear. As good a starting point as any is to use one hidden layer, with the number of units equal to half the sum of the number of input and output units. Again, we will discuss how to choose a sensible number later.

- **Training Multilayer Perceptrons:** Once the number of layers, and number of units in each layer, has been selected, the network's weights and thresholds must be set so as to minimize the prediction error made by the network. This is the role of the *training algorithms*. The historical cases that you have gathered are used to automatically adjust the weights and thresholds in order to minimize this error. This process is equivalent to fitting the model represented by the network to the training data available. The error of a particular configuration of the network can be determined by running all the training cases through the network, comparing the actual output generated with the desired or target outputs. The differences are combined together by an *error function* to give the network error. The most common error functions are the sum squared error (used for regression problems), where the individual errors of output units on each case are squared and summed together, and the cross entropy functions (used for maximum likelihood classification).

In traditional modeling approaches (e.g., linear modeling) it is possible to algorithmically determine the model configuration that absolutely minimizes this error. The price paid for the greater (non-linear) modeling power of neural networks is that although we can adjust a network to lower its error, we can never be sure that the error could not be lower still.

A helpful concept here is the error surface. Each of the *N* weights and thresholds of the network (i.e., the free parameters of the model) is taken to be a dimension in space. The *N+1*th dimension is the network error. For any possible configuration of weights the error can be plotted in the *N+1*th dimension, forming an *error surface*. The objective of network training is to find the lowest point in this many-dimensional surface.

In a linear model with sum squared error function, this error surface is a parabola (a quadratic), which means that it is a smooth bowl-shape with a single minimum. It is therefore "easy" to locate the minimum.

Neural network error surfaces are much more complex, and are characterized by a number of unhelpful features, such as local minima (which are lower than the surrounding terrain, but above the global minimum), flat-spots and plateaus, saddle-points, and long narrow ravines.

It is not possible to analytically determine where the global minimum of the error surface is, and so neural network training is essentially an exploration of the error surface. From an initially random configuration of weights and thresholds (i.e., a random point on the error surface), the training algorithms incrementally seek for the global minimum. Typically, the gradient (slope) of the error surface is calculated at the current point, and used to make a downhill move. Eventually, the algorithm stops in a low point, which may be a local minimum (but hopefully is the global minimum).

- **The Back Propagation Algorithm:** The best-known example of a neural network training algorithm is back propagation (Haykin, 19994), (Patterson, 19996), (Fausett, 19994). Modern second-order algorithms such as conjugate gradient descent and *Levenberg-Marquardt* (see Bishop, 1995; Shepherd, 1997) (both included in *ST Neural Networks*) are substantially faster (e.g., an order of magnitude faster) for many problems, but *back propagation* still has advantages in some circumstances, and is the easiest algorithm to understand. We will introduce this now, and discuss the more advanced algorithms later. In *back propagation*, the gradient vector of the error surface is calculated. This vector points along the line of steepest descent from the current point, so we know that if we move along it a "short" distance, we will decrease the error. A sequence of such moves (slowing as we near the bottom) will eventually find a minimum of some sort. The difficult part is to decide how large the steps should be.

  Large steps may converge more quickly, but may also overstep the solution or (if the error surface is very eccentric) go off in the wrong direction. A classic example of this in neural network training is where the algorithm progresses very slowly along a steep, narrow, valley, bouncing from one side across to the other. In contrast, very small steps may go in the correct direction, but they also require a large number of iterations. In practice, the step size is proportional to the slope (so that the algorithm settles down in a minimum) and to a special constant: the learning rate. The correct setting for the learning rate is application-dependent, and is typically chosen by experiment; it may also be time-varying, getting smaller as the algorithm progresses.

The algorithm is also usually modified by inclusion of a momentum term: this encourages movement in a fixed direction, so that if several steps are taken in the same direction, the algorithm "picks up speed", which gives it the ability to (sometimes) escape local minimum, and also to move rapidly over flat spots and plateaus.

The algorithm therefore progresses iteratively, through a number of epochs. On each epoch, the training cases are each submitted in turn to the network, and target and actual outputs compared and the error calculated. This error, together with the error surface gradient, is used to adjust the weights, and then the process repeats. The initial network configuration is random, and training stops when a given number of epochs elapses, or when the error reaches an acceptable level, or when the error stops improving (you can select which of these stopping conditions to use).

- **Over-learning and Generalization:** One major problem with the approach outlined above is that it doesn't actually minimize the error that we are really interested in - which is the expected error the network will make when *new* cases are submitted to it. In other words, the most desirable property of a network is its ability to *generalize* to new cases. In reality, the network is trained to minimize the error on the training set, and short of having a perfect and infinitely large training set, this is not the same thing as minimizing the error on the real error surface - the error surface of the underlying and unknown model (Bishop C. M., 1995).

The most important manifestation of this distinction is the problem of over-learning, or over-fitting. It is easiest to demonstrate this concept using polynomial curve fitting rather than neural networks, but the concept is precisely the same.

A polynomial is an equation with terms containing only constants and powers of the variables. For example:

$$y=2x+3$$
$$y=3x^2+4x+1$$

Different polynomials have different shapes, with larger powers (and therefore larger numbers of terms) having steadily more eccentric shapes. Given a set of data, we may want to fit a polynomial curve (i.e., a model) to explain the data. The data is probably noisy, so we don't necessarily expect the best model to pass exactly through all the points. A low-order polynomial may not be sufficiently flexible to fit close to the points, whereas a high-order polynomial is actually too flexible, fitting the data exactly by adopting a highly eccentric shape that is actually unrelated to the underlying function. See figure 4 below.

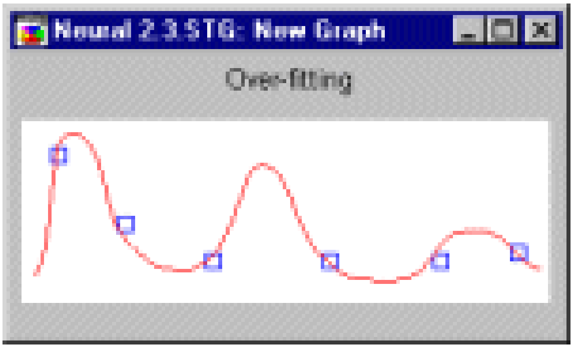


Fig. 4. High-order polynomial sample

Neural networks have precisely the same problem. A network with more weights models a more complex function, and is therefore prone to over-fitting. A network with less weight may not be sufficiently powerful to model the underlying function. For example, a network with no hidden layers actually models a simple linear function. How then can we select the right complexity of network? A larger network will almost invariably achieve a lower error eventually, but this may indicate over-fitting rather than good modeling.

The answer is to check progress against an independent data set, the selection set. Some of the cases are reserved, and not actually used for training in the back propagation algorithm. Instead, they are used to keep an independent check on the progress of the algorithm. It is invariably the case that the initial performance of the network on training and selection sets is the same (if it is not at least approximately the same, the division of cases between the two sets is probably biased). As training progresses, the training error naturally drops, and providing training is minimizing the true error function, the selection error drops too. However, if the selection error stops dropping, or indeed starts to rise, this indicates that the network is starting to overfit the data, and training should cease. When over-fitting occurs during the training process like this, it is called over-learning. In this case, it is usually advisable to decrease the number of hidden units and/or hidden layers, as the network is over-powerful for the problem at hand. In contrast, if the network is not sufficiently powerful to model the underlying function, over-learning is not likely to occur, and neither training nor selection errors will drop to a satisfactory level.

The problems associated with local minima, and decisions over the size of network to use, imply that using a neural network typically involves experimenting with a large number of different networks, probably training each one a number of times (to avoid being fooled by local minima), and observing individual performances. The key guide to performance here is the selection error. However, following the standard scientific precept that, all else being equal, a simple model is always preferable to a complex model, you can also select a smaller network in preference to a larger one with a negligible improvement in selection error.

A problem with this approach of repeated experimentation is that the selection set plays a key role in selecting the model, which means that it is actually part of the training process. Its reliability as an independent guide to performance of the model is therefore compromised - with sufficient experiments, you may just hit upon a lucky network that happens to perform well on the selection set. To add confidence in the performance of the final model, it is therefore normal practice (at least where the volume of training data allows it) to reserve a third set of cases - the test set. The final model is tested with the test set data, to ensure that the results on the selection and training set are real, and not artifacts of the training process. Of course, to fulfill this role properly the test set should be used only once - if it is in turn used to adjust and reiterate the training process, it effectively becomes selection data!

This division into multiple subsets is very unfortunate, given that we usually have less data than we would ideally desire even for a single subset. We can get around this problem by resampling. Experiments can be conducted using different divisions of the available data into training, selection, and test sets. There are a number of approaches to this subset, including random (monte-carlo) resampling, cross-validation, and bootstrap. If we make design decisions, such as the best configuration of neural network to use, based upon a number of experiments with different subset examples, the results will be much more reliable. We can then either use those experiments solely to guide the decision as to which network types to use, and train such networks from scratch with new samples (this removes any sampling bias); or, we can retain the best networks found during the sampling process, but average their results in an ensemble, which at least mitigates the sampling bias.

To summarize, network design (once the input variables have been selected) follows a number of stages:

- Select an initial configuration (typically, one hidden layer with the number of hidden units set to half the sum of the number of input and output units).
- Iteratively conduct a number of experiments with each configuration, retaining the best network (in terms of selection error) found. A number of experiments are required with each configuration to avoid being fooled if training locates a local minimum, and it is also best to resample.
- On each experiment, if under-learning occurs (the network doesn't achieve an acceptable performance level) try adding more neurons to the hidden layer(s). If this doesn't help, try adding an extra hidden layer.

- If over-learning occurs (selection error starts to rise) try removing hidden units (and possibly layers).
- Once you have experimentally determined an effective configuration for your networks, resample and generate new networks with that configuration.

- **Data Selection:** All the above stages rely on a key assumption. Specifically, the training, verification and test data must be representative of the underlying model (and, further, the three sets must be independently representative). The old computer science adage "garbage in, garbage out" could not apply more strongly than in neural modeling. If training data is not representative, then the model's worth is at best compromised. At worst, it may be useless. It is worth spelling out the kind of problems which can corrupt a training set:

The future is not the past. Training data is typically historical. If circumstances have changed, relationships which held in the past may no longer hold. All eventualities must be covered. A neural network can only learn from cases that are present. If people with incomes over $100,000 per year are a bad credit risk, and your training data includes nobody over $40,000 per year, you cannot expect it to make a correct decision when it encounters one of the previously-unseen cases. Extrapolation is dangerous with any model, but some types of neural network may make particularly poor predictions in such circumstances.
A network learns the easiest features it can. A classic (possibly apocryphal) illustration of this is a vision project designed to automatically recognize tanks. A network is trained on a hundred pictures including tanks, and a hundred not. It achieves a perfect 100% score. When tested on new data, it proves hopeless. The reason? The pictures of tanks are taken on dark, rainy days; the pictures without on sunny days. The network learns to distinguish the (trivial matter of) differences in overall light intensity. To work, the network would need training cases including all weather and lighting conditions under which it is expected to operate - not to mention all types of terrain, angles of shot, distances...
Unbalanced data sets. Since a network minimizes an overall error, the proportion of types of data in the set is critical. A network trained on a data set with 900 good cases and 100 bad will bias its decision towards good cases, as this allows the algorithm to lower the overall error (which is much more heavily influenced by the good cases). If the representation of good and bad cases is different in the real population, the network's decisions may be wrong. A good example would be disease diagnosis. Perhaps 90% of patients routinely tested are clear of a disease. A network is trained on an available data set with a 90/10 split. It is then used in diagnosis on patients complaining of specific problems, where the likelihood of disease is 50/50. The network will react over-cautiously and fail to recognize disease in some unhealthy patients. In contrast, if trained on the "complainants" data, and then tested on "routine" data, the network may raise a high number of false positives. In such circumstances, the data set may need to be crafted to take account of the distribution of data (e.g., you could replicate the less numerous cases, or remove some of the numerous cases), or the network's decisions modified by the inclusion of a loss matrix (Bishop C. M., 1995). Often, the best approach is to ensure even representation of different cases, then to interpret the network's decisions accordingly.

1.3.3 Self Organised Map

Self Organizing Feature Map (SOFM, or Kohonen) networks are used quite differently to the other networks. Whereas all the other networks are designed for supervised learning tasks, SOFM networks are designed primarily for unsupervised learning (Haykin, 19994), (Patterson, 19996), (Fausett, 19994) (Whereas in supervised learning the training data set contains cases featuring input variables together with the associated outputs (and the network must infer a mapping from the inputs to the outputs), in unsupervised learning the training data set contains only input variables. At first glance this may seem strange. Without outputs, what can the network learn? The answer is that the SOFM network attempts to learn the structure of the data.

Also Kohonen (Kohonen, 1997) explained one possible use is therefore in exploratory data analysis. The SOFM network can learn to recognize clusters of data, and can also relate similar classes to each other. The user can build up an understanding of the data, which is used to refine the network. As classes of data are recognized, they can be labelled, so that the network becomes capable of classification tasks. SOFM networks can also be used for classification when output classes are immediately available - the advantage in this case is their ability to highlight similarities between classes.

A second possible use is in novelty detection. SOFM networks can learn to recognize clusters in the training data, and respond to it. If new data, unlike previous cases, is encountered, the network fails to recognize it and this indicates novelty.

A SOFM network has only two layers: the input layer, and an output layer of radial units (also known as the topological map layer). The units in the topological map layer are laid out in space - typically in two dimensions (although *ST Neural Networks* also supports one-dimensional Kohonen networks).

SOFM networks (Patterson, 19996) are trained using an iterative algorithm. Starting with an initially-random set of radial centres, the algorithm gradually adjusts them to reflect the clustering of the training data. At one level, this compares with the sub-sampling and *K*-Means algorithms used to assign centres in SOM network and indeed the SOFM algorithm can be used to assign centres for these types of networks. However, the algorithm also acts on a different level.

The iterative training procedure also arranges the network so that units representing centres close together in the input space are also situated close together on the topological map. You can think of the network's topological layer as a crude two-dimensional grid, which must be folded and distorted into the N-dimensional input space, so as to preserve as far as possible the original structure. Clearly any attempt to represent an N-dimensional space in two dimensions will result in loss of detail; however, the technique can be worthwhile in allowing the user to visualize data which might otherwise be impossible to understand.

The basic iterative Kohonen algorithm simply runs through a number of epochs, on each epoch executing each training case and applying the following algorithm:

- Select the winning neuron (the one who's centre is nearest to the input case);
- Adjust the winning neuron to be more like the input case (a weighted sum of the old neuron centre and the training case).

The algorithm uses a time-decaying learning rate, which is used to perform the weighted sum and ensures that the alterations become more subtle as the epochs pass. This ensures that the centres settle down to a compromise representation of the cases which cause that neuron to win. The topological ordering property is achieved by adding the concept of a neighbourhood to the algorithm. The neighbourhood is a set of neurons surrounding the winning neuron. The neighbourhood, like the learning rate, decays over time, so that initially quite a large number of neurons belong to the neighbourhood (perhaps almost the entire topological map); in the latter stages the neighbourhood will be zero (i.e., consists solely of the winning neuron itself). In the Kohonen algorithm, the adjustment of neurons is actually applied not just to the winning neuron, but to all the members of the current neighbourhood.

The effect of this neighbourhood update is that initially quite large areas of the network are "dragged towards" training cases - and dragged quite substantially. The network develops a crude topological ordering, with similar cases activating clumps of neurons in the topological map. As epochs pass the learning rate and neighbourhood both decrease, so that finer distinctions within areas of the map can be drawn, ultimately resulting in fine-tuning of individual neurons. Often, training is deliberately conducted in two distinct phases: a relatively short phase with high learning rates and neighbourhood, and a long phase with low learning rate and zero or near-zero neighbourhoods.

Once the network has been trained to recognize structure in the data, it can be used as a visualization tool to examine the data. The Win Frequencies *Datasheet* (counts of the number of times each neuron wins when training cases are executed) can be examined to see if distinct clusters have formed on the map. Individual cases are executed and the topological map observed, to see if some meaning can be assigned to the clusters (this usually involves referring back to the original application area, so that the relationship between clustered cases can be established). Once clusters are identified, neurons in the topological map are labelled to indicate their meaning (sometimes individual cases may be labelled, too). Once the topological map has been built up in this way, new cases can be submitted to the network. If the winning neuron has been labelled with a class name, the network can perform classification. If not, the network is regarded as undecided.

SOFM networks also make use of the accept threshold, when performing classification. Since the activation level of a neuron in a SOFM network is the distance of the neuron from the input case, the accept threshold acts as a maximum recognized distance. If the activation of the winning neuron is greater than this distance, the SOFM network is regarded as undecided. Thus, by labelling all neurons and setting the accept threshold appropriately, a SOFM network can act as a novelty detector (it reports undecided only if the input case is sufficiently dissimilar to all radial units).

SOFM networks as expressed by Kohonen (Kohonen, 1997) are inspired by some known properties of the brain. The cerebral cortex is actually a large flat sheet (about 0.5m squared; it is folded up into the familiar convoluted shape only for convenience in fitting into the skull!) with known topological properties (for example, the area corresponding to the hand is next to the arm, and a distorted human frame can be topologically mapped out in two dimensions on its surface).

1.4 Grouping Data Using Self Organise Map

The first part of a SOM is the data. Above are some examples of 3 dimensional data which are commonly used when experimenting with SOMs. Here the colours are represented in three dimensions (red, blue, and green.) The idea of the self-organizing maps is to project the n-dimensional data (here it would be colour and would be 3 dimensions) into something that be better understood visually (in this case it would be a 2 dimensional image map).

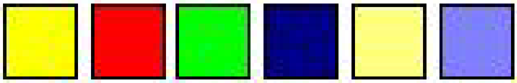


Fig. 5. Sample Data

In this case one would expect the dark blue and the greys to end up near each other on a good map and yellow close to both the red and the green. The second components to SOMs are the weight vectors. Each weight vector has two components to them which I have here attempted to show in the image below. The first part of a weight vector is its data. This is of the same dimensions as the sample vectors and the second part of a weight vector is its natural location. The good thing about colour is that the data can be shown by displaying the color, so in this case the color is the data, and the location is the x,y position of the pixel on the screen.

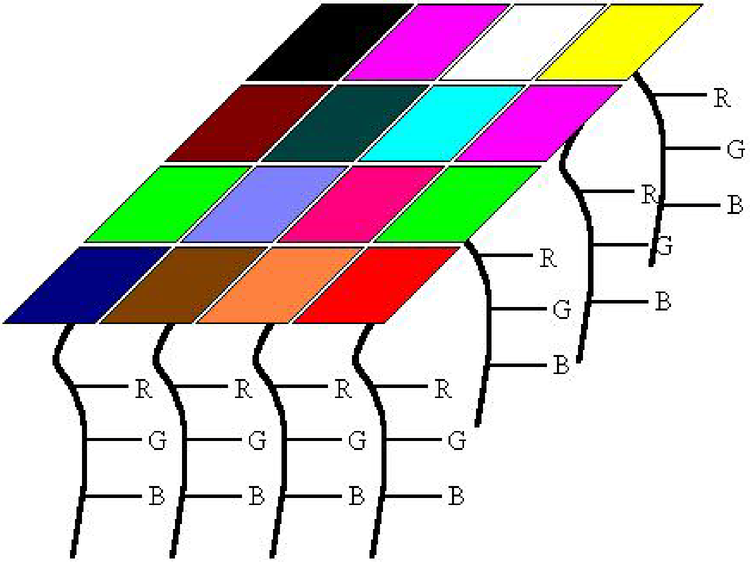


Fig. 6. 2D Array Weight of Vector

In this example, 2D array of weight vectors was used and would look like figure 5 above. This picture is a skewed view of a grid where you have the n-dimensional array for each weight and each weight has its own unique location in the grid. Weight vectors don't necessarily have to be arranged in 2 dimensions, a lot of work has been done using SOMs of 1 dimension, but the data part of the weight must be of the same dimensions as the sample vectors.Weights are sometimes referred to as neurons since SOMs are actually neural networks. SOM Algorithm. The way that SOMs go about organizing themselves is by competeting for representation of the samples. Neurons are also allowed to change themselves by learning to become more like samples in hopes of winning the next competition. It is this selection and learning process that makes the weights organize themselves into a map representing similarities.

So with these two components (the sample and weight vectors), how can one order the weight vectors in such a way that they will represent the similarities of the sample vectors? This is accomplished by using the very simple algorithm shown here.

```
Initialize Map
For t from 0 to 1
    Randomly select a sample
    Get best matching unit
    Scale neighbors
    Increase t a small amount
End for
```

Fig. 7. A Sample SOM Algorithm

The first step in constructing a SOM is to initialize the weight vectors. From there you select a sample vector randomly and search the map of weight vectors to find which weight best represents that sample. Since each weight vector has a location, it also has neighbouring weights that are close to it. The weight that is chosen is rewarded by being able to become more like that randomly selected sample vector. In addition to this reward, the neighbours of that weight are also rewarded by being able to become more like the chosen sample vector. From this step we increase *t* some small amount because the number of neighbours and how much each weight can learn decreases over time. This whole process is then repeated a large number of times, usually more than 1000 times.

In the case of colours, the program would first select a color from the array of samples such as green, then search the weights for the location containing the greenest color. From there, the colour surrounding that weight are then made more green. Then another color is chosen, such as red, and the process continues. They processes are:

- **Initializing the Weights**

  Here are screen shots of the three different ways which decided to initialize the weight vector map. We should first mention the palette here. In the java program below there are 6 intensities of red, blue, and green displayed, it really does not take away from the visual experience. The actual values for the weights are floats, so they have a bigger range than the six values that are shown in figure 7 below.

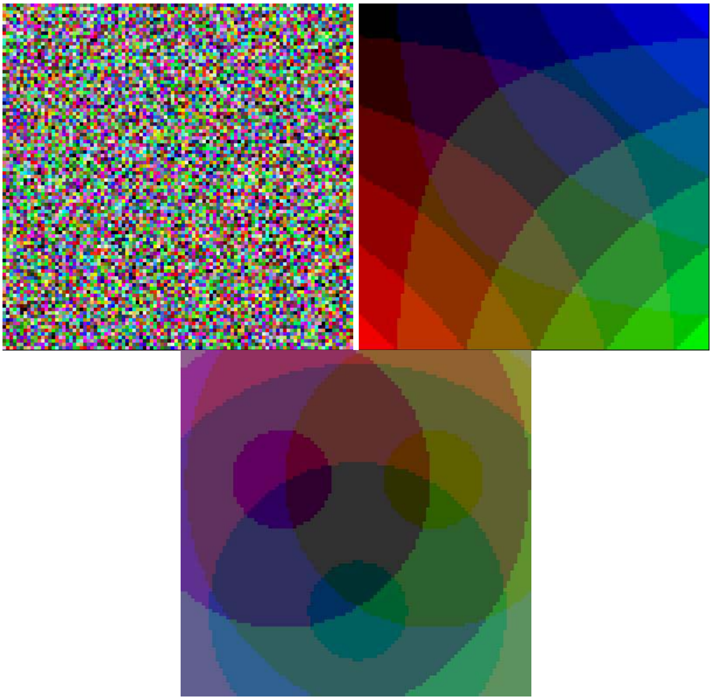


Fig. 8. Weight Values

There are a number of ways to initialize the weight vectors. The first you can see is just give each weight vector random values for its data. A screen of pixels with random red, blue, and green values is shown above on the left. Unfortunately calculating SOMs according to Kohonen (Kohonen, 1997) is very computationally expensive, so there are some variants of initializing the weights so that samples that you know for a fact are not similar start off far away. This way you need less iteration to produce a good map and can save yourself some time.

Here we made two other ways to initialize the weights in addition to the random one. This one is just putting red, blue, green, and black at all four corners and having them slowly fade toward the center. This other one is having red, green, and blue equally distant from one another and from the center.

- **B. Get Best Matching Unit**

  This is a very simple step, just go through all the weight vectors and calculate the distance from each weight to the chosen sample vector. The weight with the shortest distance is the winner. If there are more than one with the same distance, then the winning weight is chosen randomly among the weights with the shortest distance. There are a number of different ways for determining what distance actually means mathematically. The most common method is to use the Euclidean distance:

$$\sqrt{\sum_{i=0}^{n} x_i^2}$$

where *x[i]* is the data value at the *ith* data member of a sample and *n* is the number of dimensions to the sample vectors.

In the case of colour, if we can think of them as 3D points, each component being an axis. If we have chosen green which is of the value (0,6,0), the color light green (3,6,3) will be closer to green than red at (6,0,0).

Light green = Sqrt((3-0)^2+(6-6)^2+(3-0)^2) = 4.24
Red = Sqrt((6-0)^2+(0-6)^2+(0-0)^2) = 8.49

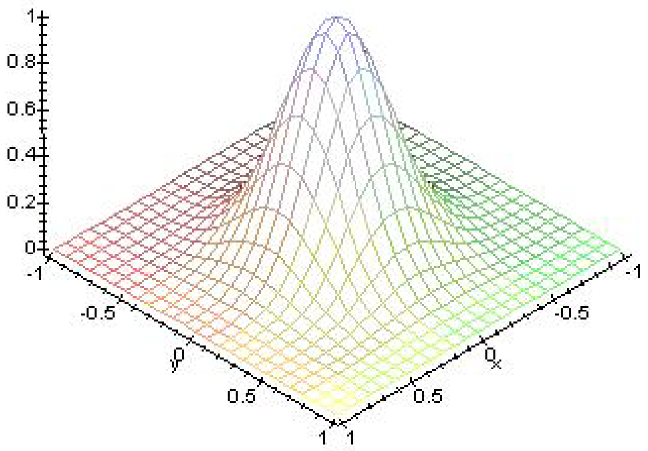
So light green is now the best matching unit, but this operation of calculating distances and comparing them is done over the entire map and the wieght with the shortest distance to the sample vector is the winner and the BMU. The square root is not computed in the java program for speed optimization for this section.

- **C. Scale Neighbors**

  1. Determining Neighbors

  There are actually two parts to scaling the neighboring weights: determining which weights are considered as neighbors and how much each weight can become more like the sample vector. The neighbors of a winning weight can be determined using a number of different methods. Some use concentric squares, others hexagons, I opted to use a gaussian function where every point with a value above zero is considered a neighbor.

  As mentioned previously, the amount of neighbors decreases over time. This is done so samples can first move to an area where they will probably be, then they jockey for position. This process is similar to coarse adjustment followed by fine tuning. The function used to decrease the radius of influence does not really matter as long as it decreases, we just used a linear function.



$$f := (x,y) \rightarrow \mathbf{e}^{\left(-6.666666667\sqrt{x^2+y^2}^{\,2}\right)}$$

Fig. 9. A graph of SOM Neighbour's determination

Figure 8 above shows a plot of the function used. As the time progresses, the base goes towards the centre, so there are less neighbours as time progresses. The initial radius is set really high, some value near the width or height of the map.

2. Learning

The second part to scaling the neighbours is the learning function. The winning weight is rewarded with becoming more like the sample vector. The neighbours also become more like the sample vector. An attribute of this learning process is that the farther away the neighbour is from the winning vector, the less it learns. The rate at which the amount a weight can learn decreases and can also be set to whatever you want. I chose to use a gaussian function. This function will return a value ranging between 0 and 1, where each neighbor is then changed using the parametric equation. The new color is:

*Current color**(1.-*t*) + *sample vector***t*

So in the first iteration, the best matching unit will get a *t* of 1 for its learning function, so the weight will then come out of this process with the same exact values as the randomly selected sample.

Just as the amount of neighbors a weight has falls off, the amount a weight can learn also decreases with time. On the first iteration, the winning weight becomes the sample vector since *t* has a full range of from 0 to 1. Then as time progresses, the winning weight becomes slightly more like the sample where the maximum value of *t* decreases. The rate at which the amount a weight can learn falls of linearly. To depict this visually, in the previous plot, the amount a weight can learn is equivalent to how high the bump is at their location. As time progresses, the height of the bump will decrease. Adding this function to the neighbourhood function will result in the height of the bump going down while the base of the bump shrinks.

So once a weight is determined the winner, the neighbours of that weight is found and each of those neighbours in addition to the winning weight change to become more like the sample vector.

1.4.1 Determining the Quality of SOMs

Below is another example of a SOM generated by the program using 500 iterations in figure 9. At first glance you will notice that similar colour is all grouped together yet again. However, this is not always the case as you can see that there are some colour who are surrounded by colour that are nothing like them at all. It may be easy to point this out with colour since this is something that we are familiar with, but if we were using more abstract data, how would we know that since two entities are close to each other means that they are similar and not that they are just there because of bad luck?

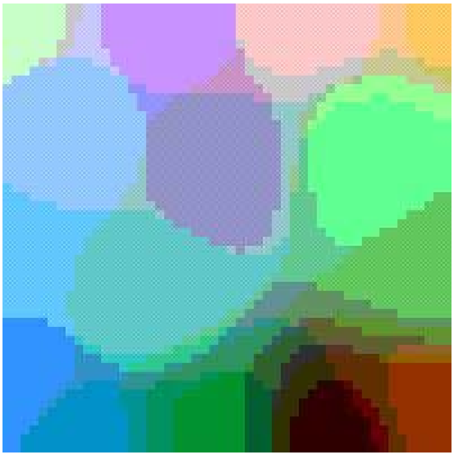


Fig. 10. SOM Iteration

There is a very simple method for displaying where similarities lie and where they do not. In order to compute this we go through all the weights and determine how similar the neighbors are. This is done by calculating the distance that the weight vectors make between the each weight and each of its neighbors. With an average of these distances a color is then assigned to that location. This procedure is located in Screen.java and named *public void update_bw()*.

If the average distance were high, then the surrounding weights are very different and a dark color is assigned to the location of the weight. If the average distance is low, a lighter color is assigned. So in areas of the center of the blobs the colour are the same, so it should be white since all the neighbors are the same color. In areas between blobs where there are similarities it should be not white, but a light grey. Areas where the blobs are physically close to each other, but are not similar at all there should be black. See Figure 8 below

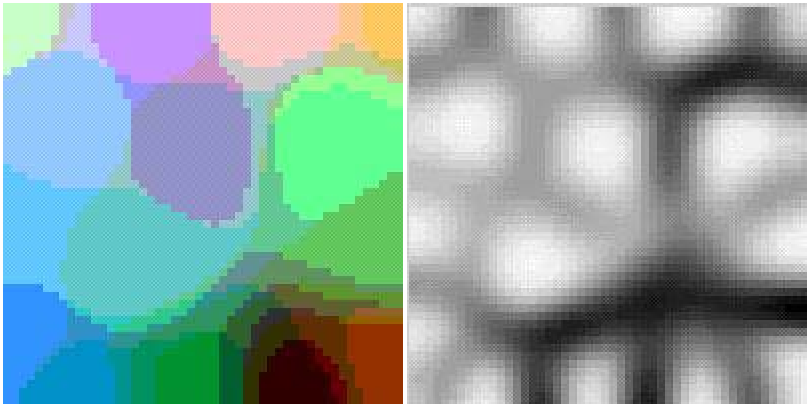


Fig. 11. A sample allocation of Weight in Colour

As shown above, the ravines of black show where the colour may be physically close to each other on the map, but are very different from each other when it comes to the actual values of the weights. Areas where there is a light grey between the blobs represent a true similarity. In the pictures above, in the bottom right there is black surrounded by colour which are not very similar to it. When looking at the black and white similarity SOM, it shows that black is not similar to the other colour because there are lines of black representing no similarity between those two colour. Also in the top corner there is pink and nearby is a light green which are not very near each other in reality, but near each other on the colored SOM. Looking at the black and white SOM, it clearly shows that the two not very similar by having black in between the two colour.

With these average distances used to make the black and white map, we can actually assign each SOM a value that determines how good the image represents the similarities of the samples by simply adding these averages.

1.4.2 Advantages and Disadvantages of SOM

Self organise map has the following advantages:

- Probably the best thing about SOMs that they are very easy to understand. It's very simple, if they are close together and there is grey connecting them, then they are similar. If there is a black ravine between them, then they are different. Unlike Multidimensional Scaling or N-land, people can quickly pick up on how to use them in an effective manner.
- Another great thing is that they work very well. As I have shown you they classify data well and then are easily evaluate for their own quality so you can actually calculated how good a map is and how strong the similarities between objects are.

These are the disadvantages:

- One major problem with SOMs is getting the right data. Unfortunately you need a value for each dimension of each member of samples in order to generate a map. Sometimes this simply is not possible and often it is very difficult to acquire all of this data so this is a limiting feature to the use of SOMs often referred to as missing data.
- Another problem is that every SOM is different and finds different similarities among the sample vectors. SOMs organize sample data so that in the final product, the samples are usually surrounded by similar samples, however similar samples are not always near each other. If you have a lot of shades of purple, not always will you get one big group with all the purples in that cluster, sometimes the clusters will get split and there will be two groups of purple. Using colour we could tell that those two groups in reality are similar and that they just got split, but with most data, those two clusters will look totally unrelated. So a lot of maps need to be constructed in order to get one final good map.
- The final major problem with SOMs is that they are very computationally expensive which is a major drawback since as the dimensions of the data increases, dimension reduction visualization techniques become more important, but unfortunately then time to compute them also increases. For calculating that black and white similarity map, the more neighbours you use to calculate the distance the better similarity map you will get, but the number of distances the algorithm needs to compute increases exponentially.

2. References

Allix, N. M. (2003, April). Epistemology And Knowledge Management Concepts And Practices. *Journal of Knowledge Management Practice* .

Alpaydin, E. (2004). *Introduction to Machine Learning.* Massachusetts, USA: MIT Press.

Anderson, J. R. (1995). *Learning and Memory.* Wiley, New York, USA.

Anil Mathur, G. P. (1999). Socialization influences on preparation for later life. *Journal of Marketing Practice: Applied Marketing Science , 5* (6,7,8), 163 - 176.

Ashby, W. R. (1960). *Design of a Brain, The Origin of Adaptive Behaviour.* John Wiley and Son.

Batista, G. &. (2003). An Analysis of Four Missing Data Treatment Methods for Suppervised Learning. *Applied Artificial Intelligence , 17*, 519-533.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford, England: Oxford University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics).* New York, New York: Springer Science and Business Media.

Block H, D. (1961). The Perceptron: A Model of Brian Functioning. *34* (1), 123-135.

Carling, A. (1992). *Introducing Neural Networks .* Wilmslow, UK: Sigma Press.

D. Michie, D. J. (1994). *Machine Learning, Neural and Statistical Classification.* Prentice Hall Inc.

Fausett, L. (19994). *Fundamentals of Neural Networks.* New York: Prentice Hall.

Forsyth, R. S. (1990). The strange story of the Perceptron. *Artificial Intelligence Review , 4* (2), 147-155.

Friedberg, R. M. (1958). A learning machine: Part, 1. *IBM Journal* , 2-13.

Ghahramani, Z. (2008). Unsupervised learning algorithms are designed to extract structure from data. *178*, pp. 1-8. IOS Press.

Gillies, D. (1996). *Artificial Intelligence and Scientific Method.* OUP Oxford.

Haykin, S. (19994). *Neural Networks: A Comprehensive Foundation.* New York: Macmillan Publishing.

Hodge, V. A. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review* , *22* (2), 85-126.

Holland, J. (1980). Adaptive Algorithms for Discovering and Using General Patterns in Growing Knowledge Bases Policy Analysis and Information Systems. *4* (3).

Hunt, E. B. (1966). Experiment in Induction.

Ian H. Witten, E. F. (2005). *Data Mining Practical Machine Learning and Techniques* (Second edition ed.). Morgan Kaufmann.

Jaime G. Carbonell, R. S. (1983). Machine Learning: A Historical and Methodological Analysis. *Association for the Advancement of Artificial Intelligence* , *4* (3), 1-10.

Kohonen, T. (1997). Self-Organizating Maps.

Luis Gonz, l. A. (2005). Unified dual for bi-class SVM approaches. *Pattern Recognition* , *38* (10), 1772-1774.

McCulloch, W. S. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophysics* , 115-133.

Mitchell, T. M. (2006). *The Discipline of Machine Learning.* Machine Learning Department technical report CMU-ML-06-108, Carnegie Mellon University.

Mooney, R. J. (2000). Learning Language in Logic. In L. N. Science, *Learning for Semantic Interpretation: Scaling Up without Dumbing Down* (pp. 219-234). Springer Berlin / Heidelberg.

Mostow, D. (1983). *Transforming declarative advice into effective procedures: a heuristic search cxamplc In I?. S. Michalski,.* Tioga Press.

Nilsson, N. J. (1982). *Principles of Artificial Intelligence (Symbolic Computation / Artificial Intelligence).* Springer.

Oltean, M. (2005). Evolving Evolutionary Algorithms Using Linear Genetic Programming. *13* (3), 387 - 410 .

Orlitsky, A., Santhanam, N., Viswanathan, K., & Zhang, J. (2005). Convergence of profile based estimators. *Proceedings of International Symposium on Information Theory. Proceedings. International Symposium on*, pp. 1843 - 1847. Adelaide, Australia: IEEE.

Patterson, D. (19996). *Artificial Neural Networks.* Singapore: Prentice Hall.

R. S. Michalski, T. J. (1983). *Learning from Observation: Conceptual Clustering.* TIOGA Publishing Co.

Rajesh P. N. Rao, B. A. (2002). *Probabilistic Models of the Brain.* MIT Press.

Rashevsky, N. (1948). *Mathematical Biophysics:Physico-Mathematical Foundations of Biology.* Chicago: Univ. of Chicago Press.

Richard O. Duda, P. E. (2000). *Pattern Classification* (2nd Edition ed.).

Richard S. Sutton, A. G. (1998). *Reinforcement Learning.* MIT Press.

Ripley, B. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain . *Psychological Review* , *65* (6), 386-408.

Russell, S. J. (2003). *Artificial Intelligence: A Modern Approach* (2nd Edition ed.). Upper Saddle River, NJ, NJ, USA: Prentice Hall.

Ryszard S. Michalski, J. G. (1955). *Machine Learning: An Artificial Intelligence Approach (Volume I).* Morgan Kaufmann .

Ryszard S. Michalski, J. G. (1955). *Machine Learning: An Artificial Intelligence Approach.*

Selfridge, O. G. (1959). Pandemonium: a paradigm for learning. *In The mechanisation of thought processes. H.M.S.O., London.* London.

Sleeman, D. H. (1983). *Inferring Student Models for Intelligent CAI. Machine Learning.* Tioga Press.

Tapas Kanungo, D. M. (2002). A local search approximation algorithm for k-means clustering. *Proceedings of the eighteenth annual symposium on Computational geometry* (pp. 10-18). Barcelona, Spain : ACM Press.

Timothy Jason Shepard, P. J. (1998). Decision Fusion Using a Multi-Linear Classifier . *In Proceedings of the International Conference on Multisource-Multisensor Information Fusion.*

Tom, M. (1997). *Machibe Learning.* Machine Learning, Tom Mitchell, McGraw Hill, 1997: McGraw Hill.

Trevor Hastie, R. T. (2001). *The Elements of Statistical Learning.* New york, NY, USA: Springer Science and Business Media.

Widrow, B. W. (2007). *Adaptive Inverse Control: A Signal Processing Approach.* Wiley-IEEE Press.

Y. Chali, S. R. (2009). Complex Question Answering: Unsupervised Learning Approaches and Experiments. *Journal of Artificial Intelligent Research* , 1-47.

Yu, L. L. (2004, October). Efficient feature Selection via Analysis of Relevance and Redundacy. *JMLR* , 1205-1224.

Zhang, S. Z. (2002). Data Preparation for Data Mining. Applied Artificial Intelligence. *17*, 375 - 381.

4

Methods for Pattern Classification

Yizhang Guan
South China University of Technology
China

1. Introduction

Pattern classification is to classify some object into one of the given categories called classes. For a specific pattern classification problem, a classifier is computer software. It is developed so that objects (x) are classified correctly with reasonably good accuracy. Through training using input-output pairs, classifiers acquire decision functions that classify an input datum into one of the given classes (ω_i). In pattern recognition applications we rarely if ever have the prior probability $P(\omega_i)$ or the class-conditional density $p(x \mid \omega_i)$. of complete knowledge about the probabilistic structure of the problem. In a typical case we merely have some vague, general knowledge about the situation, together with a number of design samples or training data—particular representatives of the patterns we want to training classify. The problem, then, is to find some way to use this information to design or data train the classifier.

The organization of this chapter is to address those cases where a great deal of information about the models is known and to move toward problems where the form of the distributions are unknown and even the category membership of training patterns is unknown. We begin in Bayes decision theory(Sec.2) by considering the ideal case in which the probability structure underlying the categories is known perfectly. In Sec.3(Maximum Likelihood) we address the case when the full probability structure underlying the categories is not known, but the general forms of their distributions are the models. Thus the uncertainty about a probability distribution is represented by the values of some unknown parameters, and we seek to determine these parameters to attain the best categorization. In Sec.4(Nonparametric techniques)we move yet further from the Bayesian ideal,and assume that we have no prior parameterized knowledge about the underlying probability structure;in essence our classification will be based on information provided by training samples alone. Classic techniques such as the nearest-neighbor algorithm and potential functions play an important role here. We then in Sec.5(Support Vector Machine) Next, in Sec.6(Nonlinear Discriminants and Neural Networks)we see how some of the ideas from such linear discriminants can be extended to a class of very powerful algorithms such as backpropagation and others for multilayer neural networks; these neural techniques have a range of useful properties that have made them a mainstay in contemporary pattern recognition research. In Sec.7(Stochastic Methods)we discuss simulated annealing by the Boltzmann learning algorithm and other stochastic methods. We explore the behaviour of such algorithms with regard to the matter of local minima that can plague other neural methods. Sec.8(Unsupervised Learning and

Clustering),by addressing the case when input training patterns are not labelled, and that our recognizer must determine the cluster structure.

2. Bayesian Decision Theory

Suppose that we know both the prior probabilities $P(\omega_j)$ and the conditional densities $p(x \mid \omega_j)$. Suppose further that we measure the features of a sample and discover that its value is x. How does this measurement influence our attitude concerning the true state of nature—that is, the category of the input? We note first that the(joint) probability density of finding a pattern that is in category ω_j and has feature value x can be written in two ways: $P(\omega_j, x) = P(\omega_j \mid x)p(x) = p(x \mid \omega_j)P(\omega_j)$. Rearranging these leads us to the answer to our question, which is called Bayes formula:

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)} \tag{1}$$

where in this case of c categories

$$p(x) = \sum_{j=1}^{c} p(x \mid \omega_j)P(\omega_j) \tag{2}$$

2.1 Two-Category Classification

If we have an observation x for which $P(\omega_1 \mid x)$ is greater than $P(\omega_2 \mid x)$, we would naturally be inclined to decide that the true state of nature is ω_1. Similarly, if $P(\omega_2 \mid x)$ is greater than $P(\omega_1 \mid x)$, we would be inclined to choose ω_2. Thus we have justified the following Bayes decision rule for minimizing the probability of error:

$$\textit{Decide} \;\; \omega_1 \;\; \textit{if} \;\; P(\omega_1 \mid x) > P(\omega_2 \mid x), \;\; \textit{otherwise} \;\; \textit{decide} \;\; \omega_2 \tag{3}$$

In Eq. (1), $p(x)$ is a scale factor and unimportant for our problem. By using Eq.(1), we can instead express the rule in terms of the conditional and prior probabilities. And we notice $P(\omega_1 \mid x) + P(\omega_2 \mid x)) = 1$. By eliminating this scale factor, we obtain the following completely equivalent decision rule:

$$\textit{Decide} \;\; \omega_1 \;\; \textit{if} \;\; p(x \mid \omega_1)P(\omega_1) > p(x \mid \omega_2)P(\omega_2), \;\; \textit{otherwise} \;\; \textit{decide} \;\; \omega_2 \tag{4}$$

While the two-category case is just a special instance of the multi-category case, it has traditionally received separate treatment.Indeed,a classifier that places a pattern in one of only two categories has a special name—a dichotomizer. Instead of using two dichotomizer discriminant functions g_1 and g_2 and assigning x toω1 if $g_1 > g_2$, it is more common to define a single discriminant function

$$g(x) = g_1(x) - g_2(x) \tag{5}$$

and to use the following decision rule:

$$Decide\ \ \omega_1\ \ if\ \ g(x) > 0,\ \ otherwise\ \ decide\ \ \omega_2$$

Thus, a dichotomizer can be viewed as a machine that computes a single discriminant function $g(x)$, and classifies x according to the algebraic sign of the result. Of the various forms in which the minimum-error-rate discriminant function can be written, the following two(derived from Eqs.(1)&(5)are particularly convenient:

$$g(x) = P(\omega_1 \mid x) - P(\omega_2 \mid x) \tag{6}$$

$$g(x) = \ln \frac{p(x \mid \omega_1)}{p(x \mid \omega_2)} + \ln \frac{p(\omega_1)}{p(\omega_2)} \tag{7}$$

2.2 Multi-Category Classification

Let $\omega_1, \cdots, \omega_c$ be the finite set of c states of nature. Let the feature vector x be a d -component vector-valued random variable, and let $P(x \mid \omega_j)$ be the state- conditional probability density function for x —the probability density function for x conditioned on ω_j being the true state of nature. As before, $P(\omega_j)$ describes the prior probability that nature is in state ω_j . Then the posterior probability $P(\omega_j \mid x)$ can be computed from $p(x \mid \omega_j)$ by Bayes formula:

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)} \tag{8}$$

where the evidence is now

$$p(x) = \sum_{j=1}^{c} p(x \mid \omega_j) P(\omega_j) \tag{9}$$

A Bayes classifier is easily and naturally represented in this way. For the minimum-error-rate case, we can simplify things further by taking gi(x)=P(ωi|x),so that the maximum discriminant function corresponds to the maximum posterior probability.

Clearly, the choice of discriminant functions is not unique. We can always multiply all the discriminant functions by the same positive constant or shift them by the same additive constant without influencing the decision. More generally, if we replace every $g\ (x)$ by $f(g\ (x))$, where $f(\cdot)$ is a monotonically increasing function, the resulting classification is unchanged. This observation can lead to significant analytical and computational simplifications. In particular, for minimum-error-rate classification, any of the following choices gives identical classification results, but some can be much simpler to understand or to compute than others:

$$g\ (x) = P(\omega_i \mid x) = \frac{p(x \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(x \mid \omega_j)P(\omega_j)} \tag{10}$$

$$g\ (x) = P(\omega_i \mid x) = p(x \mid \omega_i)P(\omega_i) \tag{11}$$

$$g\ (x) = P(\omega_i \mid x) = \ln p(x \mid \omega_i) + \ln P(\omega_i) \tag{12}$$

where ln denotes natural logarithm.

3. Maximum-likelihood Method

It is important to distinguish between supervised learning and unsupervised learning. In both cases, samples x are assumed to be obtained by selecting a state of nature ω_i with probability $P(\omega_i)$,and then independently selecting x according to the probability law $p(x \mid \omega_i)$. The distinction is that with supervised learning we know the state of nature(class label)for each sample, whereas with unsupervised learning we do not. As one would expect, the problem of unsupervised learning is the more difficult one. In this section we shall consider only the supervised case, deferring consideration of unsupervised learning to Section 8.

The problem of parameter estimation is a classical one in statistics, and it can be approached in several ways. We shall consider two common and reasonable procedures, maximum likelihood estimation and Bayesian estimation. Although the results obtained with these two procedures are frequently nearly identical, the approaches are conceptually quite different. Maximum likelihood and several other methods view the parameters as quantities whose values are fixed but unknown. The best estimate of their value is defined to be the one that maximizes the probability of obtaining the samples actually observed. In contrast, Bayesian methods view the parameters as random variables having some known a priori distribution. Observation of the samples converts this to a posterior density, thereby revising our opinion about the true values of the parameters. In the Bayesian case, we shall see that a typical effect of observing additional samples is to sharpen the a posteriori density function, causing it to peak near the true values of the parameters. This phenomenon is known as Bayesian learning. In either case, we use the posterior densities for our classification rule, as we have seen before.

3.1 Maximum Likelihood

Maximum likelihood estimation methods have a number of attractive attributes. First, they nearly always have good convergence properties as the number of train- ing samples increases. Further, maximum likelihood estimation often can be simpler than alternate methods, such as Bayesian techniques or other methods presented in subsequent section.

Suppose that we separate a collection of samples according to class, so that we have c sets, $D_1, \cdots, D_c$, with the samples in D_i having been drawn independently according to the probability law $p(x \mid \omega_i)$. We say such samples are i.i.d.—independent identically distributed random variables. We assume that $p(x \mid \omega_i)$ has a known parametric form, and is therefore determined uniquely by the value of a parameter vector θ_i. For example, we might have $p(x \mid \omega_i) \sim N(\mu_i, \Sigma_i)$, where θ_i consists of the components of μ_i and Σ_i. To show the dependence of $p(x \mid \omega_i)$ on θ_i explicitly, we write $p(x \mid \omega_i)$ as $p(x \mid \omega_i, \theta_i)$. Our problem is to use the information provided by the training samples to obtain good estimates for the unknown parameter vectors $\theta_1, \cdots, \theta_c$ associated with each category.

To simplify treatment of this problem, we shall assume that samples in D_i give no information about θ_i if $i = j$ —that is, we shall assume that the parameters for the different classes are functionally independent. This permits us to work with each class separately, and to simplify our notation by deleting indications of class distinctions. With this assumption we thus have c separate problems of the following form: Use a set D of training samples drawn independently from the probability density p(x | θ)to estimate the unknown parameter vector θ.

Suppose that D contains n samples, $x_1, \cdots, x_n$. Then, since the samples were drawn independently, we have

$$p(D \mid \theta) = \prod_{k=1}^{n} p(x_k \mid \theta) \tag{13}$$

Viewed as a function of θ, $p(D \mid \theta)$ is called the likelihood of θ with respect to the set of samples. The maximum likelihood estimate of θ is, by definition, the value θ that maximizes $p(D \mid \theta)$. Intuitively, this estimate corresponds to the value of θ that in some sense best agrees with or supports the actually observed training samples.

For analytical purposes, it is usually easier to work with the logarithm of the likelihood than with the likelihood itself. Since the logarithm is monotonically increasing, the θ that maximizes the log-likelihood also maximizes the likelihood. If $p(D \mid \theta)$ is a well behaved, differentiable function of θ, θ can be found by the standard methods of differential calculus. If the number of parameters to be set is p, then we let θ denote the p-component vector $\theta = (\theta_1, \cdots, \theta_p)^T$, and ∇ be the gradient operator

$$\nabla_\theta = \begin{bmatrix} \dfrac{\partial}{\partial \theta_1} \\ \vdots \\ \dfrac{\partial}{\partial \theta_p} \end{bmatrix} \tag{14}$$

We define $L(\theta)$ as the log-likelihood function?

$$L(\theta) = \ln p(D \mid \theta) \tag{15}$$

We can then write our solution formally as the argument θ that maximizes the log-likelihood, i.e.,

$$\hat{\theta} = \arg\max_{\theta} L(\theta) \tag{16}$$

where the dependence on the data set D is implicit. Thus we have from Eq.(13)

$$L(\theta) = \sum_{k=1}^{n} \ln p(x_k \mid \theta) \tag{17}$$

and

$$\nabla_{\theta} L = \sum_{k=1}^{n} \nabla_{\theta} \ln p(x_k \mid \theta) \tag{18}$$

Thus, a set of necessary conditions for the maximum likelihood estimate for θ can be obtained from the set of p equations

$$\nabla_{\theta} L = 0 \tag{19}$$

A solution θ to Eq.(19) could represent a true global maximum, a local maximum or minimum, or(rarely)an inflection point of $L(\theta)$. One must be careful, too, to check if the extremum occurs at a boundary of the parameter space, which might not be apparent from the solution to Eq.(19). If all solutions are found, we are guaranteed that one represents the true maximum, though we might have to check each solution individually(or calculate second derivatives)to identify which is the global optimum. Of course, we must bear in mind that θ is an estimate; it is only in the limit of an infinitely large number of training points that we can expect that our estimate will equal to the true value of the generating function.

3.2 Bayesian estimation

We now consider the Bayesian estimation or Bayesian learning approach to pattern classification problems. Although the answers we get by this method will generally be nearly identical to those obtained by maximum likelihood, there is a conceptual difference: whereas in maximum likelihood methods we view the true parameter vector we seek, θ, to be fixed, in Bayesian learning we consider θ to be a random variable, and training data allows us to convert a distribution on this variable into a posterior probability density.

The computation of the posterior probabilities $P(\omega_i \mid x)$ lies at the heart of Bayesian classification. Bayes formula allows us to compute these probabilities from the prior probabilities $P(\omega_i)$ and the class-conditional densities $p(x \mid \omega_i)$, but how can we proceed when these quantities are unknown? The general answer to this question is that the best we can do is to compute $P(\omega_i \mid x)$ using all of the information at our disposal. Part of this information might be prior knowledge, such as knowledge of the functional forms for unknown densities and ranges for the values of unknown parameters. Part of this information might reside in a set of training samples. If we again let D denote the set of samples, then we can emphasize the role of the samples by saying that our goal is to compute the posterior probabilities $P(\omega_i \mid x, D)$. From these probabilities we can obtain the Bayes classifier.

Given the sample D, Bayes formula then becomes

$$P(\omega_i \mid x, D) = \frac{p(x \mid \omega_i, D)P(\omega_i \mid D)}{\sum_{j=1}^{c} p(x \mid \omega_j, D)P(\omega_j \mid D)} \tag{20}$$

As this equation suggests, we can use the information provided by the training samples to help determine both the class-conditional densities and the a priori probabilities.

Although we could maintain this generality, we shall henceforth assume that the true values of the a priori probabilities are known or obtainable from a trivial calculation; thus we substitute $P(\omega_i) = P(\omega_i \mid D)$. Furthermore, since we are treating the supervised case, we can separate the training samples by class into c subsets $D_1, \cdots, D_c$ with the samples in D_i belonging to ω_i . As we mentioned when addressing maximum likelihood methods, in most cases of interest(and in all of the cases we shall consider), the samples in D_i have no influence on $p(x \mid \omega_j, D)$ if $i = j$. This has two simplifying consequences. First, it allows us to work with each class separately, using only the samples in D_i to determine $p(x \mid \omega_i, D)$. Used in conjunction with our assumption that the prior probabilities are known, this allows us to write Eq. 23 as

$$P(\omega_i \mid x, D) = \frac{p(x \mid \omega_i, D_j)P(\omega_i)}{\sum_{j=1}^{c} p(x \mid \omega_j, D_j)P(\omega_j)} \tag{21}$$

Second, because each class can be treated independently, we can dispense with needless class distinctions and simplify our notation. In essence, we have c separate problems of the following form: use a set D of samples drawn independently according to the fixed but unknown probability distribution p(x)to determine $p(x \mid D)$ .This is the central problem of Bayesian learning.

4. Nonparametric Techniques

We treat supervised learning under the assumption that the forms of the underlying density functions are known in the last section. But in most pattern recognition applications, the common parametric forms rarely fit the densities. In this section we shall examine nonparametric procedures that can be used with arbitrary distributions and without the assumption that the forms of the underlying densities are known.

There are several types of nonparametric methods of interest in pattern recognition. One is to estimate the density functions $p(x \mid \omega_j)$ from sample. And it can be substituted for the true densities. Another is to estimate a posteriori probabilities $P(\omega_j \mid x)$ directly. such as the nearest-neighbor rule Finally, there are nonparametric procedures for transforming the feature space in the hope that it may be possible to employ parametric methods in the transformed space.

The following obvious estimate for $p(x)$:

$$p(x) = \frac{k_n / n}{V_n} \tag{22}$$

4.1 Parzen Windows

Assume that the region R^n is a d-dimensional hypercube. hn is the length of an edge of that hypercube, then its volume is given by

$$V_n = h_n^d \tag{23}$$

Define the window function as

$$\varphi(u) = \begin{cases} 1 & |u_j| \le 1/2 \\ 0 & otherwise \end{cases} \qquad j = 1, \cdots, d \tag{24}$$

The number of samples in this hypercube is given by

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{x - x_i}{h_n} \right) \tag{25}$$

Substitute this into Eq (22). we obtain the estimate

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi\left(\frac{x - x_i}{h_n} \right) \tag{26}$$

Eq.(26) expresses the estimate for $p(x)$ as an average of functions of x and the samples xi. In essence, the window function is being used for interpolation—each sample contributing to the estimate in accordance with its distance from x.

It is natural to ask that the estimate $p_n(x)$ be a legitimate density function. We can require that

$$\varphi(x) \ge 0\,, \tag{27}$$

and

$$\int \varphi(u) du = 1 \tag{28}$$

Maintain the relation $V_n = h_n^d$, then the $p_n(x)$ also satisfies these conditions.

Define $\delta_n(x)$ by

$$\delta_n(x) = \frac{1}{V_n} \varphi\left(\frac{x}{h_n} \right) \tag{29}$$

Then $p_n(x)$ can be written as the average

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \delta_n(x - x_i) \tag{30}$$

Since $V_n = h_n^d$, h_n clearly affects both the amplitude and the width of $\delta_n(x)$. If h_n is very large, the amplitude of $\delta_n(x)$ is small, and x must be far from xi before $\delta_n(x - x_i)$ changes much from $\delta_n(0)$ .In this case, pn(x) is the superposition of n broad, slowly changing functions and is a very smooth "out-of-focus" estimate of $p(x)$. On the other hand, if h_n is very small, the peak value of $\delta_n(x - x_i)$ is large and occurs near x=xi. In this case $p(x)$ is the superposition of n sharp pulses centered at the samples—an erratic, "noisy" estimate. For any value of h_n, the distribution is normalized, that is

$$\int \delta_n(x - x_i)dx = \int \frac{1}{V_n} \varphi\left(\frac{x - x_i}{h_n}\right)dx = \int \varphi(u)du = 1 \tag{31}$$

Let V_n slowly approach zero as n increases and $p_n(x)$ converges to the unknown density p(x). $p_n(x)$ has some mean $\overline{p}_n(x)$ and variance $\sigma_n^2(x)$. $p_n(x)$ converges to p(x) if

$$\lim_{n\to\infty} \overline{p}_n(x) = p(x) \tag{32}$$

and

$$\lim_{n\to\infty} \sigma_n^2(x) = 0 \tag{33}$$

To prove convergence we must place conditions on the unknown density $p(x)$, on the window function $\varphi(u)$, and on the window width h_n. In general, continuity of $p(\cdot)$ at x is required, and the conditions imposed by Eqs.(27)&(28) are customarily invoked. With care, it can be shown that the following additional conditions assure convergence:

$$\sup_{\mu} \varphi(u) < \infty \tag{34}$$

$$\lim_{\|\mu\|\to\infty} \varphi(u) \prod_{i=1}^{d} \mu_i = 0 \tag{35}$$

$$\lim_{n\to\infty} V_n = 0 \tag{36}$$

$$\lim_{n\to\infty} nV_n = \infty \tag{37}$$

Equations (34)&(35) keep $\varphi(\cdot)$ well behaved, and are satisfied by most density functions that one might think of using for window functions. Equations (36)&(37) state that the volume V_n must approach zero, but at a rate slower than $1/n$. We shall now see why these are the basic conditions for convergence.

4.2 k_n –Nearest-Neighbor Estimation

A potential remedy for the problem of the unknown "best" window function is to let the cell volume be a function of the training data, rather than some arbitrary function of the overall number of samples. For example, to estimate p(x)from n training samples or prototypes we can center a cell about x and let it grow until I captures k_n samples, where k_n is some specified function of n. These samples are the k_n nearest-neighbors of x . It the density is high near x , the cell will be relatively small, which leads to good resolution. If the density is low, it is true that the cell will grow large, but it will stop soon after it enters regions of higher density. In either case, if we take

$$p(x) = \frac{k_n / n}{V_n} \tag{38}$$

we want k_n to go to infinity as n goes to infinity, since this assures us that k_n / n will be a good estimate of the probability that a point will fall in the cell of volume V_n . However, we also want k_n to grow sufficiently slowly that the size of the cell needed to capture k_n training samples will shrink to zero. Thus, it is clear from Eq.(38) that the ratio k_n / n must go to zero. Although we shall not supply a proof, it can be shown that the conditions $\lim_{n\to\infty} k_n = \infty$ and $\lim_{n\to\infty} k_n / n = 0$ are necessary and sufficient for $p_n(x)$ to converge to p(x)in probability at all points where $p(x)$ is continuous. If we take $k_n = \sqrt{n}$ and assume that $p_n(x)$ is a reasonably good approximation to $p(x)$ we then see from Eq.(38) that $V_n = 1/(\sqrt{n}\, p(x))$. Thus, V_n again has the form $V_1 / \sqrt{n}$, but the initial volume V_1 is determined by the nature of the data rather than by some arbitrary choice on our part. Note that there are nearly always discontinuities in the slopes of these estimates, and these lie away from the prototypes themselves.

5. Support Vector Machine

In a support vector machine, the direct decision function that maximizes the generalization ability is determined for a two-class problem. Assuming that the training data of different classes do not overlap, the decision function is determined so that the distance from the training data is maximized. We call this the optimal decision function. Because it is difficult to determine a nonlinear decision function, the original input space is mapped into a high-dimensional space called feature space. And in the feature space, the optimal decision function, namely, the optimal hyper-plane is determined.

Support vector machines outperform conventional classifiers, especially when the number of training data is small and the number of input variables is large. This is because the conventional classifiers do not have the mechanism to maximize the margins of class boundaries. Therefore, if we introduce some mechanism to maximize margins, the generalization ability is improved.

If the decision function is linear, namely, $g_i(x)$ is given by

$$g_i(x) = w^T x + b \tag{39}$$

where w is an m-dimensional vector and b is a bias term, and if one class is on the positive side of the hyper-plane, i.e., $g_i(x) > 0$, and the other class is on the negative side, the given problem is said to be linearly separable.

5.1 Indirect Decision Functions

For an $n(>2)$ -class problem, suppose we have indirect decision functions $g_i(x)$ for classes i. To avoid unclassifiable regions, we classify x into class j given by

$$j = \arg\max_i g_i(x) \tag{40}$$

where arg returns the subscript with the maximum value of $g_i(x)$ .If more than one decision function take the same maximum value for x, namely, x is on the class boundary, it is not classifiable.

In the following we discuss several methods to obtain the direct decision functions for multi-class problems.

The first approach is to determine the decision functions by the one-against-all formulation. We determine the i th decision function $g_i(x)$ $i = 1, \cdots, n$, so that when x belongs to class i,

$$g_i(x) > 0 \tag{41}$$

and when x belongs to one of the remaining classes,

$$g_i(x) < 0 \tag{42}$$

When x is given, we classify x into class i if $g_i(x) > 0$ and $g_j(x) < 0$ $j \neq i, j = 1, \cdots, n$. But by these decision functions, unclassifiable regions exist when more than one decision function are positive or no decision functions are positive.

The second approach is based on a decision tree. It is considered to be a variant of one-against-all formulation. We determine the i th decision function $g_i(x)$ $i = 1, \cdots, n$, so that when x belongs to class i,

$$g_i(x) > 0 \tag{43}$$

and when x belongs to one of the classes $i + 1, \cdots, n$,

$$g_i(x) < 0 \tag{44}$$

In classifying x, starting from $g_1(x)$, we find the first positive $g_i(x)$ and classify x into class i. If there is no such i among $g_i(x)$ ($i = 1, \cdots, n$), we classify x into class n.

The decision functions change if we determine decision functions in descending order or in an arbitrary order of class labels. Therefore, in this architecture, we need to determine the decision functions so that classification performance in the upper level of the tree is more accurate than in the lower one. Otherwise, the classification performance may not be good.

Pair-wise Formulation

The third approach is to determine the decision functions by pair-wise formulation. For classes i and j we determine the decision function $g_{ij}(x)$ ($i \neq j,,i,j = 1,\cdots,n$, so that

$$g_{ij}(x) > 0 \tag{45}$$

when x belongs to class i and

$$g_{ij}(x) > 0 \tag{46}$$

when x belongs to class j .

In this formulation, $g_{ij}(x) = -g_{ji}(x)$, and we need to determine n(n?1)/2 decision functions. Classification is done by voting, namely, we calculate $g_i(x)$

$$g_i(x) = \sum_{j \neq i, j=1}^{n} sign(g_{ij}(x)) \tag{47}$$

where

$$sign(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{48}$$

and we classify x into the class with the maximum $g_i(x)$. By this formulation also, unclassifiable regions exist if $g_i(x)$ take the maximum value for more than one class. These can be resolved by decision-tree formulation or by introducing membership functions.

The fourth approach is to use error-correcting codes for encoding outputs. One-against-all formulation is a special case of error-correcting code with no error-correcting capability, and so is pair-wise formulation, as if"don't"care bits are introduced.

The fifth approach is to determine decision functions at all once. Namely, we determine the decision functions $g_i(x)$ by

$$g_i(x) > g_j(x) \text{ for } j = i, j = 1,\cdots,n \tag{49}$$

In this formulation we need to determine n decision functions at all once. This results in solving a problem with a larger number of variables than the previous methods. Unlike one-against-all and pair-wise formulations, there is no unclassifiable region.

Determination of decision functions using input-output pairs is called training. In training a multilayer neural network for a two-class problem, we can determine a direct decision function if we set one output neuron instead of two. But because for an n-class problem we set n output neurons with the i th neuron corresponding to the class i decision function, the obtained functions are indirect. Similarly, decision functions for fuzzy classifiers are indirect because membership functions are defined for each class. Conventional training methods determine the indirect decision functions so that each training input is correctly classified into the class designated by the associated training output. Assuming that the circles and rectangles are training data for Classes 1 and 2, respectively, even if the decision function $g_2(x)$ moves to the right as shown in the dotted curve, the training data are still correctly classified. Thus there are infinite possibilities of the positions of the decision functions that correctly classify the training data. Although the generalization ability is directly affected by the positions, conventional training methods do not consider this.

5.2 Linear Learning Machines

In training a classifier, usually we try to maximize classification performance for the training data. But if the classifier is too fit for the training data, the classification ability for unknown data, i.e., the generalization ability is degraded. This phenomenon is called over-fitting. Namely, there is a trade-of between the generalization ability and fitting to the training data. For a two-class problem, a support vector machine is trained so that the direct decision function maximizes the generalization ability. Namely, the m -dimensional input space x is mapped into the l -dimensional($l \geq m$)feature space z . Then in z , the quadratic programming problem is solved to separate two classes by the optimal separating hyper-plane. One of the main ideas is, like support vector machines, to add a regularization term, which controls the generalization ability, to the objective function.

Let M m -dimensional training inputs x_i ($i = 1, \cdots, M$) belong to Class 1 or 2 and the associated labels be $y_i = 1$ for Class 1 and -1 for Class 2. If these data are linearly separable, we can determine the decision function:

$$D(x) = w^T x_i + b \tag{50}$$

where w is an m-dimensional vector, b is a bias term, and for $i = 1, \cdots, M$

$$w^T x_i + b \begin{cases} > 0 & for \quad y_i = 1 \\ < 0 & for \quad y_i = -1 \end{cases} \tag{51}$$

Because the training data are linearly separable, no training data satisfy $w^T x_i + b = 0$. Thus, to control separability, instead of(51),we consider the following inequalities:

$$w^T x_i + b \begin{cases} \geq 1 & for \quad y_i = 1 \\ \leq -1 & for \quad y_i = -1 \end{cases} \tag{52}$$

Equation(2.3)is equivalent to

$$y_i (w^T x_i + b) \geq 1 \text{ for } i = 1, \cdots, M \tag{53}$$

The hyper-plane

$$D(x) = w^T x_i + b = c \text{ for } -1 < c < 1 \tag{54}$$

forms a separating hyper-plane that separates x_i ($i = 1, \cdots, M$). When $c = 0$, the separating hyper-plane is in the middle of the two hyper-planes with $c = 1$ and > 1. The distance between the separating hyper-plane and the training datum nearest to the hyper-plane is called the margin. Assuming that the hyper-planes $D(x) = 1$ and > 1 include at least one training datum, the hyper-plane $D(x) = 0$ has the maximum margin for $-1 < c < 1$. The region $\{x \mid -1 \le D(x) \le 1\}$ is the generalization region for the decision function.

Now consider determining the optimal separating hyper-plane. The Euclidean distance from a training datum x to the separating hyper-plane is given by $| D(x) | / w$. Because the vector w is orthogonal to the separating hyper-plane, the line that goes through x and that is orthogonal to the hyper-plane is given by $aw/ \| w \| + x$, where $| a |$ is the Euclidean distance from x to the hyper-plane. It crosses the hyper-plane at the point where

$$D(aw/ \| w \| + x) = 0 \tag{55}$$

is satisfied. Solving(2.6)for a, we obtain $a = -D(x) / w$.

Then all the training data must satisfy

$$\frac{y_k D(x_k)}{\| w \|} \ge \delta \text{ for } k = 1, \cdots, M \tag{56}$$

Where δ is the margin.

Now if (w, b) is a solution, (aw, ab) is also a solution, where a is a scalar. Thus we impose the following constraint:

$$\delta \cdot w = 1 \tag{57}$$

From (56) and (57),to find the optimal separating hyper-plane, we need to find w with the minimum Euclidean norm that satisfies(52). Therefore, the optimal separating hyper-plane can be obtained by minimizing

$$Q(w) = \frac{1}{2} \| w \|^2 \tag{58}$$

with respect to w and b subject to the constraints:

$$y_i (w^T x_i + b) \ge 1 \text{ for } i = 1, \cdots, M \tag{59}$$

Here, the square of the Euclidean norm w in (58) is to make the optimization problem quadratic programming. The assumption of linear separability means that there exist w and b that satisfy (59).We call the solutions that satisfy (2.10) feasible solutions. Because the optimization problem has the quadratic objective function with the inequality constraints, even if the solutions are non-unique, the value of the objective function is unique(see Section 2.6.4).Thus non-uniqueness is not a problem for support vector machines. This is one of the advantages of support vector machines over neural networks, which have numerous local minima.

5.3 SVM and Change-point Theory

To detect the change-points in signal data is an important practical problem. The classical method to solve this problem is using the statistical algorithms which are based on Bayesian theory. The efficiency of these methods always depends on the character of the given data. In this paper, we introduce support vector machine method to detect the abrupt change on signal data. The experience shows that the idea is effective, and it does not limit to the character of the distribution.

Consider time series $x_1^{(1)}, x_2^{(1)}, \cdots, x_m^{(1)}$ and time series $x_{m+1}^{(2)}, x_{m+2}^{(2)}, \cdots, x_n^{(2)}$, and assume some character has been change during $x_m^{(1)}$ to $x_{m+1}^{(2)}$, that means the first m points submit to the distribution (1), and the following data occur with the other distribution. The time series is recombination like this:

$$\begin{aligned} y_1^{(k)} &= (x_1^{(1)}, x_2^{(1)}, \cdots, x_k^{(1)}), \\ y_2^{(k)} &= (x_2^{(1)}, x_3^{(1)}, \cdots, x_{k+1}^{(1)}), \\ &\vdots \end{aligned} \tag{60}$$

where k is greatly less than m and $n-m$.

We assume that the change-point is at m , and the changing information has been distributed to several distinct and continuous y_i 's. the vectors contain only the point on state (1) and state (2) are classifiable since the different distributions. Of course, it seems arbitrary here. We must do a great deal of experience to support this point of view.

In this paper, we illustrate our method to detect change-point with support vector machine method firstly in next section. The result of experience shows that the method is effective. And in the third section, we discuss on the detail of method. After that, the last section is our conclusion. .

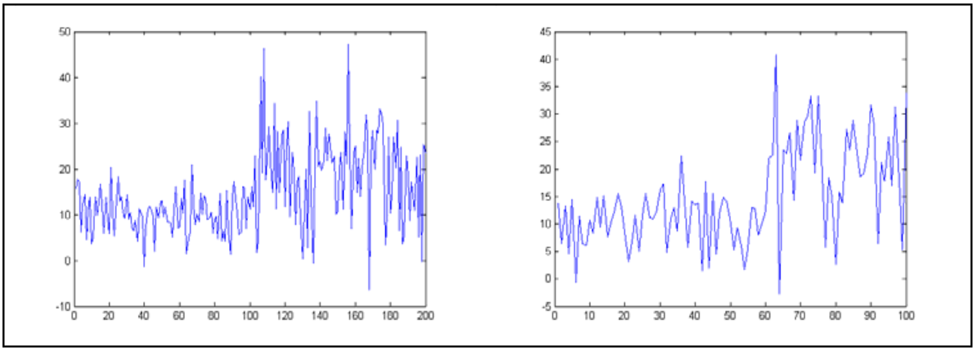


Fig. 1. Plots of train data and test data

The method is still effective for classifying the samples from the distributions with distinct variance. Let us consider a new simulation data set. The randomizer produces 100 samples that submit to $N(\mu_1,\sigma_1^2)$ and the others that submit to $N(\mu_2,\sigma_2^2)$. Let $\mu_1 = 10$, $\mu_2 = 20$, $\sigma_1^2 = 5$, $\sigma_2^2 = 10$. The randomizer produce 100 samples under two group of parameter respectively. These samples are taken as training data set. Repeat the steps to produce more 60 and 40 samples under the different group of parameter. These samples act as test data set. The values of samples are described by figure below.

The result of output file is list in the table below. The terms of columns are: (1) Dimension; (2) Accuracy on test set; (3) support vectors; (4) Count of incorrect samples; and (5) Mean Squared Error.

k	Accuracy	SV	incorrect	MSE
2	88.78%	100	11	0.890499
3	90.63%	85	9	1.687711
4	94.68%	74	5	1.437032
5	96.74%	68	3	1.260453
6	97.78%	62	2	1.098423
7	97.73%	57	2	0.859724
8	100.00%	55	0	0.814199
9	100.00%	52	0	0.727360
10	100.00%	53	0	0.754256

Table 1. Result of experience

We are interested in considering the location of the incorrect samples. Figure 4 tell us the information.

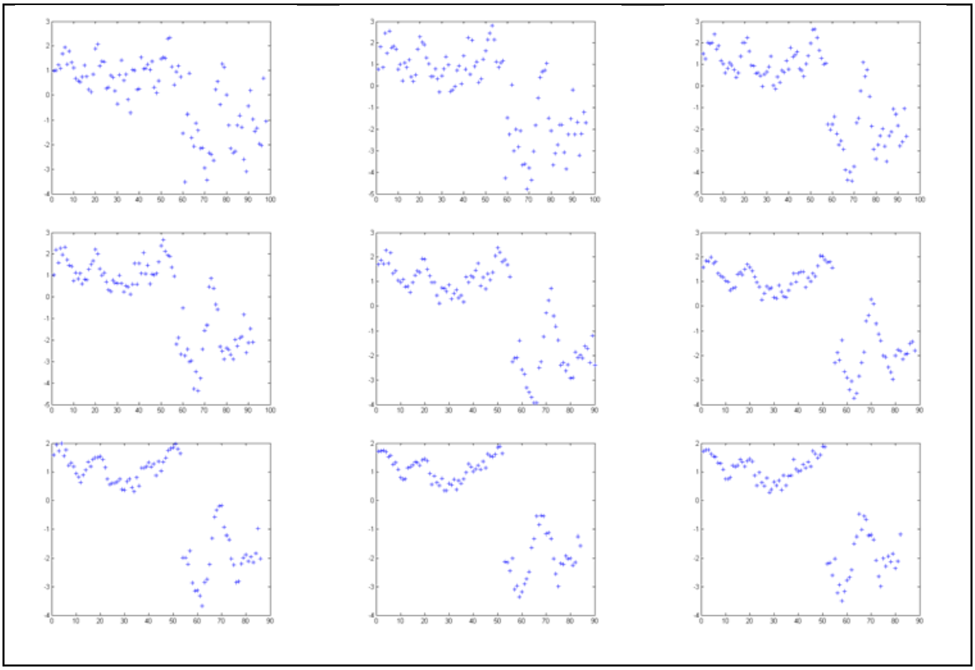


Fig. 2. Predictions with SVMs which k vary from 2 to 10

To detect the change-points in signal processing is an important practical problem. The classical method to solve this problem is using the statistical algorithms which are based on Bayesian theory. The efficiency of these methods always depends on the character of the given data. In this paper, we introduce support vector machine method to detect the abrupt change on signal data. A change-point detecting problem is transformed to a classification problem. The experience shows that the idea is effective,

6. Neural Networks

For classification, we will have c output units, one for each of the categories, and the signal from each output unit is the discriminant function $g_k(x)$ as:

$$g_k(x) = z_k = f\left(\sum_{j=1}^{n_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right) \tag{61}$$

This, then, is the class of functions that can be implemented by a three-layer neural network. An even broader generalization would allow transfer functions at the output layer to differ from those in the hidden layer, or indeed even different functions at each individual unit. Kolmogorov proved that any continuous function $g(x)$ defined on the unit hypercube I^n (I=[0,1]and n ≥ 2)can be represented in the form

$$g(x) = \sum_{j=1}^{2n+1} E_j\left(\sum_{i=1}^{d} \psi_{ij}(x_i)\right) \tag{62}$$

or properly chosen functions E_j and $\psi_{ij}(x_i)$.

We consider the training error on a pattern to be the sum over output units of the training squared difference between the desired output t_k (given by a teacher)and the actual error output z_k , much as we had in the LMS algorithm for two-layer nets:

$$J(w) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}(t - z)^2 \tag{63}$$

where t and z are the target and the network output vectors of length c ; w represents all the weights in the network.

The back propagation learning rule is based on gradient descent. The weights are initialized with random values, and are changed in a direction that will reduce the error:

$$\Delta w = -\eta \frac{\partial J}{\partial w} \tag{64}$$

or in component form

$$\Delta w_{\min} = -\eta \frac{\partial J}{\partial w_{\min}} \tag{65}$$

where η is the learning rate, and merely indicates the relative size of the change in weights. This iterative algorithm requires taking a weight vector at iteration m and updating it as:

$$w(m+1) = w(m) + \Lambda w(m) \tag{66}$$

where m indexes the particular pattern presentation

7. Stochastic Search

Search methods based on evolution—genetic algorithms and genetic programming — perform highly parallel stochastic searches in a space set by the designer. The fundamental representation used in genetic algorithms is a string of bits, or chromosome; the representation in genetic programming is a snippet of computer code. Variation is introduced by means of crossover, mutation and insertion. As with all classification methods, the better the features, the better the solution. There are many heuristics that can be employed and parameters that must be set. As the cost of computation contiues to decline, computationally intensive methods, such as Boltzmann networks and evolutionary methods, should become increasingly popular.

7.1 Simulated annealing

In physics, the method for allowing a system such as many magnets or atoms in an alloy to find a low-energy configuration is based on annealing. Annealing proceeds by gradually lowering the temperature of the system—ultimately toward zero and thus no randomness—so as to allow the system to relax into a low-energy configuration. Such annealing is effective because even at moderately high temperatures, the system slightly favors regions in the configuration space that are overall lower in energy, and hence are more likely to contain the global minimum. As the temperature is lowered, the system has increased probability of finding the optimum configuration.

This method is successful in a wide range of energy functions Fortunately, the problems in learning we shall consider rarely involve such pathological functions.

The statistical properties of large number of interacting physical components at a temperature T , such as molecules in a gas or magnetic atoms in a solid, have been thoroughly analyzed. A key result, which relies on a few very natural assumptions, is the following: the probability the system is in a(discrete)configuration indexed by γ

having energy $E\gamma$ is given by

$$P(\gamma) = \frac{e^{-E_\gamma / T}}{Z(T)} \tag{67}$$

where Z is a normalization constant. The numerator is the Boltzmann factor and the denominator the partition function, the sum over all possible configurations

$$Z(T) = \sum_{\gamma'} e^{-E_{\gamma'} / T} \tag{68}$$

which guarantees Eq. 2 represents a true probability. The number of configurations is very high, $2N$, and in physical systems Z can be calculated only in simple cases. Fortunately, we need not calculate the partition function, as we shall see.

7.2 Genetic Algorithms

In basic genetic algorithms, the fundamental representation of each classifier is a binary string, called a chromosome. The mapping from the chromosome to the features chromosome and other aspects of the classifier depends upon the problem domain, and the designer has great latitude in specifying this mapping. In pattern classification, the score is usually chosen to be some monotonic function of the accuracy on a data set, possibly with penalty term to avoid overfitting. We use a desired fitness, θ, as the stopping criterion. Before we discuss these points in more depth, we first consider more specifically the structure of the basic genetic algorithm, and then turn to the key notion of genetic operators, used in the algorithm.

There are three primary genetic operators that govern reproduction: Crossover, Mutation and Selection.: Crossover involves the mixing—"mating"—of two chromosomes. A mating split point is chosen randomly along the length of either chromosome. The first part of chromosome A is spliced to the last part of chromosome B, and vice versa, thereby yielding two new chromosomes. Each bit in a single chromosome is given a small chance, Pmut, of being changed from a 1 to a 0 or vice versa. Other genetic operators may be employed, for instance inversion—where the chromosome is reversed front to back. This operator is used only rarely since inverting a chromosome with a high score nearly always leads to one with very low score. Below we shall briefly consider another operator, insertions.

The process of selection specifies which chromosomes from one generation will be sources for chromosomes in the next generation. Up to here, we have assumed that the chromosomes would be ranked and selected in order of decreasing fitness until the next generation is complete. This has the benefit of generally pushing the population toward higher and higher scores. Nevertheless, the average improvement from one generation to the next depends upon the variance in the scores at a given generation, and because this standard fitness-based selection need not give high variance, other selection methods may prove superior.

The principle alternative selection scheme is fitness-proportional selection, or fitness proportional reproduction, in which the probability that each chromosome is selecte is proportional to its fitness. While high-fitness chromosomes are preferentially selected, occasionally low-fitness chromosomes are selected, and this may preserve diversity and increase variance of the population.

A minor modification of this method is to make the probability of selection proportional to some monotonically increasing function of the fitness. If the function instead has a positive second derivative, the probability that high-fitness chromosomes is enhanced. One version of this heuristic is inspired by the Boltzmann factor of Eq.2; the probability that chromosome i with fitness fi will be selected is

$$P(i) = \frac{e^{f_i/T}}{E\left(e^{f_i/T}\right)} \tag{69}$$

where the expectation is over the current generation and T is a control parameter loosely referred to as a temperature. Early in the evolution the temperature is set high, giving all chromosomes roughly equal probability of being selected. Late in the evolution the temperature is set lower so as to find the chromosomes in the region of the optimal classifier. We can express such search by analogy to biology: early in the search the population remains diverse and explores the fitness landscape in search of promising areas; later the population exploits the specific fitness opportunities in a small region of the space of possible classifiers.

When a pattern recognition problem involves a model that is discrete or of such high complexity that analytic or gradient descent methods are unlikely to work, we may employ stochastic techniques—ones that at some level rely on randomness to find model parameters.Simulated annealing,based on physical annealing of metals, consists in randomly perturbing the system,and gradually decreasing the randomness to a low final level,in order to find an optimal solution.Boltzmann learning trains the weights in a network so that the probability of a desired final output is increased. Such learning is based on gradient descent in the Kullback-Liebler divergence between two distributions of visible states at the output units:one distribution describes these units when clamped at the known category information, and the other when they are free to assume values based on the activations throughout the network. Some graphical models, such as hidden Markov models and Bayes belief networks, have counterparts in structured Boltzmann networks, and this leads to new applications of Boltzmann learning.

8. Unsupervised Learning and Clustering

Until now we have assumed that the training samples used to design a classifier were labelled by their category membership. Procedures that use labelled samples are said to be supervised. Now we shall investigate a number of unsupervised procedures, which use unlabeled samples.

Let us reconsider our original problem of learning something of use from a set of unlabeled samples. Viewed geometrically, these samples may form clouds of points in a d -dimensional space. Suppose that we knew that these points came from a single normal distribution. Then the most we could learn form the data would be contained in the sufficient statistics—the sample mean and the sample covariance matrix. In essence, these statistics constitute a compact description of the data. The sample mean locates the centre of gravity of the cloud; it can be thought of as the single point m that best represents all of the data in the sense of minimizing the sum of squared distances from m to the samples. The sample covariance matrix describes the amount the data scatters along various directions. If the data points are actually normally distributed, then the cloud has a simple hyperellipsoidal shape, and the sample mean tends to fall in the region where the samples are most densely concentrated.

If we assume that the samples come from a mixture of c normal distributions, we can approximate a greater variety of situations. In essence, this corresponds to assuming that the samples fall in hyperellipsoidally shaped clouds of various sizes and orientations. If the number of component densities is sufficiently high, we can approximate virtually any density function as a mixture model in this way, and use the parameters of the mixture to describe the data. Alas, we have seen that the problem of estimating the parameters of a mixture density is not trivial. Furthermore, in situations where we have relatively little prior knowledge about the nature of the data, the assumption of particular parametric forms may lead to poor or meaningless results. Instead of finding structure in the data, we would be imposing structure on it.

One alternative is to use one of the nonparametric methods to estimate the unknown mixture density. If accurate, the resulting estimate is certainly a complete description of what we can learn from the data. Regions of high local density, which might correspond to significant subclasses in the population, can be found from the peaks or modes of the estimated density.

If the goal is to find subclasses,a more direct alternative is to use a clustering procedure. Roughly speaking, clustering procedures yield a data description in terms clustering of clusters or groups of data points that possess strong internal similarities. Formalprocedure clustering procedures use a criterion function, such as the sum of the squared distances from the cluster centres, and seek the grouping that extremizes the criterion function. Because even this can lead to unmanageable computational problems, other procedures have been proposed that are intuitively appealing but that lead to solutions having few if any established properties. Their use is usually justified on the ground that they are easy to apply and often yield interesting results that may guide the application of more rigorous procedures.

8.1 Similarity Measures

The most obvious measure of the similarity(or dissimilarity)between two samples is the distance between them. One way to begin a clustering investigation is to define a suitable distance function and compute the matrix of distances between all pairs of samples. If distance is a good measure of dissimilarity, then one would expect the distance between samples in the same cluster to be significantly less than the distance between samples in different clusters.

Suppose for the moment that we say that two samples belong to the same cluster if the Euclidean distance between them is less than some threshold distance d_0. It is immediately obvious that the choice of d0 is very important. If d_0 is very large, all of the samples will be assigned to one cluster. If d_0 is very small, each sample will form an isolated, singleton cluster. To obtain "natural" clusters, d_0 will have to be greater than the typical within-cluster distances and less than typical between-cluster distances.

Less obvious perhaps is the fact that the results of clustering depend on the choice of Euclidean distance as a measure of dissimilarity. That particular choice is generally justified if the feature space is isotropic and the data is spread roughly evenly a long all directions. Clusters defined by Euclidean distance will be invariant to translations or rotations in feature space—rigid-body motions of the data points. However, they will not be invariant to linear transformations in general, or to other transformations that distort the distance relationships. Thus, a simple scaling of the coordinate axes can result in a different grouping of the data into clusters. Of course, this is of no concern for problems in which arbitrary rescaling is an unnatural or meaningless transformation. However, if clusters are to mean anything, they should be invariant to transformations natural to the problem.

One way to achieve invariance is to normalize the data prior to clustering. For example, to obtain invariance to displacement and scale changes, one might translate and scale the axes so that all of the features have zero mean and unit variance— standardize the data. To obtain invariance to rotation, one might rotate the axes so that they coincide with the eigenvectors of the sample covariance matrix. This trans- formation to principal components can be preceded and/or followed by normalization for scale.

However, we should not conclude that this kind of normalization is necessarily desirable. Consider, for example, the matter of translating and whitening—scaling the axes so that each feature has zero mean and unit variance. The rationale usually given for this normalization is that it prevents certain features from dominating distance calculations merely because they have large numerical values, much as we saw in networks trained with backpropagation. Subtracting the mean and dividing by the standard deviation is an appropriate normalization if this spread of values is due to normal random variation;however,it can be quite inappropriate if the spread is due to the presence of subclasses.

Instead of scaling axes, we can change the metric in interesting ways. For instance, one broad class of distance metrics is of the form

$$d(x,x') = \left(\sum_{k=1}^{d} | x_k - x_k'|^q \right)^{1/q} \tag{70}$$

where q ≥1 is a selectable parameter—the general Minkowski metric we considered in. Setting q =2 gives the familiar Euclidean metric while setting q =1 the Manhattan or city block metric—the sum of the absolute distances along each of the d coordinate axes. Note that only q =2 is invariant to an arbitrary rotation or translation in feature space. Another alternative is to use some kind of metric based on the data itself, such as the Mahalanobis distance.

More generally, one can abandon the use of distance altogether and introduce a nonmetric similarity function $s(x,x')$ to compare two vectors x and x . Convention-similarity ally, this is a symmetric functions whose value is large when x and x are somehowfunction "similar." For example, when the angle between two vectors is a meaningful measure of their similarity, then the normalized inner product

$$s(x,x') = \frac{x^t x'}{\| x \| \cdot \| x' \|} \tag{71}$$

may be an appropriate similarity function. This measure, which is the cosine of the angle between x and x ,is invariant to rotation and dilation, though it is not invariant to translation and general linear transformations.

8.2 Criterion Functions

8.2.1 The Sum-of-Squared-Error Criterion

The simplest and most widely used criterion function for clustering is the sum-of- squared-error criterion. Let ni be the number of samples in D_i and let mi be the mean of those samples,

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x \tag{72}$$

Then the sum-of-squared errors is defined by

$$J_e = \sum_{i=1}^{c} \sum_{x \in D_i} \| x - m_i \|^2 \tag{73}$$

8.2.2 Related Minimum Variance Criteria

By some simple algebraic manipulation we can eliminate the mean vectors from the expression for J_e and obtain the equivalent expression

$$J_e = \frac{1}{2} \sum_{i=1}^{c} n_i \vec{s}_i \tag{74}$$

where

$$\vec{s}_i = \frac{1}{n_i^2} \sum_{x \in D_i} \sum_{x' \in D_i} \| x - x' \|^2 \tag{75}$$

Equation 51 leads us to interprets $\vec{s}_i$ as the average squared distance between points in the i-th cluster, and emphasizes the fact that the sum-of-squared-error criterion uses Euclidean distance as the measure of similarity. It also suggests an obvious way of obtaining other criterion functions. For example, one can replaces $\vec{s}_i$ by the average, the median, or perhaps the maximum distance between points in a cluster. More generally, one can introduce an appropriate similarity function s(x,x)and replaces $\vec{s}_i$ by functions such as

$$\vec{s}_i = \frac{1}{n_i^2} \sum_{x \in D_i} s(x, x') \tag{76}$$

or

$$\vec{s}_i = \min_{x, x' \in D_i} s(x, x') \tag{77}$$

8.3 Hierarchical Clustering

The most natural representation of hierarchical clustering is a corresponding tree, called a dendrogram, which shows how the samples are grouped. If it is possible to measure the similarity between clusters, then the dendrogram is usually drawn to scale to show the similarity between the clusters that are grouped. We shall see shortly how such similarity values can be obtained, but first note that the similarity values can be used to help determine whether groupings are natural or forced. If the similarity values for the levels are roughly evenly distributed throughout the range of possible values, then there is no principled argument that any particular number of clusters is better or “more natural” than another. Conversely, suppose that there is a unusually large gap between the similarity values for the levels corresponding to c =3 and to c =4 clusters. In such a case, one can argue that c =3 is the most natural number of clusters.

8.3.1 The Nearest-Neighbor Algorithm

When $d_{\max}$ is used to measure the distance between clusters the algorithm is sometimes called the nearest-neighbor cluster algorithm, or minimum algorithm Moreover, if it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the single-linkage algorithm. Suppose that we think of the data points as being nodes of a graph, with edges forming a path between the nodes in the same subset D_i . When dmin is used to measure the distance between subsets, the nearest neighbor nodes determine the nearest subsets. The merging of D_i and D_j corresponds to adding an edge between the nearest pair of nodes in D_i and D_j . Since edges linking clusters always go between distinct clusters, the resulting graph never has any closed loops or circuits;in the terminology of graph theory, this procedure generates a tree. If it is allowed to continue until all of the subsets are linked, the result is a spanning tree—a tree with a path from any node to any other node. Moreover,it can be shown that the sum of the edge lengths of the resulting tree will not exceed the sum of the edge lengths for any other spanning tree for that set of samples. Thus, with the use of $d_{\min}$ as the distance measure, the agglomerative clustering procedure becomes an algorithm for generating a minimal spanning tree.

8.3.2 The Farthest-Neighbor Algorithm

When dmax(Eq.75)is used to measure the distance between subsets, the algorithm is sometimes called the farthest-neighbor clustering algorithm, or maximum algorithm. If it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the complete-linkage algorithm. The farthest-neighbor algorithm discourages the growth of elongated clusters. Application of the procedure can be thought of as producing a graph in which edges connect all of the nodes in a cluster. In the terminology of graph theory, every cluster constitutes a complete subgraph. The distance between two clusters is determined by the most distant nodes in the two clusters. When the nearest clusters are merged, the graph is changed by adding edges between every pair of nodes in the two clusters.

Unsupervised learning and clustering seek to extract information from unlabeled samples. If the underlying distribution comes from a mixture of component densities described by a set of unknown parameters θ , then θ can be estimated by Bayesian or maximum- likelihood methods. A more general approach is to define some measure of similarity between two clusters, as well as a global criterion such as a sum-squared- error or trace of a scatter matrix. Since there are only occasionally analytic methods for computing the clustering which optimizes the criterion, a number of greedy(locally step-wise optimal)iterative algorithms can be used, such as k-means and fuzzy k-means clustering.

If we seek to reveal structure in the data at many levels—i.e., clusters with sub-clusters and sub-subcluster—then hierarchical methods are needed. Agglomerative or bottom-up methods start with each sample as a singleton cluster and iteratively merge clusters that are "most similar" according to some chosen similarity or distance measure.Conversely, divisive or top-down methods start with a single cluster representing the full data set and iteratively splitting into smaller clusters, each time seeking the subclusters that are most dissimilar. The resulting hierarchical structure

is revealed in a dendrogram. A large disparity in the similarity measure for successive cluster levels in a dendrogram usually indicates the "natural" number of clusters. Alternatively, the problem of cluster validity—knowing the proper number of clusters —can also be addressed by hypothesis testing. In that case the null hypothesis is that there are some number c of clusters; we then determine if the reduction of the cluster criterion due to an additional cluster is statistically significant.

Competitive learning is an on-line neural network clustering algorithm in which the cluster center most similar to an input pattern is modified to become more like that pattern. In order to guarantee that learning stops for an arbitrary data set, the learning rate must decay. Competitive learning can be modified to allow for the creation of new cluster centers, if no center is sufficiently similar to a particular input pattern, as in leader-follower clustering and Adaptive Resonance. While these methods have many advantages, such as computational ease and tracking gradual variations in the data, they rarely optimize an easily specified global criterion such as sum-of-squared error.

Component analysis seeks to find directions or axes in feature space that provide an improved, lower-dimensional representation for the full data space. In(linear) principal component analysis, such directions are merely the largest eigenvectors of the covariance matrix of the full data; this optimizes a sum-squared-error criterion. Nonlinear principal components, for instance as learned in an internal layer an auto- encoder neural network, yields curved surfaces embedded in the full d-dimensional feature space, onto which an arbitrary pattern x is projected. The goal in independent component analysis—which uses gradient descent in an entropy criterion—is to determine the directions in feature space that are statistically most independent. Such directions may reveal the true sources(assumed independent)and can be used for segmentation and blind source separation.

Two general methods for dimensionality reduction is self-organizing feature maps and multidimensional scaling. Self-organizaing feature maps can be highly nonlinear, and represents points close in the source space by points close in the lower-dimensional target space. In preserving neighborhoods in this way, such maps also called "topologically correct." The source and target spaces can be of very general shapes, and the mapping will depend upon the the distribution of samples within the source space. Multidimensional scaling similarly learns a nonlinear mapping that, too, seeks to preserve neighborhoods, and is often used for data visualization. Because the basic method requires all the inter-point distances for minimizing a global criterion function, its space complexity limits the usefulness of multidimensional scaling to problems of moderate size.

9. Conclusion

One approach to this problem is to use the samples to estimate the unknown probabilities and probability densities, and to use the resulting estimates as if they were the true values. In typical supervised pattern classification problems, the estimation of the prior probabilities presents no serious difficulties. However, estimation of the class-conditional densities is quite another matter. The number of available samples always seems too small, and serious problems arise when the dimensionality of the feature vector x is large. If we know the number of parameters in advance and our general knowledge about the problem permits us to parameterize the conditional densities, then the severity of these problems can be reduced significantly. Suppose, for example, that we can reasonably assume that $p(x \mid \omega_i)$ is a normal density with mean μ_i and covariance matrix Σ_i, although we do not know the exact values of these quantities. This knowledge simplifies the problem from one of estimating an unknown function $p(x \mid \omega_i)$ to one of estimating the parameters μ_i and Σ_i.

In general there are two approaches to develop classifiers: a parametric approach and a nonparametric approach. in a parametric approach, a priori knowledge of data distributions is assumed, otherwise, a nonparametric approach will be employed. Neural networks, fuzzy systems, and support vector machines are typical nonparametric classifiers.

10. References

Abe, S. (2005). *Support vector machines for pattern classification,* Springer, 1852339292, USA

Richard, D.; Peter, H. & David, S. (1999). *Pattern Classification,* Wiley, John & Sons, Incorporated, 0471056693, USA

Simon S. Haykin (2008). *Neural Networks and Learning Machines,* Prentice Hall, 0131471392

Thorsten Joachims(2002), Learning to Classify Text Using Support Vector Machines. Dissertation, Kluwer,.

Vapnik, V. (1998). Statistical learning theory. Chichester, UK: Wiley

Yizhang, G.; Zhifeng, H. (2007). Using SVMs Method to Detect Abrupt Change, Proceedings of 2007 International Conference on Machine Learning and Cybernetics, pp. 3298-3301, 1-4244-0972-1, Aug. 2007, Hong Kong

Classification of support vector machine and regression algorithm

CAI-XIA DENG[1], LI-XIANG XU[2] and SHUAI LI[1]
[1]*Harbin University of Science and Technology,* [2]*Hefei University*
China

1. Introduction

Support vector machine (SVM) originally introduced by Vapnik. V. N. has been successfully applied because of its good generalization. It is a kind of learning mechanism which is based on the statistical learning theory and it is a new technology based on the kernel which is used to solve the problems of learning from the samples. Support vector machine was presented in 1990s, and it has been researched deeply and extensively applied in some practical application since then, for example text cataloguing, handwriting recognition, image classification etc. Support vector machine can provide optimal learning capacity, and has been established as a standard tool in machine learning and data mining. But learning from the samples is an ill–posed problem which can be solved by transforming into a posed problem by regularization. The RK and its corresponding reproducing kernel Hilbert space (RKHS) play the important roles in the theory of function approach and regularization. However, different functions approach problems need different approach functional sets. Different kernel's SVM can solve different actual problems, so it is very significant to construct the RK function which reflects the characteristics of this kind of approach function. In kernel-based method, one map which put the input data into a higher dimensional space. The kernel plays a crucial role during the process of solving the convex optimization problem of SVM. How to choose a kernel function with good reproducing properties is a key issue of data representation, and it is closely related to choose a specific RKHS. It is a valuable issue whether a better performance could be obtained if we adopt the RK theory method. Actually it has caused great interest of our researchers. In order to take the advantage of the RK, we propose a LS-SVM based on RK and develop a framework for regression estimation in this paper. The Simulation results are presented to illustrate the feasibility of the proposed method and this model can give a better experiment results, comparing with Gauss kernel on regression problem.

2. Small Sample Statistical Learning Theory

In order to avoid the assumption that the distribution of sample points and sample purpose of the request created a new principle of statistical inference ---- structured risk minimization principle.

We discussed the two classification problems, that is

$$(x_1, y_1), (x_2, y_2), \cdots, (x_l, y_l) \in R^n \cdot Y$$

where $Y = \{-1, +1\}$; $x_i\left(i = 1, 2, \cdots, l\right)$ is the independent and identical distribution data based on distribution density function $p(x, y)$.

Suppose f to be classifier, which is defined as the expectations of risk

$$R(f) = \int \left| f(x) - y \right| p(x, y) dxdy \, . \tag{1}$$

Experience of risk is defined as

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} \left| f(x_i) - y_i \right| . \tag{2}$$

Since the distribution density function $p(x,y)$ is unknown, it is virtually impossible to calculate the risk expectations $R(f)$.

If $l \to \infty$, we have $R(f) \to R_{emp}(f)$. Accordingly, the process from control theory modeling method to the neural network learning algorithm always constructs model with minimum experience risk. It is called as Empirical Risk Minimization principle.

If $R(f)$ and $R_{emp}(f)$ converge to the same limitation $\inf R(f)$ in probability, that is,

$$R(f) \xrightarrow[n \to \infty]{p} \inf R(f) \, , \qquad R_{emp}(f) \xrightarrow[n \to \infty]{p} \inf R(f) \, .$$

Then Empirical Risk Minimization principle (method) has the consistency.

Unfortunately, as early as in 1971 Vapnik had proved that the minimum of experience of risk may not converge to the minimum of expectations of risk, that is, the experience of risk minimization principle is not established.

Vapnik and Chervonenkis proposed structural risk minimization principle, laid the foundation for small sample statistical theory. They studied the relationship between experience of risk and expectations of risk in-depth, and obtained the following inequality, that is

$$R(f) \le R_{emp}(f) + \sqrt{\frac{h(\ln \frac{2l}{h} + 1) - \ln \frac{\eta}{4}}{l}} \, , \tag{3}$$

which l ----- samples number; η ----- Parameters $\left(0 \le \eta \le 1\right)$; h ----- Dimension of function f, for short VC-dimension.

The importance of formula (3): the right side of inequality has nothing to do with the specific distribution of the sample, that is, Vapnik's statistical learning theory do not need the assumption about the distribution of samples, it overcomes the problem of the high-dimensional distribution to the demand of samples number as exponential growth with the dimension growth. This is essence distinction with the classic statistical theory and the reasons for we call the Vapnik statistical methods for small samples of statistical theory.

From the formula (3), if l/h is large, the expectations of risk (real risk) is decided mainly by the experienced of risk, and this is the reason of the experience of risk minimization principle can often give good results for large sample set. However, if l/h is small, the small value of the experience of risk $R_{emp}(f)$ has not necessarily a small value of the actual risk. In this case, in order to minimize the actual risk, we must consider two items of the right in formula (3): the experience of risk $R_{emp}(f)$ and confidence range (called the VC dimension confidence). VC dimension h play an important role, in fact, confidence range is an increasing function about h . When fixed the number l of points in the sample, the more complex the classifier, that is, the greater the VC dimension h , the greater the range of confidence, leading to the difference between the actual risks and experience gets greater. Therefore, in order to ensure the actual risk to be the smallest, to make sure experience risk minimization, but also to make the VC classifier peacekeeping function as small as possible, this is the principle of structural risk minimization.

With Structural risk minimization principle, the design of a classifier has two-step process:

(1) Choice of model classifier to a smaller VC dimension, that is, small confidence range.

(2) Estimate the model parameters to minimize the risk experience

3. Classification of support vector machine based on quadratic program

3.1 Solving quadratic programming with inequality constraints

On the target of finding a classifying space H which can exactly separate the two-class sample, and maximize the spacing of classification. The classifying space is called optimal classifying hyper plane.

In mathematics, the equation of classifying space is

$$\langle w,x\rangle + b = 0\,,$$

where $\langle w,x\rangle$ is the inner product of the two vector, w is the weight number, b is a constant.

So we can conclude that the problem which maximizes the spacing of classification between the two-class samples corresponds with an optimization problem as followed:

$$\min \Phi(w) = \min_{w,b} \frac{1}{2}\|w\|^2 = \min_{w,b} \frac{1}{2}\langle w,w\rangle\,. \tag{4}$$

The constraint condition is

$$y_i\left[\langle w,x_i\rangle + b\right] \ge 1 \qquad i = 1,2,\cdots,l\,. \tag{5}$$

The (4) and (5) are current methods which describes the date sample is separated by the rule of the support vector machine. Inherently it's a quadratic program problem solved by inequality constraint.

We adopt the Lagrange optimization method to solve the quadratic optimization problem. Therefore, we have to find the saddle point of a Lagrange function

$$L(w,b,\alpha) = \frac{1}{2}\langle w,w\rangle - \sum_{i=1}^{l}\alpha_i\left[y_i\left(\langle w,x_i\rangle + b\right) - 1\right], \tag{6}$$

where $\alpha_i \ge 0$ is the Lagrange multiplier.

By extremal condition, we can obtain

$$Q(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j \langle x_i,x_j\rangle\,. \tag{7}$$

Then we have already changed the symbol from $L(w,b,\alpha)$ to $Q(\alpha)$ for reflecting the final transform.

The expression (7) is called Lagrange dual objective function. Under the constraint condition

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \,, \tag{8}$$

$$\alpha_i \geq 0, \quad i = 1,2,\cdots,l \,, \tag{9}$$

we find that α_i which can maximize the function $Q(\alpha)$. Then, the sample is the support vector when α_i are not zero.

3.2 Kernel method and its algorithm implementation

When the samples are not separated by liner classification, the way of the solution is using a liner transforms $\phi(x)$ to put the samples from input data space to higher dimensional character space, and then we separate the samples by liner classification in higher dimensional character space, and finally we use the $\phi^{-1}(x)$ to put the samples from higher dimensional character space to input data, which is a nonlinear classification in input data.
The basic thought of the kernel method is that, for any kernel function $K(x,x_i)$ which satisfies with the condition of Mercer, there is a character space $(\phi_1(x),\phi_2(x),\cdots,\phi_l(x),\cdots)$ and in this space the kernel function implies inner product. So the inner product has been replaced by kernel in input space.
The advantage of the kernel method is that, the kernel function of input space is equivalent to the inner product in character space, so we only choose the kernel function $K(x,x_i)$ without finding out the nonlinear transforms $\phi(x)$.
Considering the Lagrange function

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\alpha_i\Big(y_i\big(\langle w,\phi(x_i)\rangle + b\big) - 1 + \xi_i\Big) - \sum_{i=1}^{l}\gamma_i\xi_i \,, \tag{10}$$

$$\alpha_i,\gamma_i \geq 0, \quad i = 1,\cdots,l \,.$$

Similar to the previous section, we can get the dual form of the optimization problem

$$\max \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K(x,x_i) \,. \tag{11}$$

The constraint condition is

$$\sum_{i=1}^{l}\alpha_i y_i = 0 \,, \tag{12}$$

$$0 \leq \alpha_i \leq C, \quad i = 1,\cdots,l \,. \tag{13}$$

Totally, the solution of the optimization problem is characterized by the majority of α_i being zero, and the support vector is that the samples correspond with the α_i which are not zero.
We can obtain the calculation formula of b from KKT as followed

$$y_i\left(\sum_{j=1}^{l}\alpha_j y_j K\left(x_j,x_i\right)+b\right)-1=0,\quad \alpha_i\in\left(0,C\right)\,. \tag{14}$$

So we can find the value of b from anyone of the support vector. In order to stabilization, we can also find the value of b from all support vectors, and then get the average of the value.

Finally, we obtain the discriminate function as followed

$$f\left(x\right)=\operatorname{sgn}\left(\sum_{i=1}^{l}\alpha_i y_i K\left(x_i,x\right)+b\right). \tag{15}$$

3.3 One-class classification problem

Let a sample's set be

$$\left\{x_i,i=1,\cdots,l\right\},\quad x_i\in R^d\ .$$

We want to find the smallest sphere with a as its center and R as the radius and can contain all samples. If we directly optimize the samples, the optimization area is a hyper sphere. Allowing some data errors existed, we can equip with slack variable ξ_i to control, and find a kernel function $K\left(x,y\right)$ which satisfies that $K\left(x,y\right)=\left\langle\phi\left(x\right),\phi\left(y\right)\right\rangle$, and the optimization problem is

$$\min F\left(R,a,\xi_i\right)=R^2+C\sum_{i=1}^{l}\xi_i\ . \tag{16}$$

The constraint condition is

$$\left(\phi\left(x_i\right)-a\right)\left(\phi\left(x_i\right)-a\right)^T\le R^2+\xi_i\qquad i=1,\cdots,l\,, \tag{17}$$

$$\xi_i\ge 0,\quad i=1,\cdots,l\ . \tag{18}$$

Type (16) will be changed into its dual form

$$\max\sum_{i=1}^{l}\alpha_i K\left(x_i,x_i\right)-\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j K\left(x_i,x_j\right). \tag{19}$$

The constraint condition is

$$\sum_{i=1}^{l}\alpha_i=1\,, \tag{20}$$

$$0\le\alpha_i\le C,\quad i=1,\cdots,l\,. \tag{21}$$

We can get α by solving (19). Usually, the majority of α will be zero, the samples corresponded with $\alpha_i\neq 0$ are still so-called the support vector.

According to the KKT condition, the samples corresponded with $0<\alpha_i<C$ are satisfied

$$R^2-\left(K\left(x_i,x_i\right)-2\sum_{j=1}^{l}\alpha_j K\left(x_j,x_i\right)+a^2\right)=0\,, \tag{22}$$

$a=\sum_{i=1}^{l}\alpha_i\phi(x_i)$. Thus, according to the (22), we can find the value of R by any support vector. For a new sample z, let

$$f(z)=(\phi(z)-a)(\phi(z)-a)^T=K(z,z)-2\sum_{i=1}^{l}\alpha_iK(z,x_i)+\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_jK(x_i,x_j).$$

If $f(z)\le R^2$, z is a normal point; otherwise, z is an abnormal point.

3.4 Multi-class support vector machine

1. One-to-many method

The idea is to take samples from a certain class as one class and consider the remaining samples as another class, and then there is a two-class classification. Afterward we repeat the above step in the remaining samples. The disadvantage of this method is that the number of training sample is large and the training is difficult.

2. One-to-one method

In multi-class classification, we only consider two-class samples every time, that is, we design a model of SVM for every two-class samples. Therefore, we need to design $\frac{k(k-1)}{2}$ models of SVM. The calculation is very complicated.

3. SVM decision tree method

It usually combines with the binary decision tree to constitute multi-class recognizer, whose disadvantage is that if the classification is wrong at a certain node, the mistake will keep down, and the classification makes nonsense at the node after that one.

4. Determine the multi-class objective function method

Since the number of the variables is very large, the method is only used in small problem.

5. DAGSVM

John C.Platt brings forward this method, combining DAG with SVM to realize the multi-class classification.

6. ECC-SVM methods

Multi-class classification problem can be changed into many two-class classification problems by binary encoding for classification. This method has certain correction capability.

7. The multi-class classification algorithm based on the one-class classification

The method is that we first find a center of hyper sphere in every class sample in higher dimensional character space, and then calculate the distance between every center and test the samples, finally, judge the class based on the minimum distance a point on it.

4. Classification of support vector machine based on linear programming

4.1 Mathematical background

Considering two hyper plane of equal rank on R^d, $H_1:\langle\omega,x\rangle+b_1=0$ and $H_2:\langle\omega,x\rangle+b_2=0$. Based on L_p two the hyper plane distance of norm is:

$$d_p(H_1,H_2) := \min_{\substack{x\in H_1\\ y\in H_2}} \|x-y\|_p , \tag{23}$$

and

$$\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p\right)^{\frac{1}{p}} . \tag{24}$$

Choose a $y \in H_2$ arbitrarily, then two hyper plane's can be write be

$$d_p(H_1,H_2) = \min_{x\in H_1} \|x-y\|_p . \tag{25}$$

Moves two parallel hyper plane to enable H_2 to adopt the zero point, can be obtain the same distance hyper plane:

$$H_1 : \langle \omega, x\rangle + (b_1, b_2) = 0 \ , \ \ H_2 : \langle \omega, x\rangle = 0 .$$

If chooses y spot is the zero point, then the distance between two hyper plane is

$$d_p(H_1,H_2) = \min_{x\in H_1} \|x\|_p . \tag{26}$$

If L_p is the L_q conjugate norm, that is p and q satisfy the equality

$$\frac{1}{p} + \frac{1}{q} = 1 . \tag{27}$$

By the Holder inequality may result in

$$\|x\|_p \|\omega\|_q \geq |\langle x, \omega\rangle| . \tag{28}$$

Regarding $x \in H_1$, we have $\langle \omega, x\rangle = b_1 - b_2$. Therefore

$$\min_{x\in H_1} \|x\|_p \|\omega\|_q = b_1 - b_2 . \tag{29}$$

So, the distance between two hyper plane is

$$d_p(H_1,H_2) = \min_{x\in H_1} \|x\|_p = \frac{|b_1 - b_2|}{\|\omega\|_q} , \tag{30}$$

4.2 Classification algorithm of linear programming

1 norm formula of L_1

The two hyper-plane $H_1 : \langle \omega, x\rangle + b_1 = 0$ and $H_2 : \langle \omega, x\rangle + b_2 = 0$, through the definition of the norm of the distance between them

$$d_1(H_1,H_2) = \frac{|b_1 - b_2|}{\|\omega\|_\infty} , \tag{31}$$

Where, $\|\omega\|_\infty$ expressed as a norm of L_∞, it is the dual norm of L_1, defined as

$$L_\infty = \max_j |\omega_j| . \tag{32}$$

Supposes $H^+ : \langle \omega, x\rangle + b = 1$, $H^- : \langle \omega, x\rangle + b = -1$, established through the two types of support Vector distance between the hyper-plane as follow

$$d_1(H^+, H^-) = \frac{|(b+1)-(b-1)|}{\|\omega\|_\infty} = \frac{2}{\max_j |\omega_j|} . \tag{33}$$

Therefore the optimized question's equation is

$$\min_{\omega,b} \max_j \left|\omega_j\right| . \tag{34}$$

The restraint is

$$y_i\left(\left\langle \omega, x_i \right\rangle + b\right) \geq 1, i = 1, \cdots, l . \tag{35}$$

Therefore obtains the following linear programming

$$\min a . \tag{36}$$

The restraint is

$$y_i\left(\left\langle \omega, x_i \right\rangle + b\right) \geq 1, i = 1, \cdots, l , \tag{37}$$

$$a \geq \omega_j, j = 1, \cdots, d , \tag{38}$$

$$a \geq -\omega_j, j = 1, \cdots, d , \tag{39}$$

$$a, b \in R, \omega \in R^d . \tag{40}$$

This is a linear optimization question, must be much simpler than the quadratic optimization.

2 norm formula of L_∞

If defines L_∞ between two hyper-planes the distances, then we may obtain other one form linear optimization equation. This time, between two hyper-planes distances is

$$d_\infty\left(H_1, H_2\right) = \frac{\left|b_1 - b_2\right|}{\|\omega\|_1} . \tag{41}$$

Regarding the linear separable situation, two support between two hyper-planes the distances is

$$d_\infty\left(H^+, H^-\right) = \frac{\left|(1-b)-(-1-b)\right|}{\|\omega\|_1} = \frac{2}{\sum_j \left|\omega_j\right|} . \tag{42}$$

Maximized type (42) is equivalent to

$$\min_{\omega,b} \sum_j \left|\omega_j\right| . \tag{43}$$

The restraint is

$$y_i\left(\left\langle \omega, x_i \right\rangle + b\right) \geq 1, i = 1, \cdots, l . \tag{44}$$

Therefore the optimized question is

$$\min \sum_{j=1}^{d} a_j . \tag{45}$$

Bound for

$$y_i\left(\left\langle \omega, x_i \right\rangle + b\right) \geq 1, i = 1, \cdots, l , \tag{46}$$

$$a_j \geq \omega_j, j = 1, \cdots, d , \tag{47}$$

$$a_j \geq -\omega_j, j = 1, \cdots, d . \tag{48}$$

4.3 One-class classification algorithm in the case of linear programming

The optimized question is

$$\min \frac{1}{2}\|\omega\|_2^2 - \rho + C\sum_{i=1}^{l}\xi_i \text{ ,} \tag{49}$$

Restrain for

$$\langle \omega, \phi(x_i) \rangle \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \cdots, l \text{ .} \tag{50}$$

Introduces Lagrange the function

$$L = \frac{1}{2}\|\omega\|^2 - \rho + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\alpha_i\left(\langle \omega, \phi(x_i) \rangle - \rho + \xi_i\right) - \sum_{i=1}^{l}\beta_i\xi_i \text{ ,} \tag{51}$$

in the formula $\alpha_i \geq 0, \beta_i \geq 0, i = 1, \ldots, \tilde{l}$.

The function L's extreme value should satisfy the condition

$$\frac{\partial}{\partial \omega}L = 0, \frac{\partial}{\partial \rho}L = 0, \frac{\partial}{\partial \xi_i}L = 0 \text{ .} \tag{52}$$

Thus

$$\omega = \sum_{i=1}^{l}\alpha_i\phi(x_i) \text{ ,} \tag{53}$$

$$\sum_{i=1}^{l}\alpha_i = 1 \text{ ,} \tag{54}$$

$$C - \alpha_i - \beta_i = 0, i = 1, \cdots, l \text{ .} \tag{55}$$

With (53)~(55) replace in Lagrange function (51). And using kernel function to replace inner product arithmetic in higher dimensional space, finally we may result in the optimized question the dual form is

$$\min \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j k(x_i, x_j) \text{ .} \tag{56}$$

Restrain for

$$0 \leq \alpha_i \leq C, \quad i = 1, \cdots, l \text{ ,} \tag{57}$$

$$\sum_{i=1}^{l}\alpha_i = 1 \text{ .} \tag{58}$$

After solving the value of α we may get the decision function

$$f(x) = \sum_{i=1}^{l}\alpha_i k(x_i, x) \text{ .} \tag{59}$$

While taking the Gauss kernel function , we may discover that the optimized equation (56) and a classification class method's of the other form ----- type (19) is equal.

We may obtain its equal linear optimization question by the reference

$$\min\left(-\rho + C\sum_{i=1}^{l}\xi_i\right) \text{ .} \tag{60}$$

Restrain for

$$\langle \omega, \phi(x_i) \rangle \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \cdots, l \text{ ,} \tag{61}$$

$$\|\omega\|_1 = 1 \text{ .} \tag{62}$$

Using kernel expansion $\sum_{j=1}^{l}\alpha_j k(x_j, x_i)$ to replace the optimized question type (60) the inequality constraint item $\langle \omega, \phi(x_i) \rangle$, so we can obtain the following linear programming form:

$$\min\left(-\rho + C\sum_{i=1}^{l}\xi_i\right). \tag{63}$$

Restrain for

$$\sum_{i=1}^{l}\alpha_i k\left(x_i, x\right) \geq \rho - \xi_i, i = 1, \cdots, l, \tag{64}$$

$$\sum_{i=1}^{l}\alpha_i = 1, \tag{65}$$

$$\alpha_i, \xi_i \geq 0, i = 1, \cdots, l. \tag{66}$$

Solving this linear programming may obtain the of value α and ρ , therefore we can obtain a decision function:

$$f(x) = \sum_{i=1}^{l}\alpha_i k\left(x_i, x\right). \tag{67}$$

According to the significance of optimization problems, regarding the majority of training samples will meet $f(x) \geq \rho$, the significance of parameter C satisfies the condition $f(x) < \rho$ to control the sample quantity , the larger parameter C will cause all samples to satisfy the condition ,and the geometry significance of parameter C will give in the 5th chapter. Hyper plane of the decision-making to be as follows:

$$\sum_{i=1}^{l}\alpha_i k\left(x_i, x\right) = \rho. \tag{68}$$

After Hyper plane of the decision-making reflected back to the original space, the training samples will be contained in the regional compact. Regarding arbitrary sample x in the region, satisfies $f(x) \geq \rho$, and for region outside arbitrary sample y to satisfy $f(x) < \rho$ In practical application, the value of parameter σ^2 in kernel function is smaller , which obtains the region to be tighter in the original space to contain the training sample , this explained that the parameter σ^2 will decide classified precisely .

4.4 Multi-class Classification algorithm in the case of linear programming

The following linear programming will be under the classification of a class which is extended to many types of classification. Using the methods implement a classification class operation to each kind of samples, then obtains a decision function to each kind. Then input the wait for testing samples in each decision function, according to the decision function to determine the maximum-point belongs to the category. The concrete algorithm is as follows stated.

Supposes the training sample is:

$$\left\{\left(x_1, y_1\right), \ldots, \left(x_l, y_l\right)\right\} \subset R^n \times Y, Y = \left\{1, 2, \cdots, M\right\},$$

where, n is the dimension of input samples ; M is a category number . Sample is divided into M -type, and various types of classifications are written separately:

$$\left\{\left(x_1^{(s)}, y_1^{(s)}\right), \ldots, \left(x_{l_s}^{(s)}, y_{l_s}^{(s)}\right), s = 1, \cdots, M\right\}$$

where $\left\{\left(x_i^{(s)}, y_i^{(s)}\right), s = 1, 2, \cdots, l_s\right\}$ represents the s -th type of training samples $l_1 + l_2 + \cdots + l_M = l$. A kind of classification thought according to 2.3 section, made the following linear programming:

$$\min\left(-\rho + C\sum_{s=1}^{M}\sum_{i=1}^{l_s}\xi_{si}\right). \tag{69}$$

Restrain for

$$\sum_{j=1}^{l_s}\alpha_j^{(s)}k\left(x_j^{(s)}, x_i^{(s)}\right) \geq \rho - \xi_{si}, s = 1, \cdots, M, i = 1, \cdots, l_s \ , \tag{70}$$

$$\sum_{j=1}^{l_s}\alpha_j^{(s)} = 1, s = 1, \cdots, M \ , \tag{71}$$

$$\alpha_i^{(s)}, \xi_{si} \geq 0, s = 1, \cdots, l_s \ . \tag{72}$$

Solving this linear programming, may obtain M decision functions

$$f_s(x) = \sum_{j=1}^{l_s}a_j^{(s)}k\left(x_j^{(s)}, x\right), s = 1, \cdots, M \ . \tag{73}$$

Assigns treats recognition sample z, calculated $\gamma_i = f_s(z), i = 1, \cdots, M$. Compared with the size, find the largest γ_{k}, then z belongs to the k -th type. At the same time, the definition of the classification results can trust is as follows:

$$\mathrm{B_k} = \begin{cases} 1 & \gamma_k \geq \rho \\ \dfrac{\gamma_k}{\rho} & otherwise \end{cases} . \tag{74}$$

When the difference of the number among samples of various types is large, we can introduce the different ρ value in optimized type (69). And using quadratic programming the similar processing methods, here no longer relates in details.

Another alternative way is to directly compare the new sample size in all decision function, and then the basis maximum value to determine where a new category of sample was taken. As a result of the decomposition algorithm to the optimization process is an independent, it can also be carried out in parallel computing.

5. The beat-wave signal regression model based on least squares reproducing kernel support vector machine

5.1 Support Vector Machine

For the given sample's set

$$\{(x_1 , y_1) , \ldots , (x_l , y_l)\}$$

$x_i \in R^d$, $y_i \in R$, l is the samples number, d is the number of input dimension. In order to precisely approach the function $f(x)$ which is about this data set, For regression analysis, SVM use the regression function as following

$$f(x) = \sum_{i=1}^{l}w_i k(x_i \ , \ x) + q \ , \tag{75}$$

w_i is the weight vector, and q is the threshold, $k(x_i\ , x)$ is the kernel function.
Training a SVM can be regarded as minimizing the value of $J(w, q)$

$$\min J(w,\ q) \ = \ \frac{1}{2}\|\ w\ \|^2 + \gamma \sum_{k=1}^{l} \left(y_k - \sum_{i=1}^{l} w_i k\ (x_i\ ,\ x_k\ ) - q \right)^2 \ . \tag{76}$$

The kernel function $k(x_i\ , x)$ must satisfy with the condition of Mercer. When we define the kernel function $k(x_i\ , x)$, we also define the mapping which is from input datas to character space. The general used kernel function of SVM is Gauss function, defined by

$$k(x,\ x') = \exp(-\|x - x'\|^2 / 2\sigma^2)\ \ . \tag{77}$$

For this equation, σ is a parameter which can be adjusted by users.

5.2 Support Vector's Kernel Function

1. The Conditions of Support Vector's Kernel Function
In fact, if a function satisfies the condition of Mercer, it is the allowable support vector's kernel function.
Lemma 2.1 The symmetry function $k(x, x')$ is the kernel function of SVM if and only if: for all function $g \neq 0$ which satisfies the condition of $\int_{R^d} g^2(\xi)d\xi < \infty$, we need to satisfy the condition as following

$$\iint_{R^d \otimes R^d} k(x,x')g(x)g(x')dxdx' \geq 0\ . \tag{78}$$

This Lemma proposes a simple method to build the kernel function.
For the horizontal floating function, we can give the condition of horizontal floating kernel function.
Lemma 2.2 The horizontal floating function $k(x, x') = k(x - x')$ is a allowable support vector's kernel function if and only if the Fourier transform of $k(x)$ need to satisfy the condition as following

$$\hat{k}(\omega) = (2\pi)^{-\frac{d}{2}} \int_{R^d} \exp(-jwx)k(x)dx \geq 0\ . \tag{79}$$

2. Reproducing Kernel Support Vector Machine on the Sobolev Hilbert space $H^1(R : a,b)$
Let $F(E)$ be the linear space comprising all complex-valued functions on an abstract set E. Let H be a Hilbert (possibly finite-dimensional) space equipped with inner product $(\ \cdot\ ,\ \cdot\ )_H$. Let $h : E \to H$ be a Hilbert space H-function on E. Then, we shall consider the linear mapping L from H into $F(E)$ defined by

$$f(q) = (Lg)(p) = (g,\ h(p))_H\ . \tag{80}$$

The fundamental problems in the linear mapping (80) will be firstly the characterization of the images $f(p)$ and secondly the relationship between g and $f(p)$.
The key which solves these fundamental problems is to form the function $K(p,q)$ on $E \times E$ defined by

$$K(p,q) = (g(q),\ g(p))_H\ . \tag{81}$$

We let $R(L)$ denote the range of L for H and we introduce the inner product in $R(L)$ induced from the norm

$$\|f\|_{R(L)} = \inf\{\| g \|_H \; ; \; f = Lg\} . \tag{82}$$

Then, we obtain

Lemma 2.3 For the function $K(p,q)$ defined by (81), the space $\left[R(L),(\cdot,\cdot)_{R(L)}\right]$ is a Hilbert (possibly finite dimensional) space satisfying the properties that

(i) for any fixed $q \in E$, $K(p,q)$ belongs to $R(L)$ as a function in p ;

(ii) for any $f \in R(L)$ and for any $q \in E$,

$$f(q) = (f(\cdot) \, , K(\cdot,q))_{R(L)} .$$

Further, the function $K(p,q)$ satisfying (i) and (ii) is uniquely determined by $R(L)$.

Furthermore, the mapping L is an isometry from H onto $R(L)$ if and only if $\{h(p); p \in E\}$ is complete in H .

On the Sobolev Hilbert space $H^1(R:a,b)$ on R comprising all complex valued and absolutely continuous functions $f(x)$ with finite norms

$$\left\{\int_{-\infty}^{\infty}(a^2\left|f'(x)\right|^2 + b^2\left|f(x)\right|^2)dx\right\}^{\frac{1}{2}} < \infty , \tag{83}$$

where a , $b > 0$.

The function

$$G_{a,b}(x,y) = \frac{1}{2ab}e^{-\frac{b}{a}|x-y|} = \frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{e^{i\omega(x-y)}}{a^2\omega^2 + b^2}d\omega . \tag{84}$$

is the RK of $H^1(R:a,b)$.

On the Hilbert space, we construct this horizontal floating kernel function:

$$k(x,x') = k(x-x') = \prod_{i=1}^{d} G_{a,b}(x_i - x_i') . \tag{85}$$

Theorem 2.1 The horizontal floating function of Sobolev Hilbert space $H^1(R:a,b)$ is defined as

$$G_{a,b}(x,x') = \frac{1}{2ab}e^{-\frac{b}{a}|x-x'|} , \tag{86}$$

and the Fourier transform of this function is positive.

Proof. By (86), we have

$$\hat{\mathrm{G}}_{a,b}(\omega) = \int_R \exp(-j\omega x)\cdot G_{a,b}(x)dx = \int_R \exp(-j\omega x)\cdot\frac{1}{2ab}e^{-\frac{b}{a}|x|}dx$$
$$= \frac{1}{2ab}\int_R e^{-\frac{b}{a}|x| - j\omega x}dx = \frac{1}{b^2 + a^2\omega^2} \geq 0$$

Theorem 2.2 The function

$$\hat{k}(\omega) = (2\pi)^{-\frac{d}{2}}\int_{R^d}\exp(-j\omega x)k(x)dx = (2\pi)^{-\frac{d}{2}}\cdot\prod_{i=1}^{d}(\frac{1}{2ab}\int_{-\infty}^{+\infty}e^{-\frac{b}{a}|x_i| - j\omega_i x_i}dx_i) , \tag{87}$$

is a allowable support vector kernel function.

Proof. By the Lemma 2.2, we only need to prove

$$\hat{k}(\omega) = (2\pi)^{-\frac{d}{2}}\prod_{i=1}^{d}\frac{1}{b^2 + a^2\omega_i^2} = (2\pi)^{-\frac{d}{2}}\prod_{i=1}^{d}\hat{G}_{a,b}(\omega_i) \geq 0 .$$

That is

$$\begin{aligned}\hat{k}(\omega) &= (2\pi)^{-\frac{d}{2}}\int_{R^d}\exp(-j\omega x)k(x)dx \\ &= (2\pi)^{-\frac{d}{2}}\cdot\prod_{i=1}^{d}\left(\frac{1}{2ab}\int_{-\infty}^{+\infty}e^{-\frac{b}{a}|x_i|-j\omega_i x_i}dx_i\right)\end{aligned}.$$

By Theorem 2.1, we have

$$\hat{k}(\omega) = (2\pi)^{-\frac{d}{2}}\prod_{i=1}^{d}\frac{1}{b^2+a^2\omega_i{}^2} = (2\pi)^{-\frac{d}{2}}\prod_{i=1}^{d}\hat{G}_{a,b}(\omega_i) \geq 0$$

For regression analysis, the output function is defined as

$$f(x) = \sum_{i=1}^{l} w_i \prod_{j=1}^{d}\frac{1}{2ab}e^{-\frac{b}{a}|x_j - x_j^i|} + q\ , \tag{88}$$

x_j^i is the value of the i-th training sample's j-th attribute.

5.3 Least Squares RK Support Vector Machine

Least squares support vector machine is a new kind of SVM. It derives from transforming the condition of inequation into the condition of equation. Firstly, we give the linear regression algorithm as follows.

For the given samples set

$$\{(x_1\ ,\ y_1)\ ,\ \cdots,\ (x_l\ ,\ y_l)\}$$

$x_i \in R^d$, $y_i \in R$, l is the sample's number, d is the number of input dimension. The linear regression function is defined as

$$f(x) = w^T x + q\ . \tag{89}$$

Importing the structure risk function, we can transform regression problem into protruding quadratic programming

$$\min\left\{\frac{1}{2}\|w\|^2 + \gamma\frac{1}{2}\sum_{i=1}^{l}\xi_i^2\right\}. \tag{90}$$

The limited condition is

$$y_i = w^T x_i + q + \xi_i\ . \tag{91}$$

We define the Lagrange function as

$$L = \frac{1}{2}\|w\|^2 + \gamma\frac{1}{2}\sum_{i=1}^{l}\xi_i^2 - \sum_{i=1}^{l}\alpha_i(w^T x_i + q + \xi_i - y_i)\ , \tag{92}$$

and we obtain

$$\begin{cases}\dfrac{\partial L}{\partial w} = 0 \rightarrow w = \displaystyle\sum_{i=1}^{l}\alpha_i x_i \\ \dfrac{\partial L}{\partial q} = 0 \rightarrow \displaystyle\sum_{i=1}^{l}\alpha_i = 0 \\ \dfrac{\partial L}{\partial \xi_i} = 0 \rightarrow \alpha_i = \gamma\xi_i\ ;\quad i = 1,\cdots,l \\ \dfrac{\partial L}{\partial \alpha_i} = 0 \rightarrow w^T x_i + q + \xi_i - y_i = 0\ ;\quad i = 1,\cdots,l\end{cases}. \tag{93}$$

From equations (93) , we can get the following linear equation

$$\begin{bmatrix} I & 0 & 0 & -x \\ 0 & 0 & 0 & \Phi^T \\ 0 & 0 & \gamma I & -I \\ x^T & \Phi & I & 0 \end{bmatrix} \begin{bmatrix} w \\ q \\ \xi \\ \alpha \end{bmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ y \end{vmatrix}, \tag{94}$$

where $x = [x_1, \cdots, x_l]$, $y = [y_1, \cdots, y_l]$, $\Phi = [1, \cdots, 1]$, $\xi = [\xi_1, \cdots, \xi_l]$, $\alpha = [\alpha_1, \cdots, \alpha_l]$.
The equation result is

$$\begin{bmatrix} 0 & \Phi^T \\ \Phi & x^T x + \gamma^{-1} I \end{bmatrix} \begin{bmatrix} q \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}, \tag{95}$$

where $w = \sum_{i=1}^{l} \alpha_i x_i$, $\xi_i = \alpha_i / \gamma$.

For non-linear problem, the non-linear regression function is defined as

$$f(x) = \sum_{i=1}^{l} \alpha_i k(x_i, x) + q \, . \tag{96}$$

The above equation result can be changed into

$$\begin{bmatrix} 0 & \Phi^T \\ \Phi & K + \gamma^{-1} I \end{bmatrix} \begin{bmatrix} q \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}, \tag{97}$$

$K = \left\{ k_{i,j} = k(x_i \, , \, x_j) \right\}_{i\,,\,j=1}^{l}$, the function $k(\cdot \, , \, \cdot)$ is given by (87). Based on the RK kernel function, we get a new learning method which is called least squares RK support vector machine (LS-RKSVM). Since using least squares method, the computation speed of this algorithm is more rapid than the other SVM.

5.4 Simulation results and analysis

We use LS-RKSVM to regress the Beat-wave signal

$$a(t) = A \sin(\omega_1 t) \cdot \sin(\omega_2 t)$$

where $a(t)$ is the Beat-wave signal, A is the Signal amplitude, ω_1 is the higher frequency of Beat-wave frequencies, that is the resonant frequency of resonant Beat-wave, ω_2 is the frequency of Beat-wave, the relationship between ω_1 and ω_2 is that $\omega_2 = \omega_1 / 2n$, where n is the cycle number of sine wave which is included in a beat-wave with a ω_1 frequency.

We assume $A = 1.1,\ \omega_2 = 0.5,\ n = 5,\ t = 2s$, and 150 sampling points. We can get the result of this experiment which can be described as figure 1, figure 2 and table 1. Figure 1 is the regression result of LS-SVM which uses the Gauss kernel function. Figure 2 is the regression result of LS-RKSVM which uses the RK kernel function.

For regression experiments, we use the approaching error as following

$$E_{ms} = \left(\frac{1}{N} \sum_{t=1}^{N} (y(t) - y^l(t))^2 \right)^{\frac{1}{2}} . \tag{98}$$

The Simulation results shows that the regression ability of RK kernel function is much better than Gauss kernel function. This reveals RK kernel function has rather strong regression ability and it can be used for pattern recognition. We can find that the LS-SVM is a very promising method based on RK kernel. The model has strong regression ability.

kernel function	kernel parameter	error
RBF kernel:γ=150	σ =0.01	0.0164
RK kernel: γ=150	a=0.1, b=1	0.0110

Table 1. The regression result for Beat-wave signal

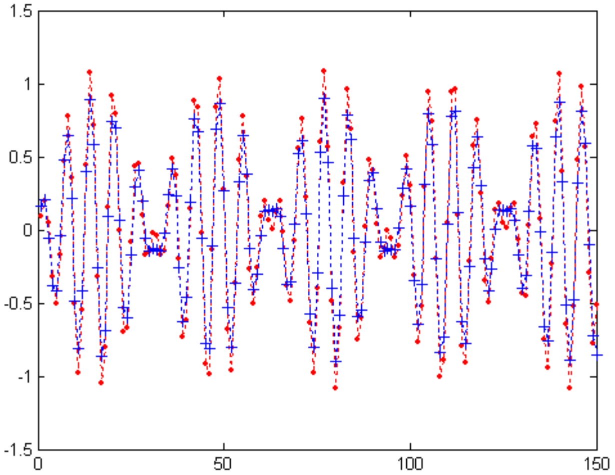


Fig. 1. The regression curve based on Gauss kernel (“.” is true value, “+” is predictive value)

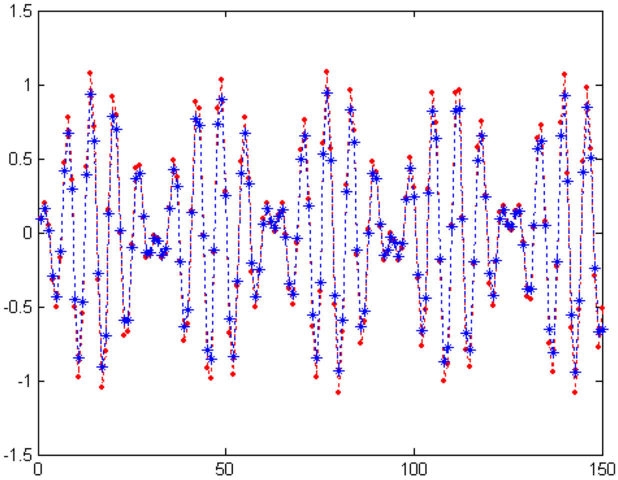


Fig. 2. The regression curve based on RK kernel (“.” is true value, “*” is predictive value)

The SVM is a new machine study method which is proposed by Vapnik based on statistical learning theory. The SVM focus on studying statistical learning rules under small sample.

Through structural risk minimization principle to enhance extensive ability, the SVM preferably solves many practical problems, such as small sample, non-linear, high dimension number and local minimum points. The LS-SVM is an improved algorithm which base on SVM. This paper proposes a new kernel function of SVM which is the RK kernel function. We can use this kind of kernel function to map the low dimension input space to the high dimension space. The RK kernel function enhances the generalization ability of the SVM. At the same time, adopting LS-SVM, we get a new regression analysis method which is called least squares RK support vector machine. Experiment shows that the RK kernel function is better than Gauss kernel function in regression analysis. The RK and LS-SVM are combined effectively. Thereby we can find that the result of regression is more precisely.

6. Prospect

Further study should be started in the following areas:
1. The kernel method provides an effective method which can change the nonlinear problem into a linear problem, that is, the kernel function plays an important role in the support vector machine. Therefore, for practical problems, rational choice of the kernel function and the parameter in it is a problem which should be research.
2. For the massive data of practical problems, a serious problem need to be solved is to propose an efficient algorithm.
3. It is a valuable research direction that fusion of the Boosting and the Ensemble methods are proposed to be a better algorithm of support vector machine.
4. It is significant to put the support vector machine, planning network, Gauss process and neural network into same frame.
5. It is a significant research subject that combines the idea of support vector machine with the Bayes Decision and consummates the maximum margin algorithm.
6. The research on support vector machine still needs to be done extensively.

7. References

Bernhard S. & Sung K.K. (1997). Comparing support vecter machines with Gaussian kernels to radical basis fuction classifiers. IEEE transaction on signal processing

Fatiha M. & Tahar M. (2007). Prediction of continuous time autoregressive processes via the reproducing kernel spaces. Computational mechanics, Vol 41, No. 1, dec

Karen A. Ames; Rhonda J. & Hughes. (2005). Structural stability for ill-posed problems in Banach space, Semigroup forum, Vol 70, No. 1, Jan

Mercer J. (1909). Function of positive and negative type and their connection with the theory of integral equations. Philosophical transactions of the royal society of London, Vol 209, pp. 415~446

O.L.Mangasarian. (1999). Arbitrary-norm Separating Plane. Operation Research Letters, 1(24): 15~23

Saitoh S. (1993). Inequalities in the most simple Sobolev space and convolutions of L_2 functions with weights. Proc. Amer. Math. Soc. Vol 118, pp. 515~520

Smola A. J.; Scholkopf B. & Muller K. R. (1998). The connection between regularization operators and support vector kernels. Neural networks, Vol 11, No. 4, pp. 637~649

Suykens J. & A. K. (2002). Least squares support vector machines, World scientific, Singapore

Vapnik. V. N. (1995). The nature of statistical learning theory. Springer-verlag, New York

Vapnik V N. (1998). Statistical Learning Theory. Wiley, New York

Yu Golubev. (2004). The principle of penalized empirical risk in severely ill-posed problems. Probability theory and related fields, Vol 130, No. 1, sep

6

Classifiers Association for High Dimensional Problem: Application to Pedestrian Recognition

Laetitia LEYRIT, Thierry CHATEAU and Jean-Thierry LAPRESTE
LASMEA - UMR 6602 CNRS & Blaise Pascal University
France

1. Introduction

Machine learning is widely used for object recognition in images (Viola & Jones, 2001). Indeed, the goal is to recognize any object of the same class whatever the background, the illumination conditions, ... The key-point of such a method is the ability to create a generic model able to describe the huge variability of an object class. A large training set is then used so as to cover all the variations taken by the object class. For each example, a simple description provides a huge feature vector from which only a subset is relevant according to the object to be recognised.
Kernel based machines like Support Vector Machine (SVM) (Vapnik, 1998) have shown great performances for object recognition in images (Papageorgiou & Poggio, 2000). But high dimensional problem can be prohibitive for it: it implies expensive time, presence of irrelevant features can disturb the classifier and overfitting often occurs.
Various approaches were proposed in order to decrease this number of variables (Guyon & Elisseeff, 2003). They are of two types: the *filters* and the *wrappers*. The filters methods use only the training set. They process the entire data before the learning step and keep only relevant characteristics. Most widespread is the Relief algorithm, introduced by Kira and Rendell (Kira & Rendell, 1992) and improved by Konenko (Kononenko, 1994), which computes a criterion of relevance for each characteristic of the training set. Another approach presented by Hall in (Hall, 2000) uses a correlation score to reduce the training set. An extension of this method was developed in (Yu & Liu, 2003) for great dimension sets.
The wrappers methods carried out the variable choice at the same time as the training process is done. Moreover, they use the process itself to select relevant characteristics (Kohavi & John, 1997). Solutions were brought for SVMs. Weston in (Weston et al., 2000) explores parameter space by a stepped gradient descent and fix an exit threshold on the classification error. Rakotomamonjy proposes in (Rakotomamonjy, 2003) a selection criterion based on the variable influence on the decision rule of a SVM classifier. Generally, these methods, which take account of the training set and the classifier in the same time, give good results but induce expensive computing times.
For an out-line learning, the computing time is not the main problem. However, studies like (Campedel et al., 2005) showed the efficiency of variable selection to improve the classifier performances: the presence of useless data can disturb the classifier and memory is misused.

With the aim of time-saving, the ideal is to use a variable selection algorithm which can be processed independently of the learning process. But not to take account of the classifier is the major disadvantage of the filtering methods: the drawback is to select attributes which are not finally useful for this one. To guarantee the relevance of the characteristics preserved for the classifier, the best tool is this classifier used itself.

AdaBoost algorithms can also be used for feature selection. In (Viola & Jones, 2001) an AdaBoost cascade is used with Haar wavelets based descriptor. At each stage of the classifier, a Haar resolution is chosen and images are divided into several sub-windows. The classifier rejects the non-informative ones. At the follow step, the Haar resolution is increased only for sub-windows which were selected at the previous iteration. This stage of the classifier rejects also the non-informative sub-windows and the process keeps on. After several stages of the classifier, the number of sub-windows decrease quickly. Moreover the decision threshold is readjusted as the classifier progresses. An extension of this cascade method is developed in (Le & Satoh, 2004) with a final SVM classifier. First stages of the classifier use AdaBoost algorithm to reduce the feature space and select relevant features. The last stage is a SVM classifier which builds a face model from the features selected previously. This both methods allow to reduce the features number in the first stages of the cascade. In this approach, the AdaBoost algorithm is only used to select relevant features from a huge set of possible ones.

We propose here an original association of a classic AdaBoost algorithm with a kernel based machine. Adaboost is an algorithm which builds a strong classifier by selecting a huge number of weak ones. It can be used for feature selection too: each feature can be seen as a weak classifier and AdaBoost selects a subset of them. Our approach consists on using the resulting subset of weak relevant classifiers (and not relevant features) as binary vectors in a kernel based machine learning classifier (like SVM).

We focus our proposal on pedestrian recognition: since pedestrians provide a large appearance variability (size, clothes, skin colour, ...) the training set used for learning must be very large. Numerous features are then used to describe correctly each sample of the training set. The association of AdaBoost and kernel machine allows to handle this high dimensional problem.

This chapter is organized as follow: section 1 describes the main features used in classification; then the classifiers methods is presented in section 2. Experiments and results of the proposed method on a pedestrian recognition task are realizes in section 3; and finally section 4 gives the conclusion.

2. Features for image description

In pattern recognition, different types of features are widely used to describe each image. For many recognition system, the goal is to realize an on-line application, and features are chosen according to that. These features have to be efficient of course but moreover to compute quickly so as to reach an on-line processing time. We present here three types of features widely used in pattren recognition: Haar wavelets, histograms of oriented gradients and binary descriptors.

2.1 Haar wavelets

They are probably the most used features for pattern recognition in images. Introduced by Papageorgiou in (Papageorgiou & Poggio, 2000), they encode a local information with an intensity difference at different scales.

The overcomplete dictionnary presented in (Papageorgiou & Poggio, 2000) allows a fast computation of haar wavelets in threes directions: horizontal, vertical and diagonal (see figure 1) The main difficulty is to find adapted sizes for a given image. Indeed a finer scale only captures noise whereas large scale doesn't capture an object characteristics.

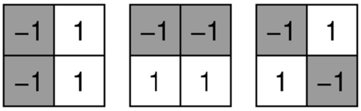


Fig. 1. The three different orientations of the Haar wavelets.

2.2 Histograms of Oriented Gradients

This image descriptor for image recognition was introced in (Dalal & Triggs, 2005). This descriptor is a based on edge orientation histograms (Freeman & Roth, 1995) and SIFT descriptor (Lowe, 2004) The idea is to count occurrences of gradient orientation in localized portions of an image called cells. We thus obtain a dense grid of uniformly spaced cells and overlapping local contrast normalization is used for improved performance. Here difficulties are to choose a correct grid to describe an object and to select a correct normalization schema.

2.3 Binary descriptors

This kind of descriptor becomes very popular used in pattern recognition, especially for low resolution images. It computes an intensity comparison of two points, which is an enough resolution for small images. Various methods were proposed to select and associated them (Lepetit & Fua, 2006; Moutarde et al., 2008). The points selection appears as the main problem of this descriptor because it gives quickly a high dimensional feature vector if all the possibilities space is cover up. We build our own binary descriptor based on the comparison result of the grey levels as presented in the figure 2. Let us note $\mathbf{u}$ the coordinates of an image point. $I(\mathbf{u})$ returns the pixel intensity at this point, i.e. the grey level associated with these coordinates in the image. Given two points of the image, $\mathbf{u}_1$ and $\mathbf{u}_2$, the descriptor carries out the following comparison:

$$(I(\mathbf{u}_1) \geq I(\mathbf{u}_2)) \; (with\; \mathbf{u}_1 \neq \mathbf{u}_2) \tag{1}$$

It returns the logical value 1 if the test is true and 0 if the test is false.

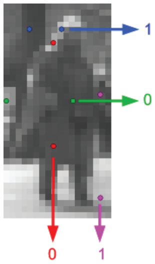


Fig. 2. Descriptor of grey-levels comparisons.

3. Machine Learning

Today the machine learning used in image recognition are either Boosting method either Kernel method. This section describes these learning methods.

Here, we follow the standard notations, representing the output labels by a scalar y which can take two possible discrete values corresponding to the object class: $y = -1$ for negative examples (non-objects) and $y = 1$ for positive examples (objects). Vectors $\mathbf{x} \in I\!R^{Q}$ represents input features provided by images descriptors. Let $S \doteq \{(\mathbf{x}^i, y^i)\}_{i=1}^{N}$ denotes a training set composed by N samples of feature vectors associated to their corresponding labels.

3.1 Adaptative Boosting or *AdaBoost*

This method is developped by Freund and Schapire in (Freund & Schapire, 1996). It's the more commun method of Boosting used for image recognition. The AdaBoost principle is straight forward. As opinions of several experts are better than only one, this algorithm combines decisions of several weak classifiers. An uniform weight is given for each data of the training set. At each iteration, a subset of the training set is drawn from S according to the weights assigned to them at the previous iteration. A weak classifier $h(t)$ is created from this subsample and the classification error ϵ_t is calculated for the entire training set. The weight vector is then updated: the weight of the elements well classified decreases, while the weight of those badly classified increases. The process is reiterated until reaching the number of required weak classifiers or until the error on the training set is lower than a given threshold.

The decision rule associated to Adaboost is then a linear combination of selected weak classifiers:

$$y = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})) \tag{2}$$

where α_t are weights estimated from the learning step. Equation (2) clearly shows that the classification rule is given by the sign of a linear equation (hyperplane) into the selected weak classifiers space.

3.2 Kernel Machine

Kernel based machines has been widely used for classification ((Suard et al., 2006),(Papageorgiou & Poggio, 2000),(Shashua et al., 2004)). The general form of the classifier is given by:

$$y = \text{sign}\left(\sum_{m=1}^{M} w_m \phi_m(\mathbf{x})\right) \tag{3}$$

Here, $\{\phi_m(\mathbf{x}) | m = 1...M\}$ are basis functions and $\{w_m | m = 1...M\}$ are the associated weights. We propose non linear basis functions:

$$\phi_m(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}^m) \tag{4}$$

where $k(\mathbf{x}, \mathbf{x}^m)$ is a kernel function. The classification rule can be written in a more compact form by the following equation:

$$y = \text{sign}(\mathbf{w}^T \phi(\mathbf{x})) \tag{5}$$

where $\mathbf{w}^T = (w_1, w_2, ..., w_M)$ is a weight vector and $\phi(\mathbf{x}) = (\phi(\mathbf{x}^1), \phi(\mathbf{x}^2), ..., \phi(\mathbf{x}^N))^T$. To train the model (estimate $\mathbf{w}$), we are given the training set $S^h = \{(\mathbf{x}^i, y^i)\}_{i=1}^{N}$. We use the Euclidean norm to measure y-space prediction errors, so the estimation problem is of the form:

$$\mathbf{w} := \arg\min_{\mathbf{w}} \{||\mathbf{w}^T \Phi - \mathbf{y}||^2\} \tag{6}$$

where $\Phi \doteq \left(\phi(\mathbf{x}^1)), \phi(\mathbf{x}^2), ..., \phi(\mathbf{x}^N)\right)$ is the design matrix and $\mathbf{y} \doteq (y^1, ...y^N)^T$ is the training set label vector. The estimation of the parameter vector $\mathbf{w}$ using the least-square criterion defined in equation (6) is given by:[1]

$$\mathbf{w}_{ls} = \mathbf{y}\Phi^{+} \tag{7}$$

Alternative methods can be used to estimate $\mathbf{w}$. A solution is to place priors over $\mathbf{w}$ in order to set many weights to zero. The resulting model is then called sparse linear model. SVM (*Support Vector Machine*)(Vapnik, 1998) is a sparse linear model where the weights are estimated by the minimization of a Lagrange multipliers based functional. Other sparse linear models, like RVM (*Relevant Vector Machines*) (Tipping, 2001) may also be employed.

Vectors used for basis functions are usually composed by a subset of the training set S^h. It is also possible to use the entire training set and in this case $M = N$. The matrix Φ is then symmetric and system resolution can be made more efficiently using Cholesky decomposition.

We make the common choice to use Gaussian data-centred basis functions:

$$\phi_m(\mathbf{x}) = \exp\left[-\left(\mathbf{x} - \mathbf{x}^m(\mathbf{x})\right)^2 / \sigma^2\right], \tag{8}$$

which gives us a "radial basis function" (RBF) type model from which the parameter σ must be adjusted. On one hand, if σ is too small, the "design matrix" Φ is mostly composed of zeros. On the other hand, if σ is too large, Φ is mostly composed of ones. We propose to adjust σ using a non linear optimization maximizing an empirical criteria based on the sum of the variances computed for each line of the design matrix Φ:

$$\sigma := \arg\max_{\sigma}[-C(\sigma)] \tag{9}$$

with

$$C(\sigma) = \sum_{n=1}^{N}\sum_{m=1}^{M}\left(\phi_m(\mathbf{x}) - \overline{\phi}(\mathbf{x}^{(n)}(\mathbf{x}))\right)^2 \tag{10}$$

and

$$\overline{\phi}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M}\phi_m(\mathbf{x}^{(n)}(\mathbf{x})) \tag{11}$$

The classifier thus obtained will be denoted by KHA (*Kernel Approximation Hyperplane*) in section 4.

3.3 Classifiers Association

Our goal is to be able to recognize object with a great variability. For that, we use dataset with a large number of examples, and each example is described by a huge features vector. Using all this features set is prohibitive for the computation of kernel machines. That's why we propose here to use an Adaboost algorithm to choose only the more relevant features into the training set. As AdaBoost selects the best weak classifiers for a classification task, the features selected will be relevant for a kernel machine too. Here, we propose an original approach which consists in selecting weak classifiers with AdaBoost (not relevant feature selection), and then using the selected weak classifiers as new binary input vectors to learn a kernel based classifier. This method provides non-linear separator in the weak learner space and classifies accurately more examples as shown in figure 3; positive examples are denoted by symbol $+$, negative examples by $-$. $h_1(x)$ and $h_2(x)$ are two weak classifiers selected by AdaBoost. They

[1] Φ^{+} denotes the pseudo-inverse of Φ.

return value 1 for examples classified as positive and -1 for examples classified as negative. In the weak learner space, AdaBoost provides a linear separator and some examples are misclassified. Our method provides non-linear separator and classified correctly all examples. So

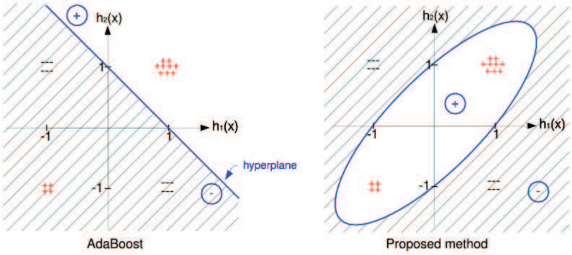


Fig. 3. Our method provides non-linear separator while AdaBoost gives linear one.

the first stage of the learning process is doing by an AdaBoost algorithm which gives a new binary training set for the kernel machines. We define $S^h \doteq \{(\mathbf{h}(\mathbf{x}^i), y^i)\}_{i=1}^N$ a new training set where $\mathbf{h}(\mathbf{x^i}) \in I\!R^T$ is a vector composed by the output of each selected classifier estimated from parameters $\mathbf{x}$ such as $\mathbf{h}(\mathbf{x^i}) \doteq \left(\mathbf{h_1}(\mathbf{x^i}), ..., \mathbf{h_T}(\mathbf{x^i})\right)^T$. Next the classification rule for kernel machines becomes:

$$y = \text{sign}\left(\sum_{m=1}^{M} w_m \phi_m(\mathbf{h}(\mathbf{x}))\right) \tag{12}$$

and

$$\phi_m(\mathbf{h}(\mathbf{x})) = k(\mathbf{h}(\mathbf{x}), \mathbf{h^m}(\mathbf{x})) \tag{13}$$

where $k(\mathbf{h}(\mathbf{x}), \mathbf{h}^m(\mathbf{x}))$ is the kernel function. The estimation problem is given by equation (6) and the resolution is done like presented in section 3.2 .

4. Experiments and Results

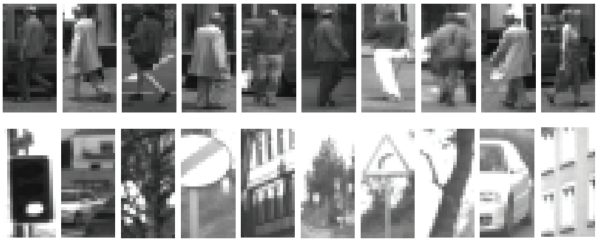


Fig. 4. Examples of pedestrian (first line) and non-pedestrian (second line) images.

We focus our work on pedestrian recognition in images coming from low-cost camera (see (Leyrit et al., 2008)). This work is challenging because pedestrian is a hard pattern to recognize due to the differences of clothes, size... added to classic illumination and background variations. Since pedestrian appearance provides a large variability, the training set used in the learning stage must be very huge and each sample can be described by a huge feature vector. We used the images dataset provided by Gavrila and Munder in (Munder & Gavrila, 2006). This base is subdivided into five parts; each one contains 4800 positive and 5000 negative images. Each picture has a size of 36x18 pixels, in grey levels. In the positive images, the pedestrians are standing and entirely visible; they were taken in various postures, and in various illumination and background conditions. Each pedestrian picture was randomly reflected and shift a few pixels in the horizontal and vertical directions. The negative images describe the urban environment: trees, buildings, cars, roadsigns... This base (some examples of which are shown in figure 4) constitutes the data used for training and the validation of the proposed method. According to (Munder & Gavrila, 2006) the three first parts are used for the training, and the two last ones are used for the validation. It assumes that the validation is doing independently of the training.

4.1 The proposed method compared to a standard AdaBoost

In this part we compare the results of a standard AdaBoost to the ones given by the proposed method. We use here the association of the binary output of the AdaBoost with a KHA kernel machine. Results are obtained for the descriptor of grey level comparisons (see paragraph 2.1). We make these comparisons between points belonging to the same line or the same column. As the image size is of 36x18 pixels, we obtain 5508 binary descriptors for the lines and 11340 for the columns.

The table 1 gives recognition errors for these experiments; the class decision rule is computed with a classical threshold set to zero. We also plot ROC curves for each experiments as shown in figure 5 for more precision. A ROC curve (*Receiver Operating Characteristic*) presents variations and sensitivity of a test for various values of the discrimination threshold. The x-axis represents the false negative rate (non-pedestrians classified as pedestrians) while the y-axis corresponds to the true positive rate (of the pedestrians which are well detected as such). Let us suppose a ROC curve through the point (0.1;0.9). That means that for 90% of the well classified pedestrians, 10% of non-pedestrians are badly classified. Most of the time, the classifiers association gives best results than a standard AdaBoost. In this way, we can achieve good recognition rate despite the high dimensional size of the features vectors.

Classifier	Learning set number	Error on validation set 4	Error on validation set 5
Standard AdaBoost	1	27,49%	25.09%
Association with KHA	1	30,61%	24,21%
Standard AdaBoost	2	26,81%	31,69%
Association with KHA	2	16,62%	14,81%
Standard AdaBoost	3	44,79%	44,71%
Association with KHA	3	21,42%	13,48%

Table 1. Comparison between standard AdaBoost and the classifiers association using grey-level features.

4.2 Comparison between two different kernel machines

We tested the proposed method with a SVM kernel machine. Results are given in table 2 (decision threshold set to zero) and ROC curves are presented in figure 6. With a decision threshold set to zero, KHA gives better results; but we can see in figure 6 that for other decision

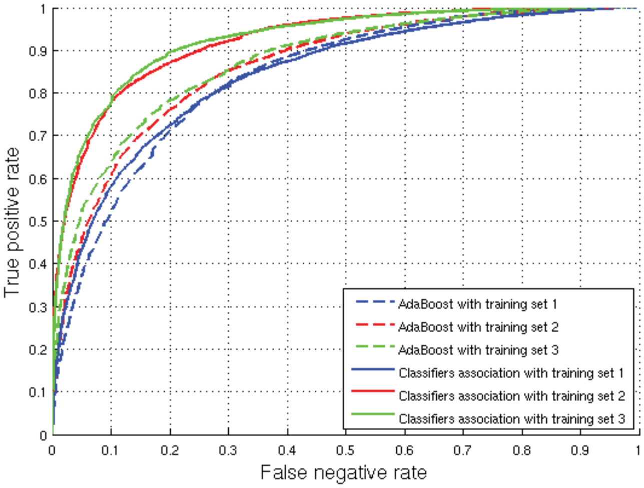


Fig. 5. Comparison between standard Adaboost and the proposed method.

thresholds, SVM reachs equivalent recognition rate. The decision threshold should be selected carrefully according to each application. It depends on the rate of misclassified an application can tolerate.

Classifier	Learning set number	Error on validation set 4	Error on validation set 5
Association with KHA	1	30,61%	24,21%
Association with SVM	1	30,62%	25,70%
Association with KHA	2	16,62%	14,81%
Association with SVM	2	20,76%	21,09%
Association with KHA	3	21,42%	13,48%
Association with SVM	3	30,96%	27,86%

Table 2. Errors of the proposed method with two different kernel machines.

4.3 Comparison between three descriptors

We have implemented the three descriptors presented in section 2: the histograms of oriented gradients (HOG), Haar wavelets and our grey-level descriptor.

We chose a Haar wavelets size of 2x2 and 4x4, shifted by $\frac{1}{4}$ of the size of the wavelet in the three directions (see paragraph 2.1). It gives 1755 features.

As regards the HOG, we chose a cutting into 3x3 cells, and histograms are computed with 8 bins. We thus obtain 576 features.

For the two previous descriptors, the feature vector is relatively small and doesn't require a previous weak classifier selection. We only use the complete method for the binary descriptor. Then a KHA kernel machine is trained on each dataset.

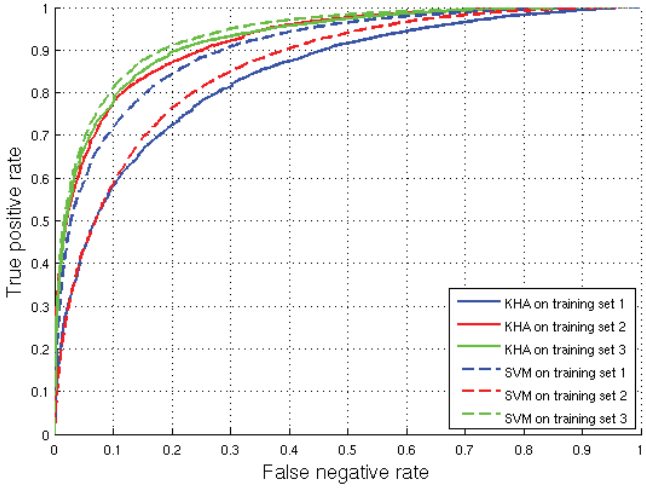


Fig. 6. The proposed method implemented with two different kernel machines.

The results presented in table 3 and in figure 7 show that these three descriptors work into almost the same range of values. With more precison, ROC curves show that histogramms of oriented gradients are better for these pedestrian recognition task. The binary descriptor, despite its simplicity, achieves almost same results. Haar wavelets doesn't reach the same performances than the two others descriptors.

Descriptor	Learning set number	Error on validation set 4	Error on validation set 5
HOG	1	24,99%	21.17%
Haar wavelets	1	31,54 %	31,18%
Grey-level descriptor	1	30,61%	24,21%
HOG	2	26,17%	21,39%
Haar wavelets	2	20,35%	20,55%
Grey-level descriptor	2	16,62%	14,81%
HOG	3	18,84%	16,32%
Haar wavelets	3	21,84%	16,32%
Grey-level descriptor	3	21,42%	13,48%

Table 3. Errors with three different descriptors.

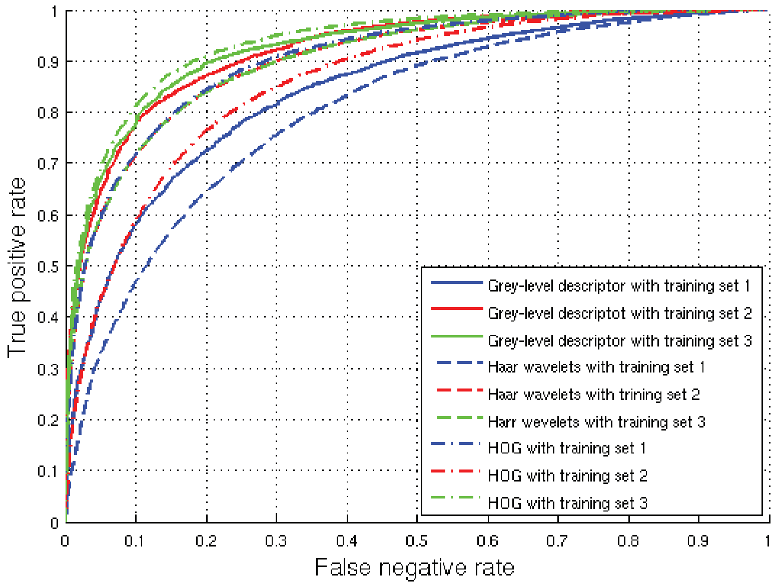


Fig. 7. Comparison between three different descriptors.

5. Conclusion and future works

We have proposed a learning based approach for high dimensional object detection. This method uses an AdaBoost algorithm to select relevant weak classifiers, which are then used as binary vectors to learn a kernel based classifier.

This method helps to solve high dimensional problem like a pedestrian recognition task; results show that the method gives good performances and outperforms standard AdaBoost.

Three popular descriptors have been tested; histograms of oriented gradients give the best results but the binary descriptors reach almost the same results, which is very interesting for a real-time application. Haar wavelets are less competitive than the two others descriptors.

We have designed our own classifier, KHA, which gives similar results than a SVM classifier. KHA is easy to develop but we explore now solutions to decrease the number of retained support vectors and to select relevant ones. It should help us to reach faster computation time.

6. References

Campedel, M., Moulines, E., Matre, H. & Dactu, M. (2005). Feature Selection for Satellite Image Indexing, *ESA-EUSC 2005: Image Information Mining - Theory and Application to Earth Observation*.

Dalal, N. & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection, *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, California, USA, pp. 886–893.

Freeman, W. T. & Roth, M. (1995). Orientation histograms for hand gesture recognition, *Intl. IEEE Workshop on Automatic Face and Gesture- Recognition*, Zurich,Switzerland, pp. 296–301.

Freund, Y. & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm, *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 148–156.

Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection, *The Journal of Machine Learning Research* **3**: 1157–1182.

Hall, M. A. (2000). Correlation-based feature selection for discrete and numeric class machine learning, *Proceedings 17th International Conference on Machine Learning (ICML)*, Morgan Kaufmann, pp. 359–366.

Kira, K. & Rendell, L. A. (1992). A practical approach to feature selection, *Proceedings of the 9th International Workshop on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 249–256.

Kohavi, R. & John, G. H. (1997). Wrappers for feature subset selection, *Artificial Intelligence* **97**(1-2): 273–324.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF, *Proceedings of the European Conference on Machine Learning (ECML)*.

Le, D. D. & Satoh, S. (2004). Feature Selection by AdaBoost for SVM-based Face Detection, *Forum on Information Technology* pp. 183–186.

Lepetit, V. & Fua, P. (2006). Keypoint Recognition using Randomized Trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(9): 1465–1479.

Leyrit, L., Chateau, T., Tournayre, C. & Lapresté, J.-T. (2008). Association of AdaBoost and Kernel Based Machine Learning Methods for Visual Pedestrian Recognition, *IEEE Intelligent Vehicles Symposium (IV 2008)*, Eindhoven, Netherlands.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* **60**(20): 91–110.

Moutarde, F., Stanciulescu, B. & Breheret, A. (2008). Real-time visual detection of vehicles and pedestrians with new efficient adaBoost features, *Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV) of International Conference on Intelligent RObots and Systems (IROS)*, Nice, France.

Munder, S. & Gavrila, D. (2006). An Experimental Study on Pedestrian Classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* **28**(11).

Papageorgiou, C. & Poggio, T. (2000). A trainable system for object detection, *International Journal of Computer Vision* **38**(1): 15–33.

Rakotomamonjy, A. (2003). Variable selection using svm-based criteria, *Journal of Machine Learning Research* **3**: 1357–1370.

Shashua, A., Gdalyahu, Y. & Hayun, G. (2004). Pedestrian Detection for Driving Assistance: Single-frame Classification and System Level Performance, *Proceedings of the IEEE Intelligent Vehicle Symposium (IV)*, Parma, Italy.

Suard, F., Rakotomamonjy, A., Bensrhair, A. & Broggi, A. (2006). Pedestrian detection using infrared images and histograms of oriented gradients, *Proceedings of the IEEE Conference of Intelligent Vehicles (IV)*, Tokyo, Japan, pp. 206–212.

Tipping, M. E. (2001). Sparse Bayesian Learning and the Relevance Vector Machine, *Journal of Machine Learning Research* **1**: 211–244.

Vapnik, V. (1998). *Statistical Learning Theroy*, Wiley.

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features, *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 511–518.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. & Vapnik, V. (2000). Feature Selection for SVMs, *Neural Information Processing Systems*, pp. 668–674.

Yu, L. & Liu, H. (2003). Feature selection for high-dimensional data: a fast correlation-based filter solution, *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 856–863.

7

From Feature Space to Primal Space: KPCA and Its Mixture Model

Haixian Wang
Key Laboratory of Child Development and Learning Science of Ministry of Education, Research Center for Learning Science, Southeast University, Nanjing, Jiangsu 210096, P. R. China

1. Introduction

Kernel principal component analysis (KPCA) (Schölkopf et al., 1998) has proven to be an exceedingly popular technique in the fields of machine learning and pattern recognition, and is discussed at length in literature. KPCA is to perform linear PCA (Hotelling, 1933; Jolliffe, 2002) in a high- (and possibly infinite-) dimensional kernel-defined feature space that is typically induced by a nonlinear mapping. In implementation, the so-called kernel trick is employed. Namely, KPCA is expressed in terms of dot products between the mapped data points, and the dot products are then evaluated by substituting an *a priori* kernel function. KPCA has demonstrated to be an incredibly useful tool for many application areas including handwritten digits recognition and de-noising (Schölkopf et al., 1998; Mika et al., 1999; Schölkopf et al., 1999), nonlinear regression (Rosipal et al., 2001), face recognition (Kim et al., 2002a; Yang, 2002; Kong et al., 2005), and complex image analysis (Kim et al., 2005; Li et al., 2008).

In practice, however, we are often confronted with the situation that needs to process a large number of data points. This raises a problem for KPCA, since KPCA has to store and diagonalize the *kernel matrix* (also known as Gram matrix), whose size is equal to the square of the number of training samples. So, for large scale data set, KPCA would consume large storage space and be computationally intensive (with time complexity $O(n^3)$, a cubic growth with n, where n is the number of the training samples). Then it is impractical for KPCA to be applied in some circumstances. Another attendant problem is that eig-decomposing large matrix directly suffers from the issue of numerical accuracy. Some algorithms have been developed to address the drawbacks associated with KPCA. By considering KPCA from a probabilistic point of view, Rosipal and Girolami (2001) presented an expectation maximization (EM) (Dempster et al., 1977; McLachlan & Krishnan, 1997) method for carrying out KPCA. Their algorithm is of computational complexity $O(pn^2)$ per iteration, where p is the number of extracted components. Whereas the EM algorithm for KPCA does alleviate computational demand, there exists a rotational ambiguity with the algorithm. To remove the obscurity, a constrained EM algorithm for KPCA (and PCA) was formulated based on coupled probability model (Ahn & Oh, 2003). Also, one deficiency of these EM-type algorithms is that the kernel matrix still required to be stored. Kim et al. (2005) then derived the kernel Hebbian algorithm (KHA), which was the counterpart of the generalized Hebbian algorithm (GHA) (Sanger, 1989), to iteratively perform KPCA, where only linear order memory complexity was involved. However, the price one has to pay for this saving is that the time complexity is not under control. Motivated by the idea “divide and rule”, Zheng et el. (2005) proposed another improved algorithm for KPCA as follows. First, the entire data set was divided into some smaller data sets, then the sample covariance matrix of each smaller data set was approximately computed, and finally kernel principal components were extracted by combining these approximate covariance matrices. With their method, the computational demand and memory requirement are effectively relieved. However, the advantages relate with many factors such as the required accuracy of extracted components, the number of the divided smaller data sets (which is usually empirically set), and the data to be processed. As a generic methodology, another thread of speeding up kernel machine learning is to seek a low-rank approximation to the kernel matrix. Since, as noted by several researchers, the spectrum of the kernel matrix tends to decay rapidly, the low-rank approximation often achieves sufficient precision of the requirement. Williams and Seeger (2001) used Nyström method to compute the approximate eigenvalue decomposition of the kernel matrix. Also, Smola and Schölkopf (2000) presented a sparse greedy approximation technique. These two methods yield similar forms and performances.

Another limitation of KPCA is that it defines only a *global* projection of the samples. When the distribution of the data points is complex and non-convex, a global subspace based on KPCA may fail to deliver good performance in terms of feature extraction and recognition. In input space, Tipping and Bishop (1999) and Roweis and Ghahramani (1999) introduced mixture of PCA to remedy the same shortcoming of PCA. Kim et al. (2002b) used mixture-of-eigenfaces for face recognition. There are many other papers on face recognition using mixture method, but as they do not focus on KPCA, references are omitted.

The contributions of this chaper are twofold: Firstly, viewing KPCA as a problem in primal space with the “samples” created by using the incomplete Cholesky decomposition, we show that KPCA is equivalent to performing linear PCA in the primal space using the created samples. So, the same kernel principal components as the standard KPCA are produced. Consequently, all the improved methods dealing with linear PCA (such as the constrained EM algorithm and the GHA method mentioned above), as well as directly diagonalizing the covariance matrix, could be applied to the created samples in the primal space to extract kernel principal components. Theoretical analysis and experimental results on both artificial and real data have shown the superiority of the proposed method for performing KPCA in terms of computational efficiency and storage space, especially when the number of the data points is large. Secondly, we extend KPCA to a mixture of local KPCA models by applying the mixture model of the probabilistic PCA in the primal space. While KPCA uses one set of features to model the data points, the mixture of KPCA uses more than one set of features. Therefore, the mixture of KPCA is expected to represent data more effectively and has better recognition performance than KPCA, which is also confirmed by the experiments.

The remainder of this chaper is organized as follows. The standard KPCA is briefly reviewed in Section 2, and in Section 3, we formulate KPCA in the primal space using the incomplete Cholesky decomposition. Next, we extend KPCA to its mixture model in Section 4. Experimental results are presented in Section 5. In Section 6, we draw the conclusion.

2. Kernel Principal Component Analysis

Suppose $\mathbf{x}_i \in \mathbb{R}^l, i = 1, \ldots, n$, are n observations. The basic idea of KPCA is as follows. First, the samples are mapped into some potentially high- (and possibly infinite-) dimensional *feature* *space* $\mathcal{F}$

$$\phi : \mathbb{R}^l \rightarrow \mathcal{F},\ \mathbf{x}_i \mapsto \phi(\mathbf{x}_i),\ (i = 1, \ldots, n) \tag{1}$$

where ϕ is a typically nonlinear function. Then a standard linear PCA is performed in $\mathcal{F}$ using the mapped samples. In evaluation, we don't have to compute the mapping ϕ explicitly. The mapped samples occur in the forms of dot products, say between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$, which are computed by choosing a *kernel function* k:

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)). \tag{2}$$

The mapping ϕ into $\mathcal{F}$ such that (2) stands exists if k is a positive definite kernel, thanks to Mercer's theorem of functional analysis. So, the mapping ϕ and thus $\mathcal{F}$ are fixed implicitly via the function k. The dth-order polynomial kernel, $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$, Gaussian kernel with width $\sigma > 0$, $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$, and sigmoid kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(a(\mathbf{x}_i \cdot \mathbf{x}_j) + b)$ are commonly used Mercer kernels.

For notation simplicity, the mapped samples are assumed to be centered, i.e. $\sum_{i=1}^{n} \phi(\mathbf{x}_i) = 0$. We wish to find eigenvalues $\lambda > 0$ and associated eigenvectors $\mathbf{v} \in \mathcal{F} \setminus \{0\}$ of the covariance matrix of the mapped samples $\phi(\mathbf{x}_i)$, given by

$$\mathbf{C}^{\phi} = \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^{\mathsf{T}}, \tag{3}$$

where T denotes the transpose of a vector or matrix. Since the mapping ϕ is implicit or $\mathbf{C}^{\phi}$ is very high dimensional, direct eigenvalue decomposition will be intractable. The difficulty is circumvented by using the so-called kernel trick; that is, linear PCA in $\mathcal{F}$ is formulated such that all the occurrences of ϕ are in the forms of dot products. And the dot products are then replaced by the kernel function k. So, dealing with the ϕ-mapped data explicitly is avoided. Specifically, since $\lambda\mathbf{v} = \mathbf{C}^{\phi}\mathbf{v}$, all solutions $\mathbf{v}$ with $\lambda \neq 0$ fall in the subspace spanned by $\{\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)\}$. Therefore, $\mathbf{v}$ could be linearly represented by $\phi(\mathbf{x}_i)$:

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i), \tag{4}$$

where $\alpha_i\ (i = 1, \ldots, n)$ are coefficients. The eigenvalue problem is then reduced as the following equivalent problem

$$\lambda(\phi(\mathbf{x}_j) \cdot \mathbf{v}) = (\phi(\mathbf{x}_j) \cdot \mathbf{C}^{\phi}\mathbf{v}) \text{ for all } j = 1, \ldots, n. \tag{5}$$

Substituting (3) and (4) into (5), we arrive at the eigenvalue equation

$$n\lambda\mathbf{K}\alpha = \mathbf{K}^2\alpha \Rightarrow n\lambda\alpha = \mathbf{K}\alpha, \tag{6}$$

where α denotes a column vector with entries $\alpha_1, \ldots, \alpha_n$, and $\mathbf{K}$, called kernel matrix, is an $n \times n$ matrix with elements defined as

$$\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)). \tag{7}$$

Assume that $\mathbf{t}$ is a testing point whose ϕ-image is $\phi(\mathbf{t})$ in $\mathcal{F}$. We calculate its kernel principal components by projecting $\phi(\mathbf{t})$ onto the kth eigenvectors $\mathbf{v}^k$. Specifically, the kth kernel principal components corresponding ϕ are

$$(\mathbf{v}^k \cdot \phi(\mathbf{t})) = \sum_{i=1}^{n} \alpha_i^k (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{t})) = \sum_{i=1}^{n} \alpha_i^k k(\mathbf{x}_i, \mathbf{t}), \tag{8}$$

where α^k has been normalized such that $\lambda_k(\alpha^k \cdot \alpha^k) = 1$. Note that centering the vectors $\phi(\mathbf{x}_i)$ and $\mathbf{t}$ in $\mathcal{F}$ is realized by centering the corresponding kernel matrices (Schölkopf et al., 1998).

3. Kernel Principal Component Analysis in Primal Space

In this section, we will derive KPCA in the primal space with the samples created by using the incomplete Cholesky decomposition. Let $\phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]$ be the data matrix containing all the ϕ-mapped training samples as columns. By partial Gram-Schmidt orthonormalization (Cristianini et al., 2002), we factorize the data matrix $\phi(\mathbf{X})$ as

$$\phi(\mathbf{X}) = \mathbf{QR}, \tag{9}$$

where $\mathbf{Q}$ has m orthonormal columns, $\mathbf{R} \in \mathbb{R}^{m \times n}$ is an upper triangular matrix with positive diagonal elements, and m is the rank of $\phi(\mathbf{X})$. Note that the matrix $\mathbf{R}$ could be evaluated row by row without computing ϕ explicitly. The partial Gram-Schmidt procedure pivots the samples and selects the linearly independent samples in the feature space $\mathcal{F}$. The orthogonalized version of the selected independent samples, i.e. the columns of $\mathbf{Q}$, is thus used as a set of basis. All ϕ-mapped data points could be linearly represented using the basis. Specifically, the ith column of the matrix $\mathbf{R}$ are the coefficients for the linear representation of $\phi(\mathbf{x}_i)$ using the columns of $\mathbf{Q}$ as the basis. So, the columns of $\mathbf{R}$ are, in fact, the new coordinates in the feature space of the corresponding data points of $\phi(\mathbf{X})$ using the basis. By (9), the following decomposition of the kernel matrix is yielded:

$$\mathbf{K} = \phi(\mathbf{X})^{\mathsf{T}}\phi(\mathbf{X}) = \mathbf{R}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{QR} = \mathbf{R}^{\mathsf{T}}\mathbf{R}, \tag{10}$$

which is the incomplete Cholesky decomposition (Fine & Scheinberg, 2001; Bach & Jordan, 2002).

From (10), if defining a new mapping

$$\tilde{\phi} : \mathcal{F} \to \mathbb{R}^m,\ \phi(\mathbf{x}_i) \mapsto \mathbf{r}_i,\ (i = 1, \ldots, n) \tag{11}$$

where $\mathbf{r}_i$ is the ith column of $\mathbf{R}$, then the n vectors $\{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$ give rise to the same Gram matrix $\mathbf{K}$ (Shawe-Taylor & Cristianini, 2004); that is

$$(\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{r}_i \cdot \mathbf{r}_j). \tag{12}$$

The space $\mathbb{R}^m$ is referred to as the *primal space*, and $\mathbf{r}_i$ $(i = 1, \ldots, n)$ are viewed as "samples". From (9), we see that, if $\phi(\mathbf{x}_i)$ are centered, then $\mathbf{r}_i$ are also centered. In other words, $\mathbf{r}_i$ could be centered by centering the kernel matrix $\mathbf{K}$. By using the samples $\mathbf{r}_i$ created in the primal space $\mathbb{R}^m$, we have the following

Theorem 1. *Given observations $\mathbf{x}_i$ $(i = 1, \ldots, n)$ and kernel function k, KPCA is equivalent to performing linear PCA in the primal space $\mathbb{R}^m$ using the created samples $\mathbf{r}_i$ $(i = 1, \ldots, n)$, both of which produce the same kernel principal components.*

Proof. It suffices to note that the dot products between the ϕ-mapped samples in the feature space $\mathcal{F}$ are the same with that between the corresponding $\tilde{\phi}$-mapped samples in the primal space $\mathbb{R}^m$, and linear PCA in both the feature space $\mathcal{F}$ (i.e., KPCA) and the primal space $\mathbb{R}^m$ could be represented through the forms of dot products between samples. The equivalence between KPCA and linear PCA in the primal space is schematically illustrated as follows:

$$\begin{array}{ccc} \boxed{\text{diagonalize Gram matrix } \mathbf{K} = \phi(\mathbf{X})^\top \phi(\mathbf{X})} & \equiv & \boxed{\text{diagonalize Gram matrix } \mathbf{K} = \mathbf{R}^\top \mathbf{R}} \\ \Updownarrow \text{ through dot products} & & \Updownarrow \text{ through dot products} \\ \boxed{\text{KPCA, i.e., linear PCA in } \mathcal{F} \text{ with samples } \phi(\mathbf{x}_i)} & & \boxed{\text{linear PCA in } \mathbb{R}^m \text{ with samples } \mathbf{r}_i} \end{array}$$

The theorem is thus established. □

In the primal space, we could, of course, carry out linear PCA by using the dual expression of dot products; but this obviously makes the motivation for creating the new samples in the primal space useless. Considering the dimension of the primal space, m, is small, we perform linear PCA by directly diagonalizing the $m \times m$ covariance matrix of the $\tilde{\phi}$-mapped data points $\mathbf{r}_i$, given by

$$\mathbf{C}^{\tilde{\phi}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{r}_i \mathbf{r}_i^\top = \frac{1}{n}\mathbf{R}\mathbf{R}^\top. \tag{13}$$

Let $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_m$ be the eigenvalues of $\mathbf{C}^{\tilde{\phi}}$, and $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_m$ the corresponding eigenvectors. We proceed to compute the kernel principal components of the testing point $\mathbf{t}$. Firstly, we need to compute its $\tilde{\phi}$-image. We carry out the projections of $\phi(\mathbf{t})$ onto the basis vectors, i.e., the columns of $\mathbf{Q}$. This is achieved by calculating an extensional column of the matrix $\mathbf{R}$ in the partial Gram-Schmidt procedure (Shawe-Taylor & Cristianini, 2004). The kernel principal components corresponding to ϕ are then computed as

$$(\tilde{\mathbf{v}}_k \cdot \bar{\mathbf{r}}) \tag{14}$$

for $k = 1, \ldots, p$, where $\bar{\mathbf{r}}$ is the $\tilde{\phi}$-image of $\phi(\mathbf{t})$ and p ($\leq m$) is the number of components.

Alternatively, all improved methods dealing with linear PCA could be applied to $\mathbf{R}$ in the primal space. For example, with the constrained EM algorithm for PCA (Ahn & Oh, 2003), we obtain iterative formula

$$\text{E step:} \qquad \mathbf{Z} = (\mathcal{L}(\Gamma^\top \Gamma))^{-1} \Gamma^\top \mathbf{R}, \tag{15}$$

$$\text{M step:} \qquad \Gamma^{\text{new}} = \mathbf{R}\mathbf{Z}^\top (\mathcal{U}(\mathbf{Z}\mathbf{Z}^\top))^{-1}, \tag{16}$$

where the element-wise lower operator $\mathcal{L}$ is defined such as $\mathcal{L}(w_{st}) = w_{st}$ for $s \geq t$ and is zero otherwise, the upper operator $\mathcal{U}$ is defined such as $\mathcal{U}(w_{st}) = w_{st}$ for $s \leq t$ and is zero otherwise, and $\mathbf{Z}$ denotes the $p \times n$ matrix of latent variables. The matrix Γ at convergence is equal to $\Gamma = \mathbf{V}\Lambda$, where the columns of $\mathbf{V} = [\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_p]$ are the first p eigenvectors of $\mathbf{C}^{\tilde{\phi}}$, with corresponding eigenvalues $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_p$ forming the diagonal matrix Λ. Another extensively used iterative method for PCA is the generalized Hebbian algorithm (GHA) (Sanger, 1989). Based on GHA, the $m \times p$ eigenvectors matrix $\mathbf{V}$ corresponding to the p largest eigenvalues is updated according to the rule

$$\mathbf{V}(t+1) = \mathbf{V}(t) + \delta(t)\left(\mathbf{r}(t)\mathbf{y}(t)^\top - \mathbf{V}(t)\mathcal{U}(\mathbf{y}(t)\mathbf{y}(t)^\top)\right), \tag{17}$$

where $\mathbf{y} = \mathbf{V}^\top \mathbf{r}$ is the principal component of $\mathbf{r}$, $\delta(t)$ is a learning rate parameter. Here, the argument t denotes a discrete time when a sample $\mathbf{r}(t)$ is selected randomly from all the samples $\mathbf{r}_i$. It has been shown by Sanger (1989) that, for proper setting of learning rate $\delta(t)$ and initialization $\mathbf{V}(0)$, the columns of $\mathbf{V}$ converge to the eigenvectors of $\mathbf{C}^{\tilde{\phi}}$ as t tends to infinity.

In summary, the procedure of the proposed algorithm for performing KPCA is outlined as follows:

1. Perform the incomplete Cholesky decomposition for the training points as well as testing points to obtain $\mathbf{R}$ and $\bar{\mathbf{r}}$;
2. Compute the p leading eigenvectors corresponding to the first p largest eigenvalues of the covariance matrix $\mathbf{C}^{\tilde{\Phi}}$ (defined in (13)) by (a) directly diagonalizing $\mathbf{C}^{\tilde{\Phi}}$, (b) using the constrained EM algorithm (according to (15) and (16)), or (c) using the GHA algorithm (according to (17));
3. Extract kernel principal components by projecting each testing point onto the eigenvectors (according to (14)).

Complexity analysis. The computational complexity of performing the incomplete Cholesky decomposition is of the order $O(m^2n)$, and the storage requirement is $O(mn)$. Next, if one explicitly evaluates the covariance matrix $\mathbf{C}^{\tilde{\Phi}}$ followed by diagonalization to obtain the eigenvectors, the computational and storage complexity are $O(m^2n + m^3)$ and $O((p+m)m)$, respectively. When $p \ll m$, it is possible to obtain computational savings by using the constrained EM algorithm, the time and storage complexity of which are respectively $O(pmn)$ per iteration and $O(p(m+n))$. If using the GHA method, the time complexity is $O(p^2m)$ per iteration and storage complexity $O(pm)$. The potential efficiency gains, nevertheless, depend on the number of iterations needed to reach the required precision and the ratio of m to p. As one has seen, the proposed method do not need to store the kernel matrix $\mathbf{K}$, the storage of which is $O(n^2)$, and its computational complexity compares favorably with that of the traditional KPCA, which scales as $O(n^3)$.

4. Mixture of Kernel Principal Component Analysis Models

Single KPCA provides only a globally linear model for data representation in a low dimensional subspace. It may be insufficient to model (heterogeneous) data with large variation. One remedy method is to model the complex data with a mixture of local linear sub-models. Considering KPCA in its equivalent form in the primal space, and using the mixture model in the primal space (Tipping & Bishop, 1999), we extend KPCA to a mixture of local KPCA models. While KPCA uses one set of features for the data points, the mixture of KPCA uses more than one set of features. Mathematically, in the primal space, we suppose that $\mathbf{r}_1, \ldots, \mathbf{r}_n$ are generated independently from a mixture of g underlying populations with unknown proportion $\pi_1, \ldots, \pi_g$

$$\mathbf{r}_i = \mu_j + \Gamma_j \mathbf{z}_{ij} + \varepsilon_{ij}, \text{with probability } \pi_j, (j = 1, \ldots, g; i = 1, \ldots, m) \tag{18}$$

where μ_j is a m-dimensional non-random vector, the $m \times p$ matrix Γ_j is known as the *factor loading matrix*, $\mathbf{z}_{ij}$ are p-dimensional latent (unobservable) variables (also known as *common factors*), ε_{ij} are m-dimensional error variables, and π_j is the corresponding mixing proportion with $\pi_j > 0$ and $\sum_{j=1}^{g} \pi_j = 1$. The generative model (18) assumes that $\mathbf{z}_{1j}, \ldots, \mathbf{z}_{nj}$ are independently and identically distributed (i.i.d.) as Gaussian with zero mean and identity covariance, $\varepsilon_{1j}, \ldots, \varepsilon_{nj}$ i.i.d. as Gaussian with mean zero and covariance matrix $\sigma_j^2 I_m$, and $\mathbf{z}_{ij}$ is independent with ε_{ij} and their joint distribution is Gaussian.

In the case of $g = 1$, it was shown by Tipping and Bishop (1999) and Roweis and Ghahramani (1999) that, as the noise level σ_1^2 becomes infinitesimal, the PCA model in the primal space was recovered, which just is KPCA as shown in Section 3. So, model (18) is an extension to KPCA. We refer to (18) as mixture of KPCA (MKPCA). Note that a separate mean vector μ_j is associated with each component of the g mixture model, therefore allowing each component to model the data covariance structure in different regions of the primal space.

Since the true classification of $\mathbf{r}_i$ into components are unknown, we use the marginal probability density function (p.d.f.) that is a g-component Gaussian mixture p.d.f.

$$f(\mathbf{r}_i;\Theta) = \sum_{j=1}^{g} \pi_j\, \varphi(\mathbf{r}_i;\mu_j,\Sigma_j) \tag{19}$$

for the observations, where $\varphi(\mathbf{r}_i;\mu_j,\Sigma_j)$ is the Gaussian p.d.f. with mean μ_j and variance $\Sigma_j = \Gamma_j\Gamma_j^{\mathsf{T}} + \sigma_j^2 I_m$, the model parameters are given by $\Theta = (\mu_1,\ldots,\mu_g,\Gamma_1,\ldots,\Gamma_g,\sigma_1^2,\ldots,\sigma_g^2,\pi_1,\ldots,\pi_g)$. For log-likelihood maximization, the model parameters could be estimated via the EM algorithm as follows (Tipping & Bishop, 1999). The E-step is

$$\hat{z}_{ij}^{(k)} = \frac{\hat{\pi}_j^{(k)}\,\varphi(\mathbf{r}_i;\hat{\mu}_j^{(k)},\hat{\Sigma}_j^{(k)})}{\sum_{i=1}^{g}\hat{\pi}_i^{(k)}\,\varphi(\mathbf{r}_i;\hat{\mu}_i^{(k)},\hat{\Sigma}_i^{(k)})}. \tag{20}$$

In the M-step, the parameters are updated according to

$$\hat{\pi}_j^{(k+1)} = \frac{1}{n}\sum_{i=1}^{n}\hat{z}_{ij}^{(k)}, \tag{21}$$

$$\hat{\mu}_j^{(k+1)} = \frac{\sum_{i=1}^{n}\hat{z}_{ij}^{(k)}\mathbf{r}_i}{\sum_{i=1}^{n}\hat{z}_{ij}^{(k)}}, \tag{22}$$

$$\hat{\Gamma}_j^{(k+1)} = \hat{\mathbf{S}}_j^{(k+1/2)}\hat{\Gamma}_j^{(k)}\left(\hat{\sigma}_j^{2(k)}I_m + (\hat{\sigma}_j^{2(k)}I_p + \hat{\Gamma}_j^{(k)\mathsf{T}}\hat{\Gamma}_j^{(k)})^{-1}\hat{\Gamma}_j^{(k)\mathsf{T}}\hat{\mathbf{S}}_j^{(k+1/2)}\hat{\Gamma}_j^{(k)}\right)^{-1}, \tag{23}$$

$$\hat{\sigma}_j^{2(k+1)} = \frac{1}{m}\mathrm{tr}\left(\hat{\mathbf{S}}_j^{(k+1/2)} - \hat{\mathbf{S}}_j^{(k+1/2)}\hat{\Gamma}_j^{(k)}(\hat{\sigma}_j^{2(k)}I_p + \hat{\Gamma}_j^{(k)\mathsf{T}}\hat{\Gamma}_j^{(k)})^{-1}\hat{\Gamma}_j^{(k+1)\mathsf{T}}\right), \tag{24}$$

where

$$\hat{\mathbf{S}}_j^{(k+1/2)} = \frac{1}{n\hat{\pi}_j^{(k+1)}}\sum_{i=1}^{n}\hat{z}_{ij}^{(k)}(\mathbf{r}_i - \hat{\mu}_j^{(k+1)})(\mathbf{r}_i - \hat{\mu}_j^{(k+1)})^{\mathsf{T}}. \tag{25}$$

In the case $\sigma_j^2 \to 0$, the converged $\hat{\Gamma}_j$ contains the (scaled) eigenvectors of the local covariance matrix $\hat{\mathbf{S}}_j$. Each component performs a local PCA weighted by the mixing proportion. And the $\hat{\Gamma}_j$ in the limit case are updated as

$$\hat{\Gamma}_j^{(k+1)} = \hat{\mathbf{R}}_j^{(k+1/2)}\hat{\mathbf{Z}}_j^{(k)\mathsf{T}}(\mathcal{U}(\hat{\mathbf{Z}}_j^{(k)}\hat{\mathbf{Z}}_j^{(k)\mathsf{T}}))^{-1}, \tag{26}$$

where

$$\hat{\mathbf{Z}}_j^{(k)} = (\mathcal{L}(\hat{\Gamma}_j^{(k)\mathsf{T}}\hat{\Gamma}_j^{(k)}))^{-1}\hat{\Gamma}_j^{(k)\mathsf{T}}\hat{\mathbf{R}}_j^{(k+1/2)},$$

and

$$\hat{\mathbf{R}}_j^{(k+1/2)} = \left(\sqrt{\frac{\hat{z}_{1j}^{(k)}}{n\hat{\pi}_j^{(k+1)}}}(\mathbf{r}_1 - \hat{\mu}_j^{(k+1)}),\ldots,\sqrt{\frac{\hat{z}_{nj}^{(k)}}{n\hat{\pi}_j^{(k+1)}}}(\mathbf{r}_n - \hat{\mu}_j^{(k+1)})\right).$$

Methods for performing KPCA	Size of matrix needed to be diagonalized	Training time (seconds)	Storage space (M bytes)
Standard KPCA	2000×2000	293	90
Proposed KPCA	233×233	36	2
MKPCA	N.A.	63	5

Table 1. Comparison of training time and storage space on the toy example with 2000 data points.

Number of	Standard KPCA			Proposed KPCA			MKPCA		
features	*d*=2	3	4	*d*=2	3	4	*d*=2	3	4
32	93.6	93.6	92.5	93.5	93.6	92.4	94.5	94.3	93.5
64	94.3	93.1	93.0	94.2	93.0	93.0	**95.2**	94.2	**94.1**
128	94.4	93.7	**93.2**	**94.4**	93.5	**93.1**	95.0	94.1	94.0
256	**94.5**	93.5	92.9	94.4	**93.8**	92.9	94.9	**94.4**	94.0
512	94.3	**93.9**	92.8	N.A.	93.8	92.8	N.A.	94.1	93.6
1024	94.5	93.9	92.4	N.A.	N.A.	92.4	N.A.	N.A.	93.1

Table 2. Recognition rates of the 2007 testing points of the USPS handwritten digit database using the standard KPCA, proposed KPCA and MKPCA methods with polynomial kernel of degree two through four.

When the noise level becomes infinitesimal, the component p.d.f. $\varphi(\mathbf{r}_i; \hat{\mu}_i^{(k)}, \hat{\Sigma}_i^{(k)})$ in (20) is singular. It, in probability 1, falls in the p-dimensional subspace $span\{\hat{\Gamma}_j^{(k)}\}$, i.e.,

$$\varphi(\mathbf{r}_i; \hat{\mu}_i^{(k)}, \hat{\Sigma}_i^{(k)}) = (2\pi)^{-p/2} \left(\det(\hat{\Gamma}_j^{(k)\top} \hat{\Gamma}_j^{(k)})\right)^{-1/2} \exp\left(-\hat{\mathbf{a}}^{(k)\top} \hat{\mathbf{a}}^{(k)}/2\right), \tag{27}$$

where $\hat{\mathbf{a}}^{(k)} = (\hat{\Gamma}_j^{(k)\top} \hat{\Gamma}_j^{(k)})^{-1} \hat{\Gamma}_j^{(k)\top} (\mathbf{r}_i - \hat{\mu}_j^{(k)})$.

As can be seen, by applying the KPCA mixture model, all the observations are softly divided into g clusters each modelled by a local KPCA. We use the most appropriate local KPCA for a given observation. Based on the probabilistic framework, a natural choice is to assign the observation to the cluster belong to which its posterior probability is the largest.

5. Experiments

In this section, we will use both artificial and real data sets to compare the performance of the proposed method with that of the standard KPCA (Schölkopf et al., 1998). In the first example, we use toy data to visually compare the results by projecting testing points onto extracted principal axes, and to show that the proposed method is superior to the standard KPCA in terms of time and storage complexity. In the second example, we perform the experiment of handwritten digital character recognition to further illustrate the effectiveness of the proposed method. All these experiments are run with the settings of 3.06GHz CPU and 3.62GB RAM using *Matlab* software.

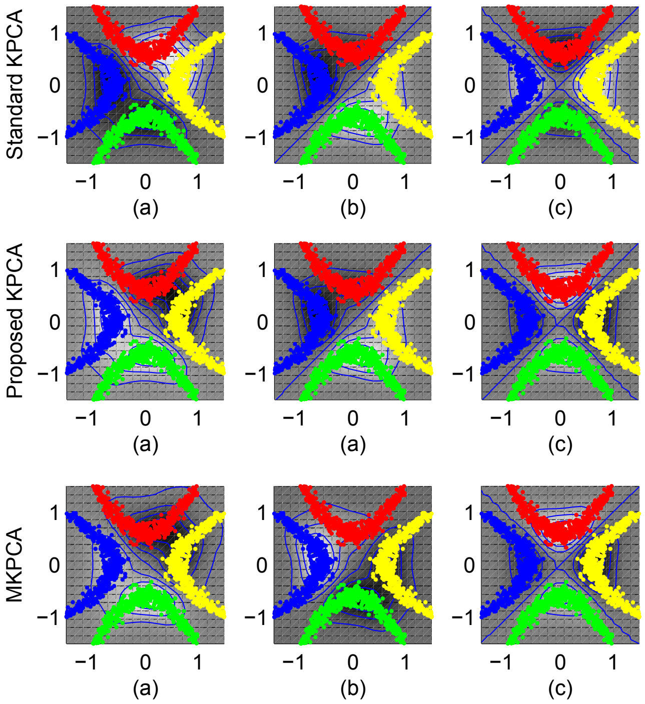


Fig. 1. From left to right, the first three kernel principal components extracted by the standard KPCA (top), proposed KPCA (middle), and MKPCA method with $g = 2$ (bottom), respectively, using the Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/0.1)$. The feature values are illustrated by shading and constant values are connected by contour lines.

5.1 Toy Example

The data are generated from four two-dimensional parabolic shape clusters that are vertically and horizontally mirrored by the function $y = x^2 + \epsilon$, where x-values are uniformly distributed within $[-1, 1]$ and ϵ is Gaussian distributed noise having mean 0.6 and standard deviation 0.1. We generate 500 data points for each parabolic shape, and use the Gaussian kernel with $\sigma^2 = 0.05$ through the experiment. The standard KPCA, proposed KPCA (by, in this experiment, using the incomplete Cholesky decomposition followed by directly diagonalizing $\mathbf{C}^{\tilde{\Phi}}$), and MKPCA methods are adopted to calculate the leading principal components of the data set. In using the two proposed methods, we choose 233 linearly independent samples from the entire 2000 samples, i.e., $m = 233$, during the incomplete Cholesky decomposition, and set $g = 2$ for performing the MKPCA. Therefore, the proposed KPCA method consumes much less time and storage space than that of the standard KPCA (to see table 1 for detailed comparison). In Fig. 1, we depict the first three kernel principal components extracted by the three methods. The features are indicated by shading and constant feature values are con-

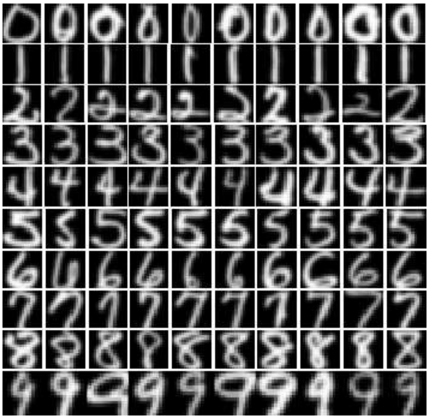


Fig. 2. Some digit images from the USPS database.

Methods for performing KPCA	Size of matrix needed to be diagonalized	Training time (seconds)	Storage space (M bytes)
Standard KPCA	5000×5000	6112	457
Proposed KPCA ($d = 2$)	371×371	114	41
Proposed KPCA ($d = 3$)	956×956	342	69
Proposed KPCA ($d = 4$)	1639×1639	634	118
MKPCA ($d = 2$)	N.A.	321	68
MKPCA ($d = 3$)	N.A.	901	115
MKPCA ($d = 4$)	N.A.	2213	304

Table 3. Comparison of training time and storage space on the USPS handwritten digit database with 5000 training points.

nected by contour lines. From Fig. 1, we see that the proposed KPCA method obtains almost the same results with that of the standard KPCA method (ignoring the sign difference), and both methods nicely separate the four clusters. For the data points, the average relative deviation of the principal components found by the standard KPCA (by diagonalizing the kernel matrix **K**) and by the proposed KPCA is less than 0.01. In this simple simulated experiment, the toy data are compact in the feature space, and are well modelled by the KPCA method. So, the MKPCA method doesn't show its advantage; in fact, most of the toy data belong to one component of the MKPCA model, since the estimated mixing proportions $\hat{\pi}_1 = 0.9645$ and $\hat{\pi}_2 = 0.0355$. As a result, the MKPCA method produces the similar result with that of the KPCA method.

5.2 Handwritten Digit Character Recognition

In the second example, we consider the recognition problem of handwritten digital character. The experiment is performed on the US Postal Service (USPS) handwritten digits database that are collected from mail envelopes in Buffalo, New York. The database contains 7291 training samples and 2007 testing samples for 10 numeral classes with dimensionality 256 (Schölkopf et al., 1998). Some digit samples are shown in Fig. 2. With this database, 5000 training points are chosen as training data and all the 2007 testing points are used as testing data. The polynomial kernel with various degree d is utilized in each trial to compute the kernel function. We employ the *nearest neighbor classifier* for classification role. In using MKPCA, we set $g = 2$. The recognition rates obtained by the three approaches are reported in Table 2, while the training times and storage spaces consumed are listed in Table 3. From Table 2, we see that the MKPCA method achieves the best recognition rate among the three systems. The standard KPCA and the proposed approach to performing KPCA have similar recognition rates. Nevertheless, the proposed KPCA reduces the time and storage complexity significantly.

6. Conclusion

We have presented an improved algorithm for performing KPCA especially when the size of training samples is large. This is achieved by viewing KPCA as a primal space problem with the “samples” produced via the incomplete Cholesky decomposition. Since the spectrum of the kernel matrix tends to decay rapidly, the incomplete Cholesky decomposition, as an elegant low-rank approximation to the kernel matrix, arrives at sufficient accuracy. Compared with the standard KPCA method, the proposed KPCA method reduces the time and storage requirement significantly for the case of large scale data set.

In order to provide a locally linear model for the data projection onto a low dimensional subspace, we extend KPCA to a mixture of local KPCA models by applying mixture model of PCA in the primal space. MKPCA supplies an alternative choice to model data with large variation. The mixture model outperforms the standard KPCA in terms of recognition rate. The methodology introduced in this chaper could be applied to other kernel-based algorithms, provided the algorithm could be expressed through dot products.

Acknowledgement

This work was supported by Specialized Research Fund for the Doctoral Program of Higher Education of China under grant 20070286030, and National Natural Science Foundation of China under grants 60803059 and 10871001.

7. References

Ahn, J. H. & Oh, J. H. (2003). A constrained EM algorithm for principal component analysis, *Neural Computation* 15: 57–65.

Bach, F. R. & Jordan, M. I. (2002). Kernel independent component analysis, *Journal of Machine Learning Research* 3: 1–48.

Cristianini, N., Lodhi, H. & Shawe-Taylor, J. (2002). Latent semantic kernels, *Journal of Intelligent Information System* 18: 127–152.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data using the EM algorithm (with discussion), *J. R. Statist. Soc. Ser. B.* 39: 1–38.

Fine, S. & Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representation, *Technical Report RC 21911*, IBM T.J. Watson Research Center.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components, *J. Educat. Psychology* 24: 417–441.

Jolliffe I.T. (2002). *Principal Component Analysis,* Second edition, Springer-Verlag, New York.

Kim, K. I., Franz, M. O. & Schölkopf, B. (2005). Iterative kernel principal component analysis for image modelling, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27: 1351–1366.

Kim, K. I., Jung, K. & Kim, H. J. (2002a). Face recognition using kernel principal component analysis, *IEEE Signal Processing Letters* 19: 40–42.

Kim, H. C., Kim, D. & Bang, S. Y. (2002b). Face recognition using the mixture-of-eigenfaces method, *Pattern Recognition Letters* 23: 1549–1558.

Kong, H., Wang, L., Teoh, E. K., Li, X., Wang, J. G. & Venkateswarlu, R. (2005). Generalized 2D principal component analysis for face image representation and recognition, *Neural Networks* 18: 585–594.

Li, J., Li, M. L. & Tao, D. C. (2008). KPCA for semantic object extraction in images, *Pattern Recognition* 41: 3244-3250.

McLachlan, G. J. & Krishnan, T. (1997). *The EM Algorithm and Extensions*, Wiley, New York.

Mika, S., Schölkopf, B., Smola, A. J., Müller, K. R., Scholz, M. & Rätsch, G. (1999). Kernel PCA and de-Noising in feature spaces, *in* M. S. Kearns, S. A. Solla & D. A. Cohn (eds.), *Advances in Neural Information Processing Systems*, Vol. 11, MIT Press, Cambridge, Mass., pp. 536–542.

Rosipal, R. & Girolami, M. (2001). An expectation-maximization approach to nonlinear component analysis, *Neural Computation* 13: 505–510.

Rosipal, R., Girolami, M., Trejo, L. J. & Cichocki, A. (2001). Kernel PCA for feature extraction and de-noising in nonlinear regression, *Neural Computing & Applications* 10: 231–243.

Roweis, S. & Ghahramani, Z. (1999). A unifying review of linear gaussian models, *Neural Computation* 11: 305–345.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network, *Neural Networks* 2: 459–473.

Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K.-R., Rätsch, G. & Smola, A. J. (1999). Input space vs. feature space in kernel-based methods, *IEEE Transactions on Neural Networks* 10: 1000–1017.

Schölkopf, B., Smola, A. & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue prolem, *Neural Computation* 10: 1299–1319.

Shawe-Taylor, J. & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*, Cambridge University Press, England.

Smola, A. J. & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning, *Proceeding of the Seventeenth International Conference on Mchine Learning*, Morgan Kaufmann.

Tipping, M. E. & Bishop, C. M. (1999). Mixtures of probabilistic principal component analysers, *Neural Computation* 11: 443–482.

Williams C. K. I. & Seeger, M. (2001). Using the Nyström method to speed up kernel machines, *Advances in Neural Information Processing Systems*, Vol. 13, MIT Press.

Yang, M. H. (2002). Kernel eigenfaces vs. kernel fisherfaces: face recognition using kernel methods, *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, Washington, DC, pp. 215–220.

Zheng, W., Zou, C. & Zhao, L. (2005). An improved algorithm for kernel principal component analysis, *Neural Processing Letters* 22: 49–56.

Machine Learning for Multi-stage Selection of Numerical Methods*

Victor Eijkhout
Texas Advanced Computing Center, The University of Texas at Austin
USA

Erika Fuentes
Innovative Computing Laboratory, University of Tennessee
USA

Abstract
In various areas of numerical analysis, there are several possible algorithms for solving a problem. In such cases, each method potentially solves the problem, but the runtimes can widely differ, and breakdown is possible. Also, there is typically no governing theory for finding the best method, or the theory is in essence uncomputable. Thus, the choice of the optimal method is in practice determined by experimentation and 'numerical folklore'. However, a more systematic approach is needed, for instance since such choices may need to be made in a dynamic context such as a time-evolving system.
Thus we formulate this as a classification problem: assign each numerical problem to a class corresponding to the best method for solving that problem.
What makes this an interesting problem for Machine Learning, is the large number of classes, and their relationships. A method is a combination of (at least) a preconditioner and an iterative scheme, making the total number of methods the product of these individual cardinalities. Since this can be a very large number, we want to exploit this structure of the set of classes, and find a way to classify the components of a method separately.
We have developed various techniques for such multi-stage recommendations, using automatic recognition of super-clases. These techniques are shown to pay off very well in our application area of iterative linear system solvers.
We present the basic concepts of our recommendation strategy, and give an overview of the software libraries that make up the Salsa (Self-Adapting Large-scale Solver Architecture) project.

1. Introduction

In various areas of numerical analysis, there are several possible algorithms for solving a problem. Examples are the various direct and iterative solvers for sparse linear systems, or rou-

*This work was funded in part by the Los Alamos Computer Science Institute through the subcontract # R71700J- 29200099 from Rice University, and by the National Science Foundation under grants 0203984 and 0406403.

tines for eigenvalue computation or numerical optimization. Typically, there is no governing theory for finding the best method, or the theory is too expensive to compute. For instance, in iterative linear system solving, there are many preconditioners and iterative schemes; the convergence behaviour is determined by the decomposition of the right-hand side in terms of the eigenvectors of the preconditioned operator. However, computing this spectrum is expensive, in fact more expensive than solving the problem to begin with, and the dependency of the iterative scheme on this decomposition is not known in explicit form.

Thus, the choice of the optimal method is in practice determined by experimentation and 'numerical folklore'. However, a more systematic approach is needed, for instance since such choices may need to be made in a dynamic context such as a time-evolving system. Certain recent efforts have tried to tackle this situation by the application of various automatic learning techniques Arnold et al. (2000); Bhowmick et al. (2006); Dongarra et al. (2006); Houstis et al. (2000); Ramakrishnan & Ribbens (2000); Wilson et al. (2000). Such methods extract the problem features, typically relying on a database of prior knowledge, in order to recommend a suitable method. However, we may characterize this earlier research as either being too general and lacking in specific implementations, or being too limited to one area, lacking the conceptual necessary for generalization to other areas.

In this paper we present a theoretical framework for such recommender systems, as well as libraries that we have developed that embody these concepts. Our framework is more general than what has been studied before, in that we consider methods to be structured objects, consisting of a composition of algorithms. Correspondingly, we describe strategies for recommending methods that acknowledge this structure, rather than considering methods to be elements out of a flat set of elements.

In addition to the abstract mathematical description, we propose software interfaces implementing the various entities in the framework. Some of this software has been realized in our Salsa (Self-Adapting large-scale Solver Architecture) project; see Salsa Project (n.d.a;n).

We start by giving a brief introduction to our field in section 2, after which we formalize the problem in sections 3 and 4. Section 5 has a general discussion of intelligent method selection, while section 6 details strategies for recommending composite method objects. Practical experiments are reported in section 7.

2. Application: linear system solving

Many problems in engineering and in sciences such as physics, astronomy, climate modeling, reduce to a few basic numerical problems:

- Linear system solving: given a square, nonsingular, matrix A and a vector f, find a vector x such that $Ax = f$. Linear system solving also appears as a subproblem in nonlinear systems and in time-dependent problems.
- Eigenvalue calculations: given a matrix A, find the pairs $\langle u, \lambda \rangle$ such that $Au = \lambda u$. Some eigenvalue calculations again lead to linear systems.
- Numerical optimization: find the vector x that maximizes a function $f(x)$, often linear: $f(x) = a^t x$, under certain constraints on x: $g(x) = 0$.

For each of these numerical problems, more than one solution algorithm exists, often without a sufficient theory that allows for easy selection of the fastest or most accurate algorithm. This can be witnessed in the NEOS server Czyzyk et al. (1996; 1998), where a user can choose from various optimization algorithm, or the PETSc numerical toolkit Balay et al. (1999); Gropp & Smith (n.d.) which allows the user to combine various predefined solver building blocks. The aim is then to find an algorithm that will reliably solve the problem, and do so in the minimum amount of time.

> Since several algorithm candidates exist, without an analytic way of deciding on the best algorithm in any given instance, we are looking at the classification problem of using non-numerical techniques for recommending the best algorithm for each particular problem instance.

In this chapter we will focus on linear system solving. The textbook algorithm, Gaussian elimination, is reliable, but for large problems it may be too slow or require too much memory. Instead, often one uses *iterative solution methods*, which are based on a principle of successive approximation. Rather than computing a solution x directly, we compute a sequence $n \mapsto x_n$ that we hope converges to the solution. (Typically, we have residuals r_n that converge to zero, $r_n \downarrow 0$, but this behaviour may not be monotone, so no decision method can be based on its direct observation.)

2.1 Feature selection

As noted above, there is no simple, efficiently computable, way of deciding on the best algorithm for solving a linear system. Theoretically it is known that the behaviour of iterative solvers depends on the expansion of the right hand side in the eigenvector basis, but computing this expansion is far more expensive than solving the linear system to begin with. Also, there are a few negative results Greenbaum & Strakos (1996) that stand in the way of easy solutions.

However, using a statistical approach, we have no lack of features. For instance, we typically are concerned with sparse matrices, that is, matrices for which only a few elements per row are nonzero. Thus, we can introduce features that describe the sparsity structure of the matrix. From a theoretical point of view, the sparsity structure is not relevant, but in practice, it can correlate to meaningful features. For instance, high order finite element methods lead to larger numbers of nonzeros per row, and are typically associated with large condition numbers, which are an indication of slow convergence. Thus, indirectly, we can correlate these structural features with solver performance.

2.2 Multilevel decisions

As we remarked above, in essence we are faced with a classification problem: given a linear system and its features, what is the best method for solving it. However, the set of classes, the numerical methods, has structure to it that makes this a challenging problem.

It is a truism among numerical mathematicians that an iterative method as such is virtually worthless; it needs to be coupled to a 'preconditioner': a transformation B of the linear system that tames some of the adverse numerical properties. In its simplest form, applying a preconditioner is equivalent to solving the equivalent system

$$\tilde{A}x = \tilde{f}, \qquad \text{where } \tilde{A} = BA \text{ and } \tilde{f} = Bf$$

or

$$\tilde{A}\tilde{x} = f, \qquad \text{where } \tilde{A} = AB \text{ and } \tilde{x} = B^{-1}x \text{ so } x = B\tilde{x}.$$

Since there are many choices for the preconditioner, we now have a set of classes of cardinality the number of preconditioners times the number of iterative schemes.

Additionally, there are other transformations such as permutations P^tAP that can be applied for considerations such as load balancing in parallel calculations. In all, we are faced with a large number of classes, but in a set that is the cartesian product of sets of individual decisions that together make up the definition of a numerical method. Clearly, given this structure, treating the set of classes as just a flat collection will be suboptimal. In this chapter we consider the problem of coming up with better strategies, and evaluating their effectiveness.

2.3 Software aspects

The research reported in this paper has led to the development of a set of software libraries, together called the SALSA project, for Self-Adapting Large-scale Solver Architecture. The components here are

- Feature extraction. We have written a custom library, AnaMod, for Analysis Modules for this.
- Storage of features and runtime results. During runtime we use our NMD, for Numerical MetaData, library. For permanent storage we are using a standard MySQL database.
- Learning. We use various Matlab toolboxes here, though ultimately we want to move to an open source infrastructure.
- Process control. The overall testbed that controls feature extraction, decision making, and execution of the chosen algorithms, is a custom library named SysPro, for System preProcessors.

Since the overall goal is to optimize solution time, the feature extraction and process control software need to be fast. Though we will not elaborate on this point, this means that we occasionally have to trade accuracy for speed.

3. Formalization

In this section we formalize the notion of numerical problems and numerical methods for solving problems. Our formalization will also immediately be reflected in the design of the libraries that make up the Salsa project.

3.1 Numerical Problems and Solution Methods

The central problem of this paper, selecting the optimal algorith for a numerical problem, implies that we have a specific problem context in mind. That can be linear system solving, eigenvalue computations, numerical quadrature, et cetera. For each of these areas, there is a definition of a numerical problem: in the case of linear systems, a problem will be a pair $\langle A, b\rangle$ of matrix and righthand side; for eigenvalue problems it denotes a matrix plus optionally an interval for the eigenvalues, et cetera.

It would be pointless to capture this diversity of problems in a rigorous definition, so we will use a leave the concept of 'numerical problem' largely undefined. We denote the space of numerical problems by

$\mathbb{A}$: the set of numerical problems in a class

where the exact definition of this will depend on the problem area.

The implementation of numerical problems is similarly domain-specific. As an example, we give the following possible definition in the area of linear system solving: [1]:

[1] After this example we will tacitly omit the `typedef` line and only give the structure definition.

```
struct Problem_ {
    LinearOperator A;
    Vector RHS,KnownSolution,InitialGuess;
    DesiredAccuracy constraint;
};
typedef struct Problem_* Problem;
```

Corresponding to the problem space, there is a result space

$$\boxed{\mathbb{R} = \mathbb{S} \times \mathbb{T}\text{: results, the space of solutions plus performance measurements}} \tag{1}$$

containing computed solutions and performance measurements of the computing process. The following is an illustration of results and performance measurements for the case of linear system solving.

```
struct PerformanceMeasurement_ {
    int success;
    double Tsetup,Tsolve;
    double *ConvergenceHistory;
    double BackwardError,ForwardError;
}
struct Result_ {
    Vector ComputedSolution;
    PerformanceMeasurement performance;
}
```

Some component of the performance measurement, typically a timing measurement, will be used to judge optimality of solution methods.
Having defined problems and results, we define the space of methods as the mappings from problems to results.

$$\boxed{\mathbb{M} = \{\mathbb{A} \mapsto \mathbb{R}\}\text{: the space of methods (potentially) solving the class of numerical problems}}$$

We allow for methods to fail to compute the solution, for instance by the divergence of an iterative method, or exceeding a set runtime limit. We formalize this by introducing a timing function $T(A, M)$, denoting the time that it takes method M to solve the problem A, where in case of failure we set $T(A, M) = \infty$.
We will defer further discussion of the method space $\mathbb{M}$ for section 4.

3.2 Features and features extraction

As in natural in any Machine Learning context, we introduce the concept of problem features, which will be the basis for any recommendation strategy.

$$\boxed{\mathbb{F}\text{: the space of feature vectors of the numerical problems}}$$

and

$$\boxed{\Phi : \mathbb{A} \mapsto \mathbb{F}\text{: a function that extracts features of numerical problems}}$$

where $\Phi(A) = \bar{x} \in \mathbb{F}$ is the feature vector describing the numerical problem.
Feature space can be complicated in structure: elements of $\bar{x}$ can be real numbers (for instance matrix elements), positive real (norms), integer (quantities related to matrix size and sparsity pattern), or elements of a finite set of choices. They can even be array-valued in any of these types.
The NMD (Numerical MetaData) library Eijkhout & Fuentes (2007) provides an API for feature storage, including various utility functions.
We posit the existence of a parametrized function that can compute one feature at a time:

```
ComputeQuantity(Problem problem,
  char *feature,ReturnValue *result,TruthValue *success);
```

Here, the `ReturnValue` type is a union of all returnable types, and the `success` parameter indicates whether the quantity was actually computed. Computation can fail for any number of reasons: if sequential software is called in parallel, if properties such as matrix bandwidth are asked of an implicitly given operator, et cetera. Failure is not considered catastrophic, and the calling code should allow for this eventuality.
For a general and modular setup, we do not hardwire the existence of any module. Instead, we use a function

```
DeclareModule(char *feature,ReturnType type,
    void(*module)(Problem,char*,ReturnValue*,TruthValue*));
```

to add individual feature computations to a runtime system.
The AnaMod (Analysis Modules) library Eijkhout & Fuentes (2007) implements the above ideas, though in a slightly different form.

4. Numerical methods

In a simple-minded view, method space can be considered as a finite, unordered, collection of methods $\{M_1, \ldots, M_k\}$, and in some applications this may even be the most appropriate view. However, in the context of linear system solving a method is a more structured entity: each method consists at least of the choice of a preconditioner and the choice of an iterative scheme (QMR, GMRES, et cetera), both of which are independent of each other. Other possible components of a method are scaling of the system, and permutations for improved load balancing. Thus we arrive at a picture of a number of preprocessing steps that transform the original problem into another one with the same solution – or with a different solution that can easily be transformed into that of the original problem – followed by a solver algorithm that takes the transformed problem and yields its solution.
This section will formalize this further structure of the method space.

4.1 Formal definition

Above, we had defined $\mathbb{M}$ as the set of mappings $\mathbb{A} \mapsto \mathbb{R}$. We now split that as the preprocessors

$$\boxed{\mathbb{P} = \{\mathbb{A} \mapsto \mathbb{A}\}\text{: the set of all mappings from problems into problems}}$$

and the solvers[2]

[2] In the context of linear system solving, these will be Krylov methods, hence the choice of the letter 'K'.

$\mathbb{K} = \{\mathbb{A} \mapsto \mathbb{R}\}$: the set of all solvers

To illustrate the fact that the preprocessing stages are really mappings $\mathbb{A} \mapsto \mathbb{A}$, consider a right scaling D of a linear system, which maps the problem/solution tuple $\langle A, b, \bar{x}\rangle$ to $\langle AD, b, D^{-1}x\rangle$.

To model the fact that we have different kinds of preprocessors, we posit the existence of subsets

$$\mathbb{P}_i \subset \mathbb{P},$$

and we will assume that the identity mapping is contained in each. For instance, one $\mathbb{P}_i$ could be the set of scalings of a linear system:

$$\mathbb{P}_4 = \{'\text{none}', '\text{left}', '\text{right}', '\text{symmetric}'\}$$

Other possibilities are permutations of the linear system, or approximations of the coeffient matrix prior to forming the preconditioner.

Applying one preprocessor of each kind then gives us the definition of a method:

$$m \in \mathbb{M}\colon m = k \circ p_n \circ \cdots \circ p_1, \quad k \in \mathbb{K}, p_i \in \mathbb{P}_i \tag{2}$$

We leave open the possibility that certain preprocessors can be applied in any sequence (for instance scaling and permuting a system commute), while for others different orderings are allowed but not equivalent. Some preprocessors may need to be executed in a fixed location; for instance, the computation of a preconditioner will usually come last in the sequence of preprocessors.

Typically, a preprocessed problem has a different solution from the original problem, so each preprocessor has a backtransformation operation, to be applied to the preprocessed solution.

4.2 Implementation

The set of system preprocessors, like that of the analysis modules above, has a two level structure. First, there is the preprocessor type; for instance 'scaling'. Then there is the specific choice within the type; for instance ''left scaling'. Additionally, but not discussed here, there can be parameters associated with either the type or the specific choice; for instance, we can scale by a block diagonal, with the parameter indicating the size of the diagonal blocks.

We implement the sequence of preprocessors by a recursive routine:

```
PreprocessedSolving
    (char *method,Problem problem,Result *solution)
{
  ApplyPreprocessor(problem,&preprocessed_problem);
  if ( /* more preprocessors */ )
    PreprocessedSolving(next_method,
      preprocessed_problem,&preprocessed_solution);
  else
    Solve(final_method,
      preprocessed_problem,&preprocessed_solution);
  UnApplyPreprocessor(preprocessed_solution,solution);
}
```

The actual implementation is more complicated, but this pseudo-code conveys the essence.

We again adopt a modular approach where preprocessors are dynamically declared:

```
DeclarePreprocessor(char *type,char *choice,
    void(*preprocessor)(Problem,Problem*));
```

The SysPro (System Preprocessor) library provides a number of preprocessors, including the forward and backward transformation of the systems. It also includes a framework for looping over the various choices of a preprocessor type, for instance for an exhaustive test.

5. Method selection

Our method selection problem can be formalized as of constructing a function

$$\boxed{\Pi : \mathbb{A} \mapsto \mathbb{M}\text{: the problem classification function}}$$

that maps a given problem to the optimal method. Including feature extraction, we can also define

$$\boxed{\Pi : \mathbb{F} \mapsto \mathbb{M}\text{: the classification function in terms of features}}$$

We start with a brief discussion of precisely what is meant by 'optimal'. After that, we will refine the definition of Π to reflect the preprocessor/solver structure, and we will address the actual construction of Π.

5.1 Different classification criteria

The simplest (non-constructive) definition of the method selection function Π is:

$$\Pi(A) = M \qquad \equiv \qquad \forall_{M' \in \mathbb{M}} \colon T(A, M) \leq T(A, M') \tag{3}$$

Several variant definitions are possible. Often, we are already satisfied if we can construct a function that picks a working method. For that criterium, we define Π non-uniquely as

$$\Pi'(A) = M \qquad \text{where } M \text{ is any method such that} \qquad T(A, M) < \infty \tag{4}$$

Also, we usually do not insist on the absolutely fastest method: we can relax equation (3) to

$$\Pi(A) = M \qquad \equiv \qquad \forall_{M' \in \mathbb{M}} \colon T(A, M) \leq (1 - \epsilon) T(A, M') \tag{5}$$

which, for sufficient values of ϵ, also makes the definition non-unique. In both of the previous cases we do not bother to define Π as a multi-valued function, but implicitly interpret $\Pi(A) = M$ to mean 'M is one possible method satisfying the selection criterion'.
Formally, we define two classification types:

classification for reliability This is the problem equation (4) of finding any method that will solve the problem, that is, that will not break down, stall, or diverge.

classification for performance This is the problem equation (3) of finding the fastest method for a problem, possibly within a certain margin.

In a logical sense, the performance classification problem also solves the reliability problem. In practice, however, classifiers are not infallible, so there is a danger that the performance classifier will mispredict, not just by recommending a method that is slower than optimal, but also by possibly recommending a diverging method. Therefore, in practice a combination of these classifiers may be preferable.
We will now continue with discussing the practical construction of (the various guises of) the selection function Π.

5.2 Examples

Let us consider some adaptive systems, and the shape that $\mathbb{F}$, $\mathbb{M}$, and Π take in them.

5.2.1 Atlas

Atlas Whaley et al. (2001) is a system that determines the optimal implementation of Blas kernels such as matrix-matrix multiplication. One could say that the implementation chosen by Atlas is independent of the inputs[3] and only depends on the platform, which we will consider a constant in this discussion. Essentially, this means that $\mathbb{F}$ is an empty space. The number of dimensions of $\mathbb{M}$ is fairly low, consisting of algorithm parameters such unrolling, blocking, and software pipelining parameters.

In this case, Π is a constant function defined by

$$\Pi(f) \equiv \min_{M \in \mathbb{M}} T(A, M)$$

where A is a representative problem. This minimum value can be found by a sequence of line searches, as done in Atlas, or using other minimization techniques such a modified simplex method Yi et al. (2004).

5.2.2 Scalapack/LFC

The distributed dense linear algebra library Scalapack Choi et al. (1992) gives in its manual a formula for execution time as a function of problem size N, the number of processors N_p, and the block size N_b. This is an example of a two-dimensional feature space (N, N_p), and a one-dimensional method space: the choice of N_b. All dimensions range through positive integer values. The function involves architectural parameters (speed, latency, bandwidth) that can be fitted to observations.

Unfortunately, this story is too simple. The LFC software Roche & Dongarra (2002) takes into account the fact that certain values of N_p are disadvantageous, since they can only give grids with bad aspect ratios. A prime number value of N_p is a clear example, as this gives a degenerate grid. In such a case it is often better to ignore one of the available processors and use the remaining ones in a better shaped grid. This means that our method space becomes two-dimensional with the addition of the actually used number of processors. This has a complicated dependence on the number of available processors, and this dependence can very well only be determined by exhaustive trials of all possibilities.

5.2.3 Collective communication

Collective operations in MPI Otto et al. (1995) can be optimized by various means. In work by Vadhyar *et al.* Vadhiyar et al. (2000), the problem is characterized by the message size and the number of processors, which makes $\mathbb{F}$ have dimension 2. The degrees of freedom in $\mathbb{M}$ are the segment size in which messages will be subdivided, and the 'virtual topology' algorithm to be used.

Assume for now that the method components can be set independently. The segment size is then computed as $s = \Pi_s(m, p)$. If we have an *a priori* form for this function, for instance $\Pi_s(m, p) = \alpha + \beta m + \gamma p$, we can determine the parameters by a least squares fit to some observations.

Suppose the virtual topology depends only on the message size m. Since for the virtual topology there is only a finite number of choices, we only need to find the crossover points, which

[3] There are some very minor caveats for special cases, such as small or 'skinny' matrices.

can be done by bisection. If the topology depends on both m and p, we need to find the areas in (m, p) space, which can again be done by some form of bisection.

5.3 Database

In our application we need to be explicit about the database on which the classification is based. That issue is explored in this section.

5.3.1 General construction

The function Π is constructed from a database of performance results that results from solving a set of problems $\mathbf{A} \subset \mathbb{A}$ by each of a collection of methods $\mathbf{M} \subset \mathbb{M}$, each combination yielding a result $r \in \mathbb{R}$ (equation (1)). Thus we store features of the problem, an identifier of the method used, and the resulting performance measurement:

$$\boxed{\mathcal{D} \subset \mathbb{D} = \{\mathbb{F} \times \mathbb{M} \to \mathbb{T}\}\text{: the database of features and performance results of solved problems}}$$

We posit a mechanism (which differs per classifcation strategy) that constructs Π from a database $\mathcal{D}$. Where needed we will express the dependency of Π on the database explicitly as $\Pi_{\mathcal{D}}$. In the case of Bayesian classification the construction of Π from $\mathcal{D}$ takes a particularly elegant form, which we will discuss next.

5.3.2 Bayesian classification; method suitability

In methods like Bayesian classification, we take an approach to constructing Π where we characterize each method individually, and let Π be the function that picks the most suitable one.

Starting with the database $\mathcal{D}$ as defined above, We note for each problem – and thus for each feature vector – which method was the most successful:

$$\mathcal{D}'\colon \mathbb{F} \times \mathbb{M} \to \{0,1\} \quad \text{defined by} \quad \mathcal{D}'(f,m) = 1 \ \equiv \ m = \arg\min_m \mathcal{D}(f,m)$$

This allows us to draw up indicator functions for each method[4]:

$$\mathbb{B}'\colon \mathbb{M} \to \text{pow}(\mathbb{F}) \quad \text{defined by} \quad f \in \mathbb{B}'(m) \Leftrightarrow \mathbb{D}'(f,m) = 1 \tag{6}$$

These functions are generalized (multi-dimensional) histograms: for each method they plot the feature (vector) values of sample problems for which this method is was found to be optimal. However, since these functions are constructed from a set of experiments we have that, most likely, $\cup_{m\in\mathbb{M}}\mathbb{B}'(m) \subsetneq \mathbb{F}$. Therefore, it is not possible to define

$$\Pi(f) = m \quad \equiv \quad f \in \mathbb{B}'(m),$$

since for many values of f, the feature vector may not be in *any* $\mathbb{B}'(m)$. Conversely, it could also be in $\mathbb{B}'(m)$ for several values of m, so the definition is not well posed.

Instead, we use a more general mechanism. First we define suitability functions:[5]

$$\boxed{\mathbb{S} = \{\mathbb{F} \to [0,1]\}\text{: the space of suitability measurements of feature vectors}} \tag{7}$$

[4] There is actually no objection to having $\mathcal{D}(f,m)$ return 1 for more than one method m; this allows us to equivocate methods that are within a few percent of each other's performance. Formally we do this by extending and redefining the arg min function.

[5] Please ignore the fact that the symbol S already had a meaning, higher up in this story.

This is to be interpreted as follows. For each numerical method there will be one function $\sigma \in \mathbb{S}$, and $\sigma(f) = 0$ means that the method is entirely unsuitable for problems with feature vector f, while $\sigma(f) = 1$ means that the method is eminently suitable.
We formally associate suitability functions with numerical methods:

$$\boxed{\mathbb{B}\colon \mathbb{M} \to \mathbb{S}\text{: the method suitability function}} \tag{8}$$

and use the function $\mathbb{B}$ to define the selection function:

$$\Pi(f) = \arg\max_{m} \mathbb{B}(m)(f). \tag{9}$$

Since elements of $\mathbb{S}$ are defined on the whole space $\mathbb{F}$, this is a well-posed definition.
The remaining question is how to construct the suitability functions. For this we need constructor functions

$$\boxed{\mathbb{C}\colon P(\mathbb{F}) \to \mathbb{S}\text{: classifier constructor functions}} \tag{10}$$

The mechanism of these can be any of a number of standard statistical techniques, such as fitting a Gaussian distribution through the points in the subset $\cup_{f\in F} f$ where $F \in P(\mathbb{F})$.
Clearly, now $\mathbb{B} = \mathbb{C} \circ \mathbb{B}'$, and with the function $\mathbb{B}$ defined we can construct the selection function Π as in equation (9).

5.4 Implementation of Bayesian classification

Strictly speaking, the above is a sufficient description of the construction and use of the Π functions. However, as defined above, they use the entire feature vector of a problem, and in practice a limited set of features may suffice for any given decision. This is clear in such cases as when we want to impose "This method only works for symmetric problems", where clearly only a single feature is needed. Computing a full feature vector in this case would be wasteful. Therefore, we introduce the notation $\mathbb{F}_I$ where $I \subset \{1, \ldots, k\}$. This corresponds to the subspace of those vectors in $\mathbb{F}$ that are null in the dimensions not in I.
We will also assume that for each method $m \in \mathbb{M}$ there is a feature set I_m that suffices to evaluate the suitability function $\mathbb{B}(m)$. Often, such a feature set will be common for all methods in a set $\mathbb{P}_i$ of preprocessors. For instance, graph algorithms for fill-in reduction only need structural information on the matrix.
Here is an example of the API (as used in the Salsa system) that defines and uses feature sets. First we define a subset of features:

```
FeatureSet symmetry;
NewFeatureSet(&symmetry);
AddToFeatureSet(symmetry,
    "simple","norm-of-symm-part",&sidx);
AddToFeatureSet(symmetry,
    "simple","norm-of-asymm-part",&aidx);
```

After problem features have been computed, a suitability function for a specific method can then obtain the feature values and use them:

```
FeatureSet symmetry; // created above
FeatureValues values;
NewFeatureValues(&values);
InstantiateFeatureSet(problem,symmetry,values);
GetFeatureValue(values,sidx,&sn,&f1);
GetFeatureValue(values,aidx,&an,&f2);
if (f1 && f2 && an.r>1.e-12*sn.r)
   printf("problem too unsymmetric\n");
```

6. Classification of composite methods

In section 4 we showed how, in our application area of linear system solving, the space of methods has a structure where a method is a composite of a sequence of preprocessing steps and a concluding solver; equation equation (2). Accordingly, our recommendation function Π will be a composite:

$$\Pi = \langle \Pi_k, \{\Pi_i\}_{i \in \mathbb{P}} \rangle.$$

In the previous section, we posited a general mechanism (which we described in detail for methods like Bayesian classification) of deriving $\Pi_{\mathcal{D}}$ from a database $\mathcal{D} \subset \{\mathbb{F} \times \mathbb{M} \to \mathbb{T}\}$. In this section we will consider ways of defining databases $\mathcal{D}_{\mathbb{K}}, \mathcal{D}_{\mathbb{P}}$ and attendant functions $\Pi_{\mathbb{K}}, \Pi_{\mathbb{P}}$, and of combining these into an overall recommendation function.

For simplicity of exposition, we restrict our composites to a combination of one preprocessor (in practice, the preconditioner), and one solver (the iterative method); that is $\mathbb{M} = \mathbb{P} \times \mathbb{K}$.

At first we consider the performance problem, where we recommend a method that will minimize solution time (refer to section 5.1 for a definition of the two types of classification). Then, in section 6.4 we consider the reliability problem of recommending a method that will converge, no matter the solution time.

6.1 Combined recommendation

In this strategy, we ignore the fact that a method is a product of constituents, and we simply enumerate the elements of $\mathbb{M}$. Our function Π is then based on the database

$$\mathcal{D} = \{\langle f, \langle p, k \rangle, t \rangle \mid \exists_{a \in \mathcal{A}} \colon t = T(p, k, a)\}$$

and the recommendation function is a straightforward mapping $\Pi^{\text{combined}}(f) = \langle p, k \rangle$.

For Bayesian classification we get for each $\langle p, k \rangle \in \mathbb{M}$ the class

$$C_{p,k} = \{A \in \mathcal{A} \colon T(p, k, A) \text{ is minimal}\}$$

and corresponding function $\sigma_{p,k}$. We can then define

$$\Pi^{\text{combined}}(f) = \arg\max_{p,k} \sigma_{p,k}(f)$$

The main disadvantage to this approach is that, with a large number of methods to choose from, some of the classes can be rather small, leading to insufficient data for an accurate classification.

In an alternative derivation of this approach, we consider the $C_{p,k}$ to be classes of preprocessors, but conditional upon the choice of a solver. We then recommend p and k, not as a pair but sequential: we first find the k for which the best p can be found:

$$\Pi^{\text{conditional}} = \begin{cases} \text{let } k := \arg\max_k \max_p \sigma_{p,k}(f) \\ \text{return } \langle \arg\max_p \sigma_{p,k}(f), k \rangle \end{cases}$$

However, this is equivalent to the above combined approach.

6.2 Orthogonal recommendation

In this strategy we construct separate functions for recommending elements of $\mathbb{P}$ and $\mathbb{K}$, and we put together their results.

We define two derived databases that associate a solution time with a feature vector and a preprocessor or solver separately, even though strictly speaking both are needed to solve a problem and thereby produce a solution time. For solvers:

$$\mathcal{D}_{\mathbb{K}} = \left\{ \langle f, k, t \rangle \mid k \in \mathbb{K},\ \exists_{a \in \mathcal{A}} \colon f = \phi(a), t = \min_{p \in \mathbb{P}} T(p, k, a) \right\}$$

and for preprocessors:

$$\mathcal{D}_{\mathbb{P}} = \left\{ \langle f, p, t \rangle \mid p \in \mathbb{P},\ \exists_{a \in \mathcal{A}} \colon f = \phi(a), t = \min_{k \in \mathbb{K}} T(p, k, a) \right\}.$$

From these, we derive the functions $\Pi_{\mathbb{K}}, \Pi_{\mathbb{P}}$ and we define

$$\Pi^{\text{orthogonal}}(f) = \langle \Pi_{\mathbb{P}}(f), \Pi_{\mathbb{K}}(f) \rangle$$

In Bayesian classification, the classes here are

$$C_k = \{A \colon \min_p T(p, k, A) \text{ is minimal over all } k\}$$

and

$$C_p = \{A \colon \min_k T(p, k, A) \text{ is minimal over all } p\},$$

giving functions σ_p, σ_k. (Instead of classifying by minimum over the other method component, we could also use the average value.) The recommendation function is then

$$\Pi^{\text{orthogonal}}(f) = \langle \arg\max_p \sigma_p(f), \arg\max_k \sigma_k(f) \rangle$$

6.3 Sequential recommendation

In the sequential strategy, we first recommend an element of $\mathbb{P}$, use that to transform the system, and recommend an element of $\mathbb{K}$ based on the transformed features.

Formally, we derive $\Pi_{\mathbb{P}}$ as above from the derived database

$$\mathcal{D}_{\mathbb{P}} = \left\{ \langle f, p, t \rangle \mid p \in \mathbb{P},\ \exists_{a \in \mathcal{A}} \colon f = \phi(a), t = \min_{k \in \mathbb{K}} T(p, k, a) \right\}.$$

but $\Pi_{\mathbb{K}}$ comes from the database of all preprocessed problems:

$$\begin{aligned} &\mathcal{D}_{\mathbb{K}} = \cup_{p \in \mathbb{P}} \mathcal{D}_{\mathbb{K},p}, \\ &\mathcal{D}_{\mathbb{K},p} = \{ \langle f, k, t \rangle \mid k \in \mathbb{K},\ \exists_{a \in \mathcal{A}} \colon f = \phi(p(a)), t = T(p, k, a) \} \end{aligned}$$

which gives us a single function $\Pi_{\mathbb{P}}$ and individual functions $\Pi_{\mathbb{K},p}$. This gives us

$$\Pi^{\text{sequential}}(f) = \langle \text{let } p := \Pi_{\mathbb{P}}(f), k := \Pi_{\mathbb{K}}(p(f)) \text{ or } \Pi_{\mathbb{K},p}(p(f))\rangle$$

For Bayesian classification, we define the classes C_p as above:

$$C_p = \{A\colon \min_k T(p,k,A) \text{ is minimal over all } p\},$$

but we have to express that C_k contains preconditioned features:

$$C_k = \{A \in \bigcup_p p(\mathcal{A})\colon T(p,k,A) \text{ is minimal, where } p \text{ is such that } A \in p(\mathcal{A})\}$$

Now we can define

$$\Pi^{\text{sequential}}(f) = \langle \text{let } p := \arg\max_p \sigma_p(f), k := \arg\max_k \sigma_k(p(f))\rangle$$

This approach to classification is potentially the most accurate, since both the preconditioner and iterator recommendation are made based on the features of the actual problem they apply to. This also means that this approach is the most expensive; both the combined and the orthogonal approach require only the features of the original problem. In practice, with a larger number of preprocessors, one can combine these approaches. For instance, if a preprocessor such as scaling can be classified based on some easy to compute features, it can be tackled sequentially, while the preconditioner and iterator are then recommended with the combined approach based on a full feature computation of the scaled problem.

6.4 The reliability problem

In the reliability problem we classify problems by whether a method converges on them or not. The above approaches can not be used directly in this case, for several reasons.

- The above approaches are based on assigning each problem to a single classes based on minimum solution time. In the reliability problem each problem would be assigned to multiple classes, since typically more than one method would converge on the problem. The resulting overlapping classes would lead to a low quality of recommendation.
- The sequential and orthogonal approaches would run into the additional problem that, given a preconditioner, there is usually at least one iterative method that gives a converging combination. Separate recommendation of the preconditioner is therefore impossible.

Instead, we take a slightly different approach. For each method m we define a function $\Pi^{(m)} : \mathbb{F} \mapsto \{0,1\}$ which states whether the method will converge given a feature vector of a problem. We can then define

$$\Pi(f) = \text{a random element of } \{m\colon \Pi^{(m)}(f) = 1\}$$

For Bayesian classification, we can adopt the following strategy. For each $M \in \mathbb{M}$, define the set of problems on which it converges:

$$C_M = \{A\colon T(M,A) < \infty\}$$

and let $\bar{C}_M$ be its complement:

$$\bar{C}_M = \{A\colon T(M,A) = \infty\}.$$

Now we construct functions $\sigma_M, \bar{\sigma}_M$ based on both these sets. This gives a recommendation function:

$$\Pi(f) = \{M\colon \sigma_M(f) > \bar{\sigma}_M(f)\}$$

This function is multi-valued, so we can either pick an arbitrary element from $\Pi(f)$, or the element for which the excess $\sigma_M(f) - \bar{\sigma}_M(f)$ is maximized.

The above strategies give only a fairly weak recommendation from the point of optimizing solve time. Rather than using reliability classification on its own, we can use it as a preliminary step before the performance classification.

7. Experiments

In this section we will report on the use of the techniques developed above, applied to the problem of recommending a preconditioner and iterative method for solving a linear system. The discussion on the experimental setup and results will be brief; results with much greater detail can be found in Fuentes (2007).

We start by introducing some further concepts that facilitate the numerical tests.

7.1 Experimental setup

We use a combination of released software from the Salsa project Salsa Project (n.d.a;n) and custom scripts. For feature computation we use AnaMod Eijkhout & Fuentes (2007); The Salsa Project (n.d.); storage and analysis of features and timings is done with MySQL and custom scripting in Matlab and its statistical toolbox.

The AnaMod package can compute 45 features, in various categories, such as structural features, norm-like features, measures of the spectrum and of the departure from normality. The latter two are obviously approximated rather than computed exactly.

The Salsa testbed gives us access to the iterative methods and preconditioners of the Petsc package. including the preconditioners of externally interfaced packages such as Hypre Falgout et al. (2006); Lawrence Livermore Lab, CASC group (n.d.).

7.2 Practical issues

The ideas developed in the previous sections are sufficient in principle for setting up a practical application of machine learning to numerical method selection. However, in practice we need some auxiliary mechanisms to deal with various ramifications of the fact that our set of test problems is not of infinite size. Thus, we need

- A way of dealing with features that can be invariant or (close to) dependent in the test problem collection.
- A way of dealing with methods that can be very close in their behaviour.
- An evaluation of the accuracy of the classifiers we develop.

7.2.1 Feature analysis

There are various transformations we apply to problem features before using them in various learning methods.

Scaling Certain transformations on a test problem can affect the problem features, without affecting the behaviour of methods, or being of relevance for the method choice. For instance, scaling a linear system by a scalar factor does not influence the convergence behaviour of iterative solvers. Also, features can differ in magnitude by order of magnitude. For this reason, we normalize features, for instance scaling them by the largest diagonal element. We also mean-center features for classification methods that require this.

Elimination Depending on the collection of test problems, a feature may be invariant, or dependent on other features. We apply Principal Component Analysis Jackson (2003) to the set of features, and use that to weed out irrelevant features.

7.2.2 Hierarchical classification

It is quite conceivable that certain algorithms are very close in behaviour. It then makes sense to group these methods together and first construct a classifier that can recommend first such a group, and subsequently a member of the group. This has the advantage that the classifiers are build from a larger number of observations, giving a higher reliability.

The algorithm classes are built by computing the independence of methods. For two algorithms x and y, the 'independence of method x from method y' is defined as

$$I_y(x) = \frac{\#\text{cases where } x \text{ works and } y \text{ not}}{\#\text{cases where } x \text{ works}}$$

The quantity $I_y(x) \in [0,1]$ describes how much x succeeds on different problems from y. Note that $I_y(x) \neq I_x(y)$ in general; if x works for every problem where y works (but not the other way around), then $I_y(x) = 0$, and $I_x(y) > 0$.

7.2.3 Evaluation

In order to evaluate a classifier, we use the concept of accuracy. The accuracy α of a classifier is defined as

$$\alpha = \frac{\#\text{problems correctly classified}}{\text{total }\#\text{problems}}$$

A further level of information can be obtained looking at the details of misclassification: a 'confusion matrix' is defined as $A = (\alpha_{ij})$ where α_{ij} is the ratio of problems belonging in class i, classified in class j, to those belonging in class i. With this, α_{ii} is the accuracy of classifier i, so, for an ideal classifier, A is a diagonal matrix with a diagonal $\equiv 1$; imperfect classifiers have more weight off the diagonal.

A further measure of experimental results is the confidence interval z, which indicates an interval in which the resulting accuracy for a random trial will be away for the presented average accuracy α by $\pm z$ Douglas & Montgomery (1999) of the time. We use z to delimit the confidence interval since we have used the *Z-test* Douglas & Montgomery (1999), commonly used in statistics. The confidence interval is a measure of how 'stable' the resulting accuracy is for an experiment.

7.3 Numerical test

We tested a number of iterative methods and preconditioners on a body of test matrices, collected from Matrix Market and a few test applications. The iterative methods and preconditioners are from the PETSc library.

As described above, we introduced superclasses, as follows:

- **B**={ *bcgs, bcgsl, bicg* }, where *bcgs* is BiCGstab van der Vorst (1992), and *bcgsl* is BiCGstab(ℓ) Sleijpen et al. (n.d.) with $\ell \geq 2$.
- **G**={ *gmres, fgmres* } where *fgmres* is the 'flexible' variant of GMRES Saad (1993).
- **T**={ *tfqmr* }
- **C**={ *cgne* }, conjugate gradients on the noral equations.

for iterative methods and

- **A** = { *asm, rasm, bjacobi* }, where *asm* is the Additive Schwarz method, and *rasm* is its restricted variant Cai & Sarkis (1999); *bjacobi* is block-jacobi with a local ILU solve.
- **BP** = { *boomeramg, parasails, pilut* }; these are three preconditioners from the *hypre* package Falgout et al. (2006); Lawrence Livermore Lab, CASC group (n.d.)
- **I** = { *ilu, silu* }, where *silu* is an ILU preconditioner with shift Manteuffel (1980).

for preconditioners.

(a) Iterative Methods

ksp	$\alpha \pm z$
bcgsl	0.59±0.02
bcgs	0.71±0.03
bicg	0.68±0.06
fgmres	0.80±0.02
gmres	0.59±0.04
lgmres	0.81±0.03
tfqmr	0.61±0.05

(b) Preconditioners

pc	$\alpha \pm z$
asm	0.72±0.05
bjacobi	0.11±0.11
boomeramg	0.71±0.06
ilu	0.66±0.02
parasails	0.46±0.12
pilut	0.80±0.06
rasm	0.70±0.04
silu	0.83±0.02

Table 1. Accuracy of classification using one class per available method

(a) Iterative methods

Super	Class	α	Compound
B		0.95	
	bcgs	0.93	0.87
	bcgsl	0.92	0.87
	bicg	0.89	0.84
G		0.98	
	fgmres	0.96	0.94
	gmres	0.91	0.89
	lgmres	0.94	0.93
T		0.91	
	tfqmr	–	0.91

(b) Preconditioners

Super	Class	α	Compound
A		0.95	
	asm	0.98	0.93
	bjacobi	0.67	0.64
	rasm	0.82	0.78
BP		0.99	
	boomeramg	0.80	0.80
	parasails	0.78	0.77
	pilut	0.97	0.96
I		0.94	
	ilu	0.82	0.75
	silu	0.97	0.91

Table 2. Hierarchical classification results

strategy	average	std.dev.
combined	.3	.15
orthogonal	.78	.04

Table 3. Average and standard deviation of the correct classification rate

In table 1 we report the accuracy (as defined above) for a classification of all individual methods, while table 2 gives the result using superclasses. Clearly, classification using superclasses is superior. All classifiers were based on decision trees Breiman et al. (1983); Dunham (2002).

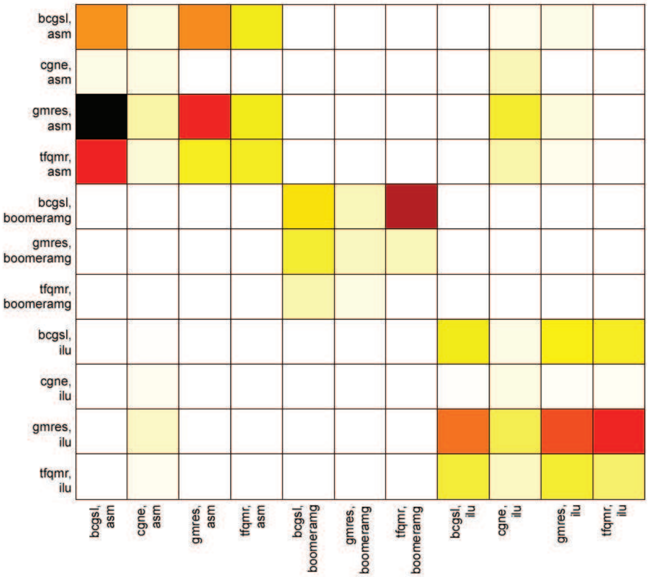


Fig. 1. Confusion matrix for combined approach for classifying (pc, ksp).

Finally, in figures 1, 2 we give confusion matrices for two different classification strategies for the preconditioner / iterative method combination. The orthogonal approach gives superior results, as evinced by the lesser weight off the diagonal. For this approach, there are fewer classes to build classifiers for, so the modeling is more accurate. As a quantative measure of the confusion matrices, we report in table 3 the average and standard deviation of the fraction of correctly classified matrices.

7.4 Reliability and Performance

Above (section 5.1) we defined the Performance recommendation problem of finding the fastest method, and the Reliability recommendation problem of finding one that works at all. Since orthogonal recommendation is not possible for the reliability problem (section 6.4), we use combined recommendation there. In our tests, this strategy turns out to recommend a subset of the actually converging methods, so it is indeed a valuable preprocessing step.

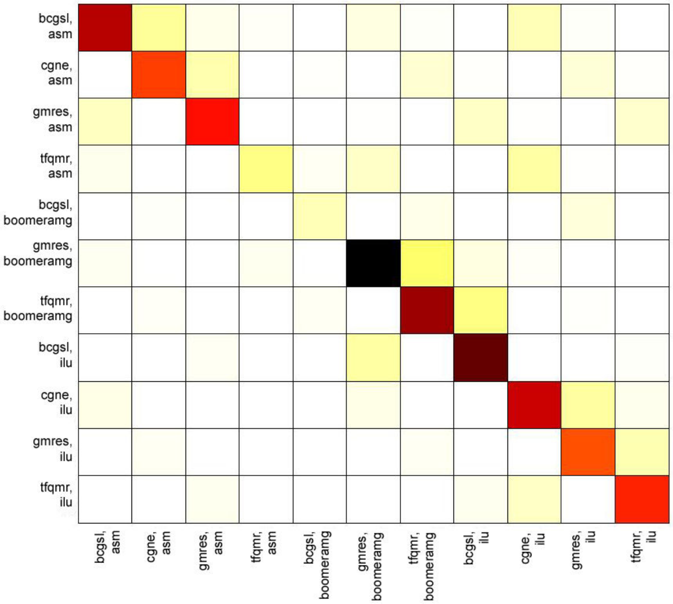


Fig. 2. Confusion matrix for orthogonal approach for classifying (pc, ksp).

8. Conclusion

We have defined the relevant concepts for the use of machine learning for algorithm selection in various areas of numerical analysis, in particular iterative linear system solving. An innovative aspect of our approach is the multi-leveled approach to the set of objects (the algorithms) to be classified. An important example of levels is the distinction between the iterative process and the preconditioner in iterative linear system solvers. We have defined various strategies for classifying subsequent levels. A numerical test testifies to the feasibility of using machine learning to begin with, as well as the necessity for our multi-leveled approach.

9. References

Arnold, D., Blackford, S., Dongarra, J., Eijkhout, V. & Xu, T. (2000). Seamless access to adaptive solver algorithms, *in* M. Bubak, J. Moscinski & M. Noga (eds), *SGI Users' Conference*, Academic Computer Center CYFRONET, pp. 23–30.

Balay, S., Gropp, W. D., McInnes, L. C. & Smith, B. F. (1999). PETSc home page. http://www.mcs.anl.gov/petsc.

Bhowmick, S., Eijkhout, V., Freund, Y., Fuentes, E. & Keyes, D. (2006). Application of machine learning to the selection of sparse linear solvers, *Int. J. High Perf. Comput. Appl.* . submitted.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1983). *CART: Classification and Regression Trees*.

Cai, X.-C. & Sarkis, M. (1999). A restricted additive Schwarz preconditioner for general sparse linear systems, *SIAM J. Sci. Comput.* **21**: 792–797.

Choi, Y., Dongarra, J. J., Pozo, R. & Walker, D. W. (1992). Scalapack: a scalable linear algebra library for distributed memory concurrent computers, *Proceedings of the fourth symposium on the frontiers of massively parallel computation (Frontiers '92), McLean, Virginia, Oct 19–21, 1992*, pp. 120–127.

Czyzyk, J., Mesnier, M. & More, J. (1996). The NetworkEnabled optimization system (neos) server, *Technical Report MCS-P615-0996*, Argonne National Laboratory, Argonne, IL.

Czyzyk, J., Mesnier, M. & Moré, J. (1998). The NEOS server, *IEEE J. Comp. Sci. Engineering* **5**: 68–75.

Dongarra, J., Bosilca, G., Chen, Z., Eijkhout, V., Fagg, G. E., Fuentes, E., Langou, J., Luszczek, P., Pjesivac-Grbovic, J., Seymour, K., You, H. & Vadiyar, S. S. (2006). Self adapting numerical software (SANS) effort, *IBM J. of R.& D.* **50**: 223–238. `http://www.research.ibm.com/journal/rd50-23.html`, also UT-CS-05-554 University of Tennessee, Computer Science Department.

Douglas, C. & Montgomery, G. (1999). *Applied Statistics and Probability for Engineers*.

Dunham, M. (2002). *Data Mining: Introductory and Advanced Topics*, Prentice Hall PTR Upper Saddle River, NJ, USA.

Eijkhout, V. & Fuentes, E. (2007). A standard and software for numerical metadata, *Technical Report TR-07-01*, Texas Advanced Computing Center, The University of Texas at Austin. to appear in ACM TOMS.

Falgout, R., Jones, J. & Yang, U. (2006). The design and implementation of hypre, a library of parallel high performance preconditioners, *Numerical Solution of Partial Differential Equations on Parallel Computers, A.M. Bruaset and A. Tveito, eds.*, Vol. 51, Springer-Verlag, pp. 267–294. UCRL-JRNL-205459.

Fuentes, E. (2007). *Statistical and Machine Learning Techniques Applied to Algorithm Selection for Solving Sparse Linear Systems*, PhD thesis, University of Tennessee, Knoxville TN, USA.

Greenbaum, A. & Strakos, Z. (1996). Any nonincreasing convergence curve is possible for GMRES, *SIMAT* **17**: 465–469.

Gropp, W. D. & Smith, B. F. (n.d.). Scalable, extensible, and portable numerical libraries, *Proceedings of the Scalable Parallel Libraries Conference, IEEE 1994*, pp. 87–93.

Houstis, E., Verykios, V., Catlin, A., Ramakrishnan, N. & Rice, J. (2000). PYTHIA II: A knowledge/database system for testing and recommending scientific software.

Jackson, J. (2003). *A User's Guide to Principal Components*, Wiley-IEEE.

Lawrence Livermore Lab, CASC group (n.d.). Scalable Linear Solvers. `http://www.llnl.gov/CASC/linear_solvers/`.

Manteuffel, T. (1980). An incomplete factorization technique for positive definite linear systems, *Math. Comp.* **34**: 473–497.

Otto, S., Dongarra, J., Hess-Lederman, S., Snir, M. & Walker, D. (1995). *Message Passing Interface: The Complete Reference*, The MIT Press.

Ramakrishnan, N. & Ribbens, C. J. (2000). Mining and visualizing recommendation spaces for elliptic PDEs with continuous attributes, *ACM Trans. Math. Software* **26**: 254–273.

Roche, K. J. & Dongarra, J. J. (2002). Deploying parallel numerical library routines to cluster computing in a self adapting fashion. Submitted.

Saad, Y. (1993). A flexible inner-outer preconditioned GMRES algorithm, *SIAM J. Sci. Stat. Comput.* **14**: 461–469.

Salsa Project (n.d.a). SALSA: Self-Adapting Large-scale Solver Architecture. `http://icl.cs.utk.edu/salsa/`.

Hierarchical Reinforcement Learning Using a Modular Fuzzy Model for Multi-Agent Problem

Toshihiko Watanabe
Osaka Electro-Communication University
Japan

1. Introduction

Reinforcement learning (Sutton & Barto, 1998; Watkins & Dayan, 1998; Grefenstette, 1988; Miyazaki et al., 1999; Miyazaki et al., 1999) among machine learning techniques is an indispensable approach to realize the intelligent agent such as autonomous mobile robots. The importance of the technique is discussed in several literatures. However there exist a lot of problems compared with the other learning techniques such as Neural Networks in order to apply reinforcement learning to actual applications. One of the main problems of reinforcement learning application of actual sized problem is "curse of dimensionality" problem in partition of multi-inputs sensory states. High dimension of input leads to huge number of rules in the reinforcement learning application. It should be avoided maintaining computational efficiency for actual applications. Multi-agent problem such as the pursuit problem (Benda et al., 1985; Ito & Kanabuchi, 2001) is typical difficult problem for reinforcement learning computation in terms of huge dimensionality. As the other related problem, learning of complex task is not easy essentially because the reinforcement learning is based only upon rewards derived from the environment.

In order to deal with these problems, several effective approaches are studied. For relaxation of task complexity, several types of hierarchical reinforcement learning have been proposed to apply actual applications (Takahashi & Asada, 1999; Morimoto & Doya, 2000). To avoid the curse of dimensionality, there exists modular hierarchical learning (Ono & Fukumoto, 1996; Fujita & Matsuno, 2005) that construct the learning model as the combination of subspaces. Adaptive segmentation (Murano & Kitamura, 1997; Hamagami et al.,2003) for constructing the learning model validly corresponding to the environment is also studied. However more effective technique of different approach is also necessary in order to apply reinforcement learning to actual sized problems.

In this chapter, I focus on the well-known pursuit problem and propose a hierarchical modular reinforcement learning that Profit Sharing learning algorithm is combined with Q Learning reinforcement learning algorithm hierarchically in multi-agent environment. As the model structure for such huge problem, I propose a modular fuzzy model extending SIRMs architecture (Seki et al., 2006; Yubazaki et al., 1997). Through numerical experiments, I show the effectiveness of the proposed algorithm compared with the conventional algorithms.

The chapter is organized as follows. In section 2, an overview of pursuit problem as multi-agent environment is presented. In section 3, I propose construction of agent model and essential learning algorithms of a hierarchical reinforcement learning using a modular model architecture. In section 4, I propose a modular fuzzy model for agent model construction. The results of numerical experiments are shown in section 5. Finally, conclusions are drawn in section 6.

2. Pursuit problem as multi-agent environment

The pursuit problem is well known and has been studied as typical benchmark problem in Distributed Artificial Intelligence research field (Benda et al., 1985). It is multi-agent based problem that hunter agents act collaboratively to capture prey agent. Figure 1 shows the 4-agent pursuit problem in 7×7 grids field. In the problem, all agent behave in turn to move upward, downward, rightward, leftward in one gird, or to stay. Collision of the agents is prohibited because one grid allows only one agent to stay. The objective of the simulation is to surround the prey agent by the hunter agents as shown in Fig.2.

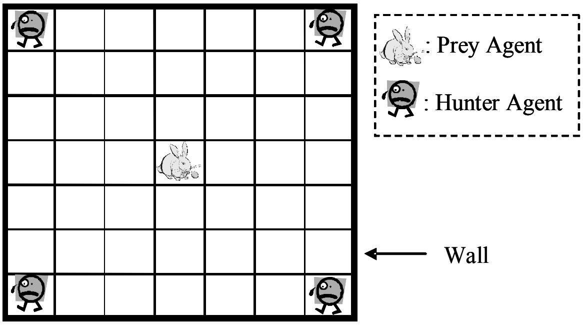


Fig. 1. 4-Pursuit Problem(7x7 grids)

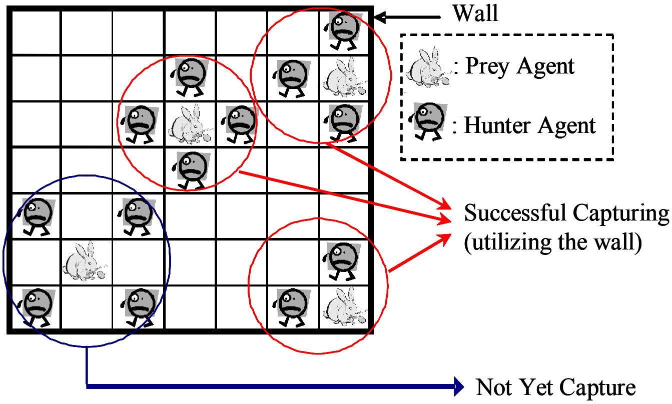


Fig. 2. Examples of Capturing Condition in Pursuit Problem

The hunter agents can utilize walls for surrounding as well as surrounding by whole hunter agents. When the surrounding is successfully performed, related hunter agents receive reward from the environment to carry out reinforcement learning. As for behavior of the prey agent, it behaves to run away from the nearest hunter agent for playing a fugitive role. For actual computer simulations or mobile robot applications, it is indispensable to avoid huge memory consumption for the state space, i.e. "curse of dimensionality", and to improve slow learning speed caused by its sparsity(e.g. acquired Q-value through reinforcement learning). In this study, I focus on the 4-agent pursuit problem to improve precision and efficiency of reinforcement learning in multi-agent environment and to demonstrate settlement of "curse of dimensionality".

For simulation study, I adopt "soft-max" strategy for selecting the action of the hunter agents. The conditional probability based on Boltzmman distribution for action selection is as follows:

$$p(a \mid s) = \frac{\exp\left(w(s,a)/T_t\right)}{\sum_{d \in N} \exp\left(w(s,d)/T_t\right)} \quad , \; T_{t+1} = T_t \times \beta \tag{1}$$

where T_t is temperature at t-th iteration, s is state vector, a is the action of the agent, β is the parameter for temperature cooling($0<\beta<1$), w denotes evaluation value for state-and-action pair, and N denotes the set of all alternative action at the state s. Owing to this mechanism, the hunter agent act like random walk(exploring) with high temperature value in the early simulation trials and act definitely based on acquired evaluation values in the later simulation trials according to the lowered temperature value.

3. A hierarchical reinforcement learning using modular model architecture

3.1 Basic concepts

There exist two problems to solve the pursuit problem efficiently. One is huge memory consumption for internal knowledge expression of the agents expressed as evaluation weights corresponding to the pair of state-and-action caused by the grid size of the environment and the number of hunter agents. In order to restrain the increase of required memory for the agents, modular structure is applied for expression of the agent knowledge base. The other is complex objective, i.e. surrounding the prey *collaboratively*. In general, it is effective for dealing with such complex task to decompose into sub-tasks. Then I decompose the task into hierarchical sub-tasks to fulfill reinforcement learning effectively. I propose a hierarchical modular reinforcement learning to solve the above described two problems in the multi-agent pursuit simulation.

3.2 Hierarchical task decomposition for agent learning

It is difficult to decide how many kinds of subtask should be decomposed into. In this study, I empirically decompose the surrounding task(capturing) into "decision of move position target" for surrounding according to current monitored state and "selection of appropriate action" to move to the target position of each agent. The latter task is native, isolated from the other hunter agents, and is not needed to be collaborative such as position control of the single agent. In other words, the task is decomposed into "surrounding" task synchronized with the other hunter agents and "exploring the environment" task. Moreover, the upper task corresponds only to collaborative surrounding strategy. Figure 3 shows the internal hierarchical structure of the hunter agent. The knowledge base of the agent is composed of the "Rules in Upper Layer" and the "Rules in Lower Layer" as shown in the figure. It is important to keep learning capability as well as task decomposition. According to the two-layered decomposition, rules in the lower layer can be adapted corresponding to the agent behavior in every step as Markov Decision Process, as shown in Fig.4.

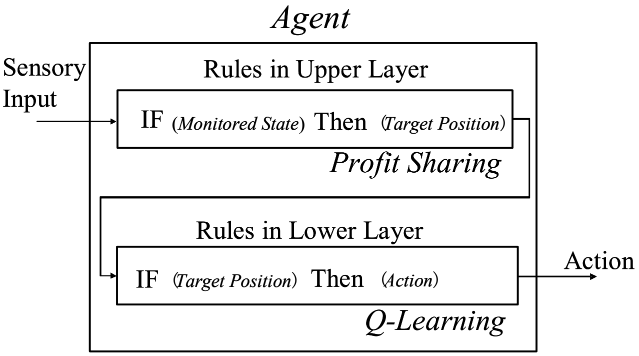


Fig. 3. Internal Hierarchical Structure of Hunter Agent

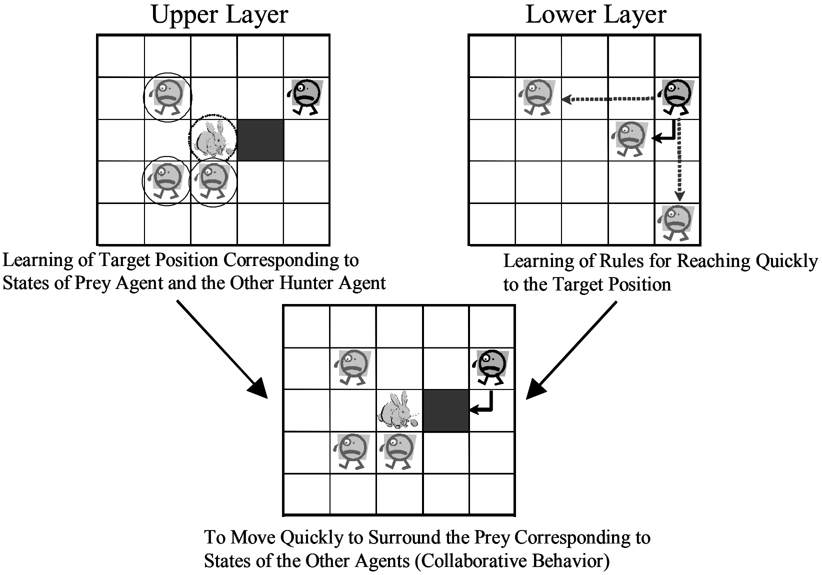

Upper Layer
Lower Layer
Learning of Target Position Corresponding to States of Prey Agent and the Other Hunter Agent
Learning of Rules for Reaching Quickly to the Target Position
To Move Quickly to Surround the Prey Corresponding to States of the Other Agents (Collaborative Behavior)

Fig. 4. Conceptual Diagram of Hierarchical Task Decomposition

3.3 A modular profit sharing learning for upper layer

In the upper layer, the target position of the agent is decided based on observed state such as the current position of the prey agent and the other hunter agents. The rules in the upper layer express goodness of the target position corresponding to the current state excluding actual actions. In order to construct the rules based on the current state combination, huge corresponding memory is needed. To avoid such requirement, the authors applied modular structure for the rule expression (Takahashi & Watanabe, 2006) in the upper layer as shown in Fig.5. In this section, the dimension of modular model is assumed to be three for explanation simplicity. Higher dimension can also be considered as the same manner. Original state space of each agent is expressed as the modular model by covering with three subspaces of oneself-and-another pair as shown in Fig.6.

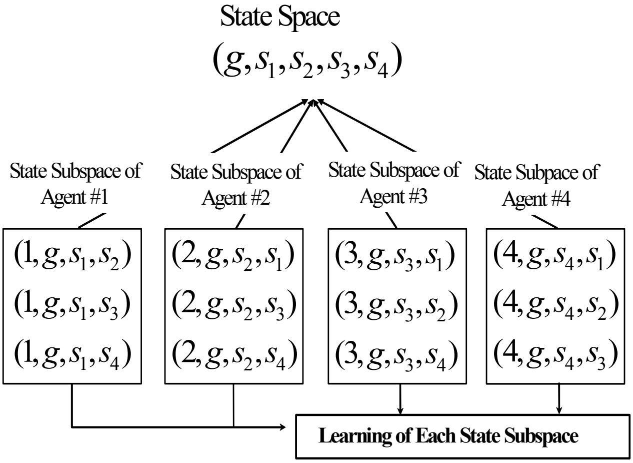


Fig. 5. Modular Structure of Agent State Maps

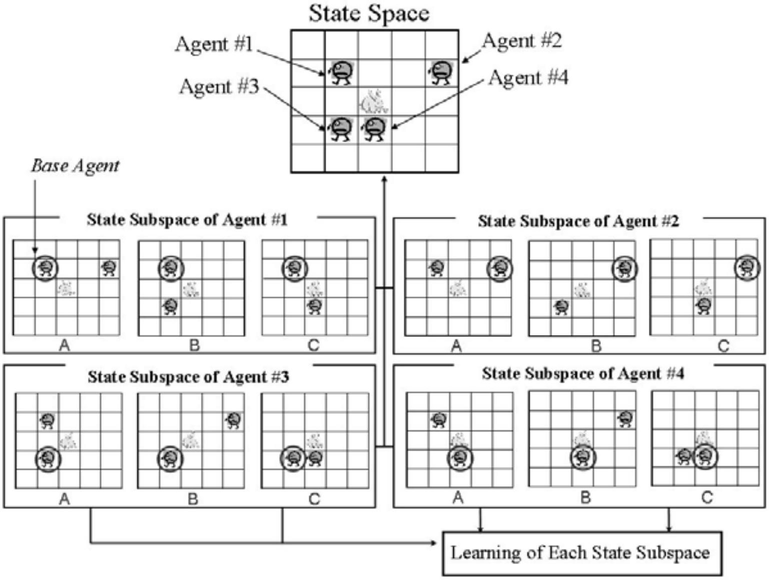

State Space
Agent #1
Agent #2
Agent #3
Agent #4
Base Agent
State Subspace of Agent #1
State Subspace of Agent #2
State Subspace of Agent #3
State Subspace of Agent #4
A
B
C
Learning of Each State Subspace

Fig. 6. An Example of Modular Structured Maps

The weights of rules in the upper layer are updated by Profit Sharing learning algorithm(Miyazaki et al., 1999), when capturing succeeds, as the following formulations:

$$
\begin{aligned}
&u(e,g_i,h_{e,i},h_{\varepsilon,i}) = u(e,g_i,h_{e,i},h_{\varepsilon,i}) + k(e,g_i,h_{e,i},h_{\varepsilon,i}) \\
&k(e,g_{i+1},h_{e,i+1},h_{\varepsilon,i+1}) = \frac{1}{\rho} k(e,g_i,h_{e,i},h_{\varepsilon,i}) \qquad (i = 0,1,\ldots,m-1,\ \varepsilon \neq e)
\end{aligned}
\tag{2}
$$

where u is the weight of the rule, g is state of the prey agent, $h_{e,i}$ denotes the state of agent e at i step ago from the current step, k denotes the reinforcement function, and ρ is the parameter.

In the action phase, the target position is desirable to be decided as a sub-goal for surrounding task instead of final goal corresponding to the current state of the prey agent according to the rule weights. In this study, the target position of the agent is generated as:

$$
p = \arg\max_{v} \sum\nolimits_q \frac{u\left(e,g,v,h_q\right)}{\mu^{\|h_e - v\|}} \qquad (q \neq e, \mu \geq 1) \tag{3}
$$

where h_e denotes the current position of the agent, v denotes candidate of the target position, q denotes the other agent, and μ is the parameter. Due to these state selections, the target position as valid sub-goal is generated and sent to the lower layer.

3.4 Q-learning for lower layer

In the lower layer, appropriate selection of concrete action to reach the target position decided at the upper layer should be fulfilled through reinforcement learning process. It should be noted that states of the other hunter agents are unnecessary for the lower task. The input state of the rule consists of the target position and the current own position. At every step in learning trial, the learning of the lower layer is employed because we can interpret every agent movement as the movement to current position considered as the movement to virtual targeted position according to another viewpoint. In the lower layer, Q-Learning (Sutton & Barto, 1998; Watkins & Dayan, 1988) can be applied successfully because the process is typical Markov Decision Process. Q-Learning is realized as:

$$
Q\left(s_{e,t}, a_{e,t}, c\right) = Q\left(s_{e,t}, a_{e,t}, c\right) + \alpha\left(r_t + \gamma \max_{\eta} Q\left(s_{e,t}, \eta, c\right) - Q\left(s_{e,t}, a_{e,t}, c\right)\right) \tag{4}
$$

where Q is Q-value, $s_{e,t}$ is the state vector of the agent e at t-th step, $a_{e,t}$ is action of the agent e at t-th step, c denotes the state for updating, r denotes the reward, and a, γ are parameters. It should be noted that the current state of the agent moved from the other position always receive rewards considered as the virtual targeted state, internally.

4. A modular fuzzy model

4.1 Model structure

As a fuzzy model having high applicability, Single Input Rule Modules(SIRMs) (Seki et al., 2006; Yubazaki et al., 1997) was proposed. The idea is to unify reasoning outputs from fuzzy rule modules comprised with single input formed fuzzy if-then rules. The number of rules can be drastically reduced as well as bringing us high maintainability in actual application. However, its disadvantage of low precision is inevitable in order to apply the method to huge multi-dimensional problems. I extend the SIRMs method by relaxing the restriction of the input space, i.e. single, to arbitrary subspace of the rule.

I propose a "Modular Fuzzy Model", for constructing the model of huge multi-dimensional space. Description of the model is as follows:

$$\begin{aligned} &Rules-1:\{if\ P_1(x)\, is\ A_j^1\ then\ y_1 = f_j^1(P_1(x))\}_{j=1}^{m_1} \\ &\qquad\vdots \\ &Rules-i:\{if\ P_i(x)\, is\ A_j^i\ then\ y_i = f_j^i(P_i(x))\}_{j=1}^{m_i} \\ &\qquad\vdots \\ &Rules-n:\{if\ P_n(x)\, is\ A_j^n\ then\ y_n = f_j^n(P_n(x))\}_{j=1}^{m_n} \end{aligned} \tag{5}$$

where "*Rules-i*" stands for the *i*-th fuzzy rule module, $P_i(x)$ denotes predetermined projection of the input vector x in *i*-th module, y_i is the output variable, and n is the number of rule modules. The number of constituent rules in the *i*-th fuzzy rule module is m_i. f is the function of consequent part of the rule like TSK-fuzzy model (Takagi & Sugeno, 1985). A_j^i denotes the fuzzy sets defined in the projected space.

The membership degree of the antecedent part of *j*-th rule in "*Rules-i*" module is calculated as:

$$h_j^i = A_j^i(P_i(x^0)) \tag{6}$$

where h denotes the membership degree and x^0 is an input vector. The output of fuzzy reasoning of each module is decided as the following equation.

$$y_i^0 = \frac{\sum_{k=1}^{m_i} h_k^i \cdot f_k^i(P_i(x^0))}{\sum_{k=1}^{m_i} h_k^i} \tag{7}$$

The final output of the "Modular Fuzzy Model" is formulated as:

$$y^0 = \sum_{i=1}^{n} w_i \cdot y_i^0 \tag{8}$$

where w_i denotes the parameter of importance of the *i*-th rule module. The parameter can be also formulated as the output of rule based system like modular neural network structure (Auda & Kamel, 1999). Figure 7 shows the structure of Modular Fuzzy Model.

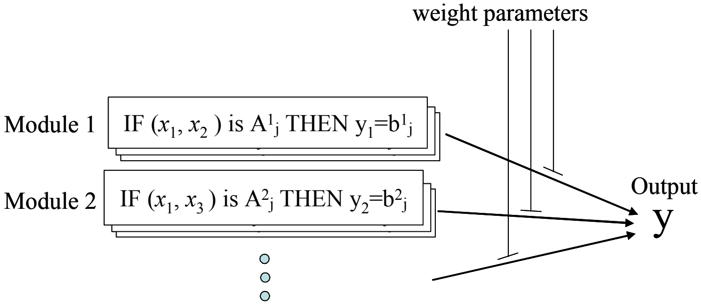

weight parameters
Module 1
IF (x_1, x_2) is A¹j THEN y₁=b¹j
Module 2
IF (x_1, x_3) is A²j THEN y₂=b²j
Output
y

Fig. 7. Modular Fuzzy Model

4.2 Application of modular fuzzy model for upper layer

I tackle to the "curse of dimensionality" in the multi-agent pursuit problem using above proposed modular fuzzy model method. The objective of this study is to restrain memory consumption of rules in reinforcement learning keeping its performance. In this study, the function of consequent part in Eq.(5) is defined as parameter of "real value", i.e. simplified fuzzy reasoning model (Ichihashi & Watanabe, 1990), in order for applying to the pursuit problem as:

$$\begin{aligned} &Rules-1:\{if\ P_1(x)\,is\ A_j^1\ then\ y_1=b_j^1\}_{j=1}^{m_1}\\ &\qquad\vdots\\ &Rules-i:\{if\ P_i(x)\,is\ A_j^i\ then\ y_i=b_j^i\}_{j=1}^{m_i}\\ &\qquad\vdots\\ &Rules-n:\{if\ P_n(x)\,is\ A_j^n\ then\ y_n=b_j^n\}_{j=1}^{m_n} \end{aligned} \tag{9}$$

The importance parameter in Eq.(8) is set as 1.0 in this study. Instead of "crisp type" modular model described in section 3.3, I apply the modular fuzzy model to the upper layer model in the hierarchical reinforcement learning for pursuit problem. In addition to the usual crisp partition of the agent position as shown in Fig.8, fuzzy sets of the position are defined as shown in Fig.9. The antecedent fuzzy sets are defined by Cartesian products of each fuzzy set on the state of the agent position.

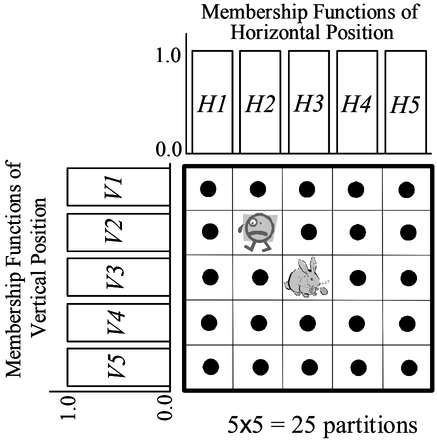

Membership Functions of Horizontal Position
1.0
0.0
H1
H2
H3
H4
H5
Membership Functions of Vertical Position
V1
V2
V3
V4
V5
1.0
0.0
5x5 = 25 partitions

Fig. 8. Usual Crisp Partition of Agent Position

u in Eq.(2) is calculated by the modular fuzzy model and is learned considering the membership degree of the rules by the profit sharing algorithm. In this study, I assume that the number of fuzzy sets and parameters in the premise part is decided in advance. The parameters of real value in the consequent part are learned by the profit-sharing algorithm. The parameters are modified as:

$$\Delta b_j^i = \frac{h_j^i}{\sum_{k=1}^{m_i} h_k^i} k \tag{10}$$

where *k* denotes the reinforcement function in Eq.(2). The denominator in Eq.(10) can be omitted in actual processing because its value is always 1.0 from the definition of fuzzy sets described above.

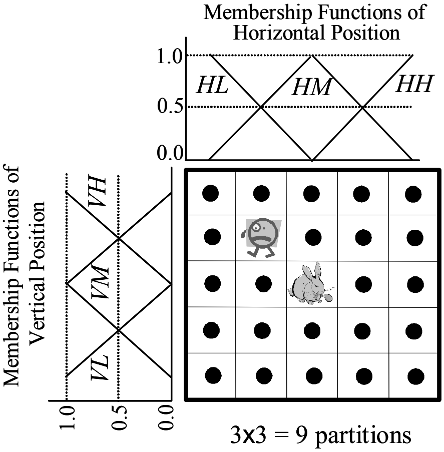

Membership Functions of Horizontal Position
1.0
0.5
0.0
HL
HM
HH
Membership Functions of Vertical Position
VH
VM
VL
1.0
0.5
0.0
3x3 = 9 partitions

Fig. 9. Fuzzy Partition of Agent Position

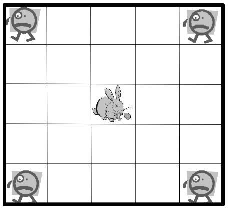


Fig. 10. Initial Placement of the Agents in 5x5 environment

5. Numerical experiments

5.1 Results compared with conventional learning methods

In the pursuit problem, the performance of the proposed hierarchical modular reinforcement learning method is compared with conventional methods through computer simulations. The size of the pursuit problem is 5x5. The absolute coordinate of the agent position is used in the experiments. The reason why relative coordinate is not used in the experiments is to evaluate essential performance of the proposed algorithm in terms of precision of learning, learning speed, and the memory consumption. As basic simulation conditions, each agent cannot communicate each other but can monitor the position of the other agents. The rule of the prey agent behavior is set as random behavior because the random behavior theoretically involves every action strategies. The initial placement of the prey agent and the hunter agents is shown in Fig.10.

The proposed methods are compared with the simple Q-Learning algorithm in order to evaluate basic performance of the methods. In the experiments, it is assumed that the Q-Learning agent(not hierarchically structured) can only utilize the position of the prey agent in addition to own position. The Q-Learning agent decides the action by calculating Q-value defined as $Q(g, s_e, a_e)$ from the sensed position of the prey agent and own position, where s_e is the position of the agent e, a_e is the corresponding action of the agent e, and g is the position of the prey agent.

As for hierarchical modular reinforcement learning agents, three methods are simulated. The expressions of the upper layer are different, though their hierarchical structures and the lower layer driven by Q-Learning are the same. The first method is structured as the complete expressed upper layer. From all positions of the hunter agents and the prey agent, the target position to move is decided. The number of rules in upper layer is 25*25*25*25*25=9,765,625. The second method is "crisp" modular model for upper layer. The number of rules in upper layer of each agent is (25*25*25*25)*3= 1,171,875. The last method is the modular fuzzy model for upper layer. Detailed constructions of the model are described in next subsection. For example, the 1st agent of the modular fuzzy model for upper layer is constructed as:

$$\begin{aligned}
&Rules-1: \ \{if \ \left[g, h_1, h_2, h_3\right] \ is \ A_j^1 \ then \ \ y_1 = b_j^1\}_{j=1}^{50,625} \\
&Rules-2: \ \{if \ \left[g, h_1, h_2, h_4\right] \ is \ A_j^2 \ then \ \ y_2 = b_j^2\}_{j=1}^{50,625} \\
&Rules-3: \ \{if \ \left[g, h_1, h_3, h_4\right] \ is \ A_j^3 \ then \ \ y_3 = b_j^3\}_{j=1}^{50,625}
\end{aligned} \tag{11}$$

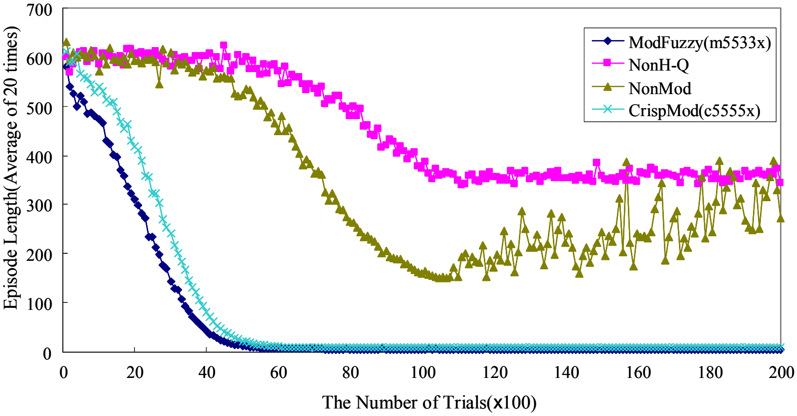


Fig. 11. Simulation Results

where g is the position of the prey agent, h is the position of the hunter agent, and b is the parameter of consequent part of the fuzzy rule. The fuzzy set A is constructed by combining the crisp sets of own agent position and prey agent position with the fuzzy sets of the other two hunter agent positions defined by partitioning the grid into 3×3 as shown in Fig.9. The number of rules in upper layer is much smaller than the others, i.e. (25*25*9*9)*3=151,875.

I perform the simulation 20 times for each method. The number of trials in the simulation are 20,000. The results are shown in Fig.11. The depicted data is averaged value of 20 series after averaging each sequential 100 trials. The results by the modular fuzzy model(depicted as ModFuzzy) show the best performance compared with the other methods. Both the learning speed and the precision of learning are desirable. Furthermore required memory amount is much smaller than the other methods. The results by “crisp” modular model(depicted as CrispMod) show also good performance. The complete expression model(depicted as NonMod) cannot acquire rules efficiently and the performance is deteriorated over time. This seems to be caused by the sparsity of model expression. The simple Q-Learning agent (NonH-Q) is not so bad unexpectedly in the small 5×5 grid world. The strategy only to approach to the prey agent acquired by the simple non-hierarchical Q-Learning might be reasonable in such small world. However, as the knowlede about surrounding task cannot be learned at all in such model expression, successful surrounding completely depends upon accidental behavior of the prey agent.

5.2 Detailed results by proposed model

In order to construct the modular fuzzy model, the important issue is to decide the dimension of projection in rule modules. Furthermore the number of partition should be also decided appropriately. In the pursuit problem, as the positions of own agent and the prey agent are indispensable by nature, the issue is restricted to decide the number of the other hunter agents included in model expression and the number of partition, i.e. crisp or fuzzy. In this study, the projection is extended step by step through modeling(reinforcement learning) from one other hunter agent added. The number of partition for each position is changed as well as the dimension. The results are summarized in Table 1. In this Table, averaged value, standard deviation, and standard error of episode lengh average of last 100 trials in 20 times simulation are shown as well as the number of partition and the number of rules corresponding to the model. From the results of first four models, own position of the agent might be partitioned by crisp sets, i.e. m353xx. From further results of next four models, own position of the agent and position of the target, i.e. prey agent, might be partitioned by crisp sets, i.e. m553xx. From these obserbations, the model construction is heuristically performed as shown in the last four results in the Table. From the results m5533x model has best performance among the models. Compared results with good

Model ID	The Number of Partition of Agent Position					The Number of Rules for One Agent	Episode Length of Last 100 trials (20 times)		
	Target	Own	Other1	Other2	Other3		Average	Standard Deviation	Standard Error
m333xx	9	9	9			2,187	225.77	310.71	69.48
m533xx	25	9	9			6,075	142.76	68.08	15.22
m335xx	9	9	25			6,075	98.27	44.75	10.01
m353xx	9	25	9			6,075	8.25	1.70	0.38
m535xx	25	9	25			16,875	121.99	85.53	19.12
m553xx	25	25	9			16,875	5.97	0.50	0.11
m355xx	9	25	25			16,875	10.94	1.06	0.24
c555xx	25	25	25			46,875	11.30	22.20	4.96
m3355x	9	9	25	25		151,875	115.76	33.90	6.92
m5533x	25	25	9	9		151,875	5.81	0.33	0.07
c5555x	25	25	25	25		1,171,875	9.07	0.67	0.14
u55555	25	25	25	25	25	9,765,625	271.49	283.88	63.48

Notes of Model ID: m5533x

The number of partition: Target, Own, Other1, Other 2, Other3

m : modular fuzzy model
c : crisp modular model
u : usual memory type

3 : fuzzy partition
5 : crisp partition
x : void (not used in model)

Table 1. Detailed Results of Modular Model

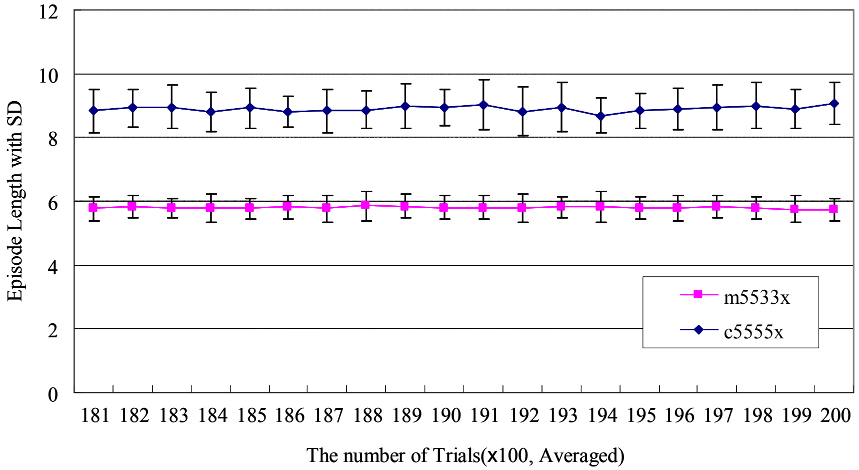


Fig. 12. Comparison of Modular Fuzzy Model and Crisp Modular Model

model(c5555x) are shown in Fig.12. The significance of the m5533x model performance compared with the other good model performance is also investigated by the *t* test. The result compared with m553xx model is that null hypothesis, i.e. the means do not differ, is rejected with statistical significance level of 0.01. As the results compared with the other model are obvious, the description is omitted.

The results by the proposed model are considered that the learned agent can perform surroundig task within six times movement against almost all behavior pattern of the prey agent. This level cannot be attained without collaborative behavior of the learned agent. In addition to its drastically improved learning speed, it can be said that the precision level of learning is sufficient compared with the conventional techniques.

6. Conclusion

In this chapter, I focused on the pursuit problem and proposed a hierarchical modular reinforcement learning that Profit Sharing learning algorithm is combined with Q Learning reinforcement learning algorithm hierarchically in multi-agent environment. As the model structure for such huge problem, I proposed a modular fuzzy model extending SIRMs architecture. Through numerical experiments, I showed the effectiveness of the proposed algorithm compared with the conventional algorithms. My future plan concerning with the proposed methods includes application of another multi-agent problem or complex task problem.

7. References

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning*, MIT Press

Watkins, C. J. & Dayan Y. (1988). Technical Note: Q-Leaning, *Machine Learning*, Vol.8, pp.58-68

Grefenstette, J. J. (1988). Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms, *Machine Learning*, Vol.3, pp.225-245

Miyazaki. K.; Kimura. H. & Kobayashi. S. (1999). Theory and Application of Reinforcement Learning Based on Profit Sharing, *Journal of JSAI*, Vol.14, No.5, pp. 800-807

Miyazaki. S.; Arai. S. & Kobayashi. S. (1999). A Theory of Profit Sharing in Multi-agent Reinforcement Learning, *Journal of JSAI*, Vol. 14, No.6, pp.1156-1164

Benda. M.; Jagannathan. V. & Dodhiawalla. R. (1985). On Optimal Cooperation of Knowledge Sources, *Technical Report*, BCS-G2010-28, Boeing AI Center

Ito. A. & Kanabuchi. M. (2001). Speeding up Multi-Agent Reinforcement Learning by Coarse-Graining of Perception –Hunter Game as an Example-, *Transaction of IEICE*, Vol.J84-D-1, No.3, pp.285-293

Takahashi. Y. & Asada. M. (1999). Behavior Acquisition by Multi-Layered Reinforcement Learning, *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics.*, pp.716-721

Morimoto. J. & Doya. K. (2000). Acquisition of Stand-up Behavior by a Real Robot using Hierarchical Reinforcement Learning, *Proceedings of International Conference on Machine Learning*, pp. 623-630

Ono. N. & Fukumoto. K. (1996). Multi-agent Reinforcement Learning: A Modular Approach, *Proceedings 2nd International Conference on Multi-agent Systems*, pp.252-258, AAAI Press

Fujita. K. & Matsuno. H. (2005). Multi-agent Reinforcement Learning with the Partly High-Dimensional State Space, *Transaction of IEICE*, Vol.J88-D-1, No.4, pp.864-872

Murano. H. & Kitamura. S. (1997). Q-Learning with Adaptive State Segmentation(QLASS), *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp.179-184

Hamagami. T.; Koakutsu. S. & Hirata. H. (2003). An Adjustment Method of the Number of States on Q-Learning Segmenting State Space Adaptively, *Transaction of IEICE*, Vol. J86-D1, No.7, pp.490-499

Seki. H.; Ishii. H. & Mizumoto. M. (2006). On the Generalization of Single Input Rule Modules Connected Type Fuzzy Reasoning Method, *Proceedings of the SCIS&ISIS2006*, pp.30-34

Yubazaki. N.; Yi. J.; Otani. M. & Hirota. K. (1997). SIRMs Dynamically Connected Fuzzy Inference Model and Its Applications, *Proceedings of IFSA'97*, vol.3, pp.410-415

Takahashi. Y. & Watanabe. T. (2006). Learning of Agent Behavior Based on Hierarchical Modular Reinforcement Learning, *Proceedings of the SCIS&ISIS2006*, pp.90-94

Ichihashi. H. & Watanabe. T. (1990). Learning Control System by a Simplified Fuzzy Reasoning Model, *Proceedings of the 3rd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp.417-419

Takagi. T. & Sugeno. M. (1985). Fuzzy Identification of Systems and Its Applications to Modeling and Control, *IEEE Transaction on Systems, Man, and Cybernetics*, Vol. 15, pp. 116-132

Auda. G. & Kamel. M. (1999). Modular Neural Networks: A Survey, *International Journal of Neural Systems*, Vol.9, No.2, pp.129-151

10

Random Forest-LNS Architecture and Vision

Hassab Elgawi Osman
Imaging Science and Engineering Laboratory, Tokyo Institute of Technology
Japan

1. Introduction

Combining multiple classifiers (e.g., decision trees) to build an ensemble is an advanced machine learning technique with substantially improvement over single-based classifiers. *Random forests* (RFs) (1), a representative decision tree-based ensemble has been emerged as a principle machine learning tool combining properties of efficient classifier and feature selection model running on general-purpose processor (GPP-based) custom-hardware and optimized operating systems. Rather than minimizing training error, RF minimizes the generalization error, while being fast to train, proven not to overfit, and computationally effective, ($O(\sqrt{V}TlogT)$, where V is the number of variables and T is the number of observations). These merits make RF a potential tool suited for adaptive classification problems. RF has been applied to vision problems such as object recognition (2–7). It has also been used for OCR (8) and for key point recognition (9). Despite of the appearance success of RF virtually no work has been done to map from its ideal mathematical model to compact and reliable hardware design.

In this chapter we present object recognition system implemented on a field programmable gate array (FPGA), enables learning algorithm to scale up. Fig.1 shows the general architecture of the proposed recognition system, composed of two main steps, each comprises several computational models. In the first step, objects are automatically represented as covariance matrices followed by a tree-based RF detector that operates on-line. We have shown in (4) utilizing a bag of covariance matrices as object descriptor improves the accuracy of object recognition while speed up the learning process, so we are extending this technique, present its hardware architecture. The on-line RF detector is designed using Logarithmic Number System (LNS) (10), RF-LNS, allows the reduction of the required word-length to 16 bits, and consequently a general-purpose microprocessor of the same word-length can be used. For the compact architecture we made RF-LNS comprises few computation modules, referred to as 'Tree Units', 'Majority Vote Unit', and 'Forest Units'. The main contribution of our approach (in addition to its impacts on the tradeoff between algorithmic setting accuracy and hardware implementation cost) is three-fold: (1) its direction towards arithmetic complexity reduction using a modified RF based on LNS (RF-LNS), (2) it has been designed in order to be easily integrated in a system-on-chip (SoC), which can perform both automatic feature selection and recognition, and (3) it allows for fair comparison with floating-point (FP) and fixed-point (FX) implementations. We test and verified the model functionality using numerical simulation, present results obtained using examples from GRAZ02 dataset (11). First, in Section 2 we present related works and highlight on general constrains in implementing hardware-based recognition systems. Section 3 shows the object descriptor we used and overview on RF al-

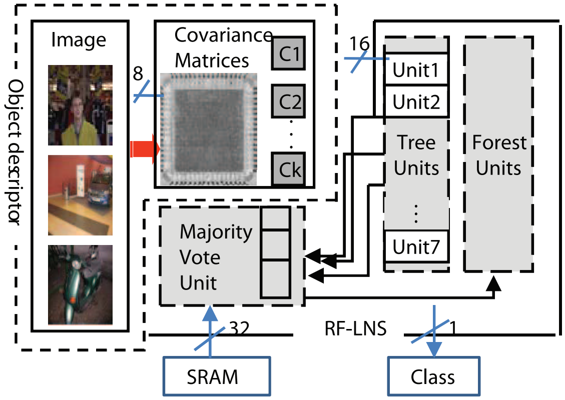


Fig. 1. Object Recognition based on RF-LNS which is optimized to be easily integrated in a System-on-Chip (SoC) platform implementation.

gorithmic settings. In Section 4 we present full architecture and design of our recognition system. We follow with experimental evaluation and estimation of the required precision in Section 5. A brief conclusion appears in Section 6.

2. Hardware-based Machine Learning

Perhaps motivated by the high computational complexity of many software-oriented machine vision algorithms, there have been several attempts to create faster execution hardware implementations which are able to identify and localize objects in a given scene or an image, achieve high recognition performance. There are studies about Pulsed Neural Network (PNN) that employ Pulsed Neuron (PN) or Spiking Neuron object localization and processing. The PN models and have the ability to adapt, much better than traditional neural nets. The Kerneltron (12; 13) is a SVM classification module, with a system precision resolution of no more than 8 bits. A fully digital architecture for SVM classification employing the linear and RBF kernels is proposed. The minimal word size they are able to use is 20 bits. In (14) hardware implementation of Decision Trees (DTs) is proposed. However to the best of our knowledge, ours is the first attempt to implement RF in hardware. We predict further progress using this approach.

2.1 Hardware implementations: problems and constraints

Any kind of hardware implementations of machine vision algorithms be it analog, digital, or optical, brings along various constraints:

- *Algorithematic design*: Automatic optimize settings of the parameters.
- *Accuracy and efficiency*: Hardware implementations can only offer limited accuracy. FP arithmetics are costly in terms of the number of logic elements required while FX implementation may speed up the algorithm but is leading to a definitively power consumption with marginally lose in precision.
- *Area*: The tradeoff between accuracy required and hardware (chip) area available. Accuracy often comes at the price of an area penalty.

- *GPP vs. FPGA*: A general purpose processor's (GPP) hardware contains all the basic blocks needed to build any logic of mathematical function imaginable but the limitations are in the parallelism available in the program, i.e. performance, and power consumption. FPGA provides flexibility to cope with the current evolving applications but at the cost of large performance, area, power and reconfiguration time penalties.

2.2 logarithmic Number System (LNS)

LNS is an alternative way to represent real numbers/values beside the conventional FP representation. The idea is to convert values into logarithms once and keep them in this representation throughout the entire computation. The LNS represents a number by the exponent in a certain base and a sign bit. The multiplication of two numbers is simply the sum of the two numbers' exponent parts, $\log_2(x \cdot y) = \log_2(x) + \log_2(y)$, divisions and square roots are implemented by fixed-point subtraction and bit shift respectively. However, the addition of two LNS numbers, $\log_2|(X,Y)| = X + \log_2|1 + 2^{Y-X}|$ is not a linear operation and requires two fixed-point adder/subtractors, and lookup-tables (LUTs) process (Function Generators (FGs)). The size of LNS adders increases exponentially as the operands' word lengths increase. Thus the LNS arithmetic systems usually have advantages of low precision and constant relative error.

3. Algorithmtic Considerations

The proposed object recognition approach consists of two basic models, a model for object descriptor based on covariance matrices (4; 15) and a classifier based on on-line variant of RF implemented on FPGA using LNS. First we introduce the algorithmtic settings of each model.

3.1 Covariance Matrices Descriptor

We have used bag of covariance matrices (Fig.2), to represent an object region.
Let I be an input color image. Let F be the dimensional feature image extracted from I

$$F(x,y) = \phi(I,x,y) \tag{1}$$

where function ϕ can be any feature maps (such as intensity, color, etc). For a given region $R \subset F$, let $\{z_k\}_{k=1\cdots n}$ be the d dimensional feature points inside R. We represent region R with $d \times d$ covariance matrix C_R of feature points.

$$C_R = \frac{1}{n-1} \sum_{k=1}^{n} (z_k - \mu)(z_k - \mu)^T \tag{2}$$

where μ is the mean of region R centered at the point.

3.2 Image Labeling

We gradually build our knowledge of the image from features to covariance matrix to a bag of covariance matrices. Starting by forming covariance matrix C from image features such that each feature Z in C has intensity $\mu(z)$ and associated variance $\lambda^{-1}(z)$, so λ is the inverse variance (precision). We then group covariance matrices as a set of spatially grouped feature in C that are likely to share common labels into a bag of covariance matrices.

Covariance matrix. Different regions of an object may have different descriptive powers and, hence, a difference impact on learning and recognition (Fig.2A). Following (15), we represent image objects with five covariance matrices $C_{i=1\cdots5}$ of the feature computed inside R (Fig.2B),

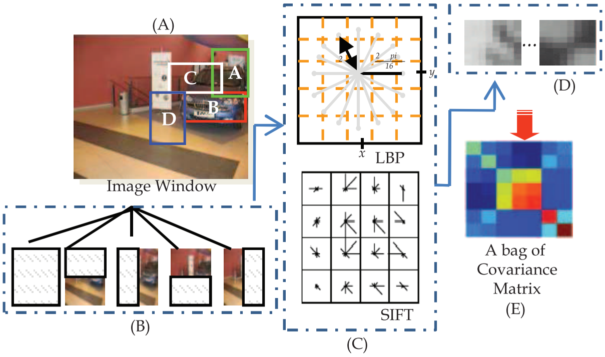


Fig. 2. (A) Rectangles are examples of possible regions for histogram features. Stable appearance in Rectangles A, B and C are good candidates for a car classifier while regions D is not. (C) Top, points sampled to calculate the LBP around a point (x,y). Bottom, the use of standard invariant feature (SIFT). (D) Any region can be represented by a covariance matrix. Size of the bag is proportional to the number of features used, while the size of the covariance matrix depends on the dimension of the features.

noting that features in the covariance matrix may be used in multiple image locations.

Color. Color is described by taken Ohta space histogram values of pixels $(I_1 = R + G + B/3$, $I_2 = R - B$, $I_3 = (2G - R - B)/2)$. This histogram is chosen because it is less sensitive to variations in illumination. Ohta values for each pixel in an image are clustered using k-means, e.g., each pixel in image I is assigned to the nearest cluster center, then histogram frequency is normalized.

Appearance. We have used histograms of Local Binary Patterns (LBPs) for representing each feature's appearance in some appearance space. Fig.2C depicts the points that must be sampled around a particular point (x,y) in order to calculate the LBP. In our implementation, each sample point lies at a distance of 2 pixels from (x,y). Instead of the traditional 3×3 rectangular neighborhood, we sample neighborhood circularly with two different radii (1 and 3). The resulting operators are denoted by $LBP_{8,1}$ and $LBP_{8,1+8,3}$, where subscripts tell the number of samples and the neighborhood radii.

A bag of covariance matrices. A bag of covariance which is a concatenation of Ohta color space histogram, and appearance model based on LBP and Scale Invariant Feature Transform (SIFT) of different features of an image region is presented in Fig.1E. Then estimate the bag of covariance matrix likelihoods $P(I_i|C, I_i)$ and the likelihood that each bag of covariance matrices is homogeneously labeled. We use this representation to automatically detect any target in images. We then apply on-line RF learner to select object descriptors and to learn an object classifier.

3.3 RF for Recognition

A detailed discussion of Breiman's RF (1) learning algorithm is beyond our scope here, however, in order to simplify the further discussion, we briefly define some fundamental terms:

Decision-tree. For the k-th tree, a random covariance matrix C_k is generated, independent of the past random covariance matrices $C_1, \ldots, C_{k-1}$, and a tree is grown using the training set of positive (contains the object relevant to the class) and negative (does not contain the object) image I, and covariance feature C_k. The decision generated by a random tree corresponds to a covariance feature selected by learning algorithm. Each tree casts a unit vote for a single matrix, resulting in a classifier $h\left(I,C_k\right)$.

Forest. Given a set of M decision trees, a forest is computed as ensemble of these tree-generated base classifiers $h\left(I,C_k\right)$, $k=1,\ldots,n$, using a majority vote.

Majority vote. If there are M Decision Trees, the majority voting method will give a correct decision if at least $floor(M/2)+1$ decision trees gives correct outputs. If each tree has probability p to make a correct decision, then the forest will have the following probability P to make a correction decision.

$$P = \sum_{\mathrm{i=floor(M/2)+1}}^{b} \left(\begin{array}{c} M \\ i \end{array}\right) p(1-p) \tag{3}$$

3.4 On-line RF for Recognition

To obtain an on-line algorithm, the steps above must be on-line where the current base classifier is updated whenever a new sample arrives. In particular our on-line RF involves two steps in inferring the object category (Algorithm 1). First, based on covariance object descriptor we develop a new, conditional permutation scheme for the computation of feature importance measure. Second, the fixed set tree K is initialized, then individual trees in RF are incrementally generated by specifically selected covariance matrix from the bag of covariance matrices. For updating, any on-line learning algorithm may be used, but we employ a standard Karman filtering technique.

Algorithm 1 On-line Random Forests

```
Initially select the number K of trees to be generated.
for $k = 1,2,\cdots,K$ do
    $\grave{T}$ b̄ootstrap sample from $T$ initialize $e=0, t=0, T_k=\phi$
    Do until $T_k = N_k$
    Vector $C_k$ that represent a bag of covariance is generate
    Construct Tree $h\left(I,C_k\right)$ using any decision tree algorithm
    Each Tree makes its estimation based on a single matrix from the bag of covariance
        matrices at $I$
    Each Tree casts a vote for most popular covariance matrix at image $I$
    The popular covariance matrix at $I$ at is predicted by selecting the matrix with max
        votes over $h_1,h_2,\ldots,h_k$
    $h_l = arg\ max_y \sum_{k=1}^{K} I(h_k(x)=y)$
    Return a hypothesis $h_l$
end for
Get the next sample set
Output: Proximity measure, feature importance, a hypothesis $h$
```

4. Hardware Architecture

4.1 FPGA Architecture

All FPGAs consist of three major components: 1) logic blocks (LBs); 2) I/O blocks; and 3) programmable routing, as shown in Fig.3(A). A logic block (LB) is functionally complete logic circuits, partitioned to LB size, mapped and routed, and place in an interconnect framework to perform a desired operation. Field programmability is achieved through switches (transistors controlled by memory element or fuses) and each I/O block is programmed to act as an input or output, as required, i.e., N-input LUTs can implement any n-input boolean function. The programmable routing is also configured to make the necessary connections between logic blocks, and from logic blocks to I/O blocks. The processing power of an FPGA is highly dependent on the processing capabilities of its LBs and the total number of LBs available in the array. Generally, FPGAs use logic blocks that contain one or more LUT, typically with at least four-inputs. A four-input LUT can implement any binary function of four logic inputs. Fig.3(B) shows the architecture of a simple LB containing one four-input LUT and one flip-flop for storage.

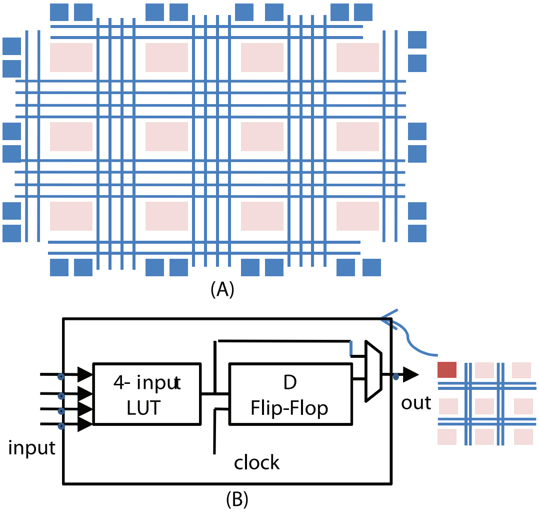


Fig. 3. (A) Granularity and interconnection structure of generic Xilinx FPGA. (B) An architecture of a logic block with one, four-input LUT use for implementation of memory and shift registers.

4.2 Transform into Log-domain

Rather than adapting the FP arithmetic we based on LNS, eliminate the need for multiplications and division, allowing all operations to be carried out using shifts and additions. In LNS, a number x is represented in signed magnitude form, i.e., as a pair (S,e), where $x = (-1)^s(r)^e$, S being the sign bit (which is either 0 or 1 according to the sign of x) and e being the signed exponent of the radix r (usually in radix 2). The exponent e is expressed in fixed-point binary mode with say, G bits for the integer part and F bits for the fractional part and one bit for the sign of the exponent, i.e., with a total of $(G + F + 1)$ bits. If the radix is considered to be 2, then the smallest number that can be represented using the scheme is 2^{-N}, where $N = (s^G - 1) + (1 - 2^{-F}) = (2^G - 2^{-F})$. The ratio between two consecutive numbers is equal to $r^{2^{-F}}$, and the corresponding precision e is roughly $(\ln r)2^{-F}$. Typically, if $G = 5$, $F = 30$, and $r = 2$, we can have a precision of 30 bits in radix 2. However, for the purpose of comparison with the precision of FP representation, e will be assumed as $2^{-23}(\approx 10^{-7})$. Numbers closer to zero, are represented with better precision in LNS than FP systems. However, LNS depart from FP in that, the relative error of LNS is constant and LNS can often achieve equivalent signal-to-noise ratio with fewer bits of precision relative to FP architectures.

4.3 Object Recognition Architecture based on RF-LNS

Fig.4 shows RF-LNS object classifier proposed in this paper. The classifier consists of three main design blocks (a) The LG block; (b) The ACC block; and (c) The SIGM block. The 'Covariance Unit' in Fig.1 contains all the features extracted from an image in a form of bag of covariance matrices. The output of covariance descriptor becomes the input of the RF-LNS classifier. However, Function ϕ given by eq(1) consists of float values which require much place for storing in an FPGA memory. In order to reduce the hardware cost, we propose to approximate the function ϕ using LG. This function will transform float elements of the ϕ into binary elements. For 'Tree Units' we compute 16 covariance matrices in 32 bit memory. Basically the decision trees consist of two types of nodes: *decision nodes*, corresponding to state variables and *least nodes*, which correspond to all possible covariance features that can be taken. In a decision node a decision is taken about one of the input. Each least node stores the state values for the corresponding region in the image, meaning that a least node stores a value for each relevant covariance matrix that can be taken. The tree starts out with only one least node that represents the entire image region then, a decision has to be made whether the node should be split or not. ACC block that does the accumulation operations at each node. Once a tree is constructed it can be used to map an input vector to a least node, which corresponds to a region in the image. Then a decision tree can be converting into an equivalent 'Tree Unit' by extracting one logic function per class from the tree structure. Each 'Tree Units' gives a unit vote for its popular object class. 'Forest Unit' is an ensemble of trees grown incrementally to a certain depth. The object is recognized as the one having the majority vote, stored at 'Majority Vote Unit'. The SIGM block that performs the sigmoid evaluation function for majority votes.

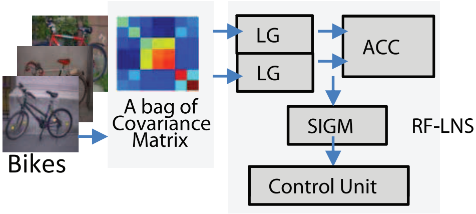


Fig. 4. RF-LNS object classifier Architecture.

5. Evaluation

The functionality of the proposed system was simulated, and the hardware is programmed. We now demonstrate the usefulness of this frame work in the area of recognition generic objects such as bikes, cars, and persons.

5.1 Dataset

We have used data derived from the GRAZ02[1] dataset (11), a collection of 640 × 480 24-bit color images. As can be seen in Fig.5, this dataset has three object classes, bikes, cars, persons, and in addition to the background class (270 images). This database contains variability with respect to scale and clutter. Objects of interest are often occluded, and they are not dominant in the image. According to (16) the average ratio of object size to image size counted in number of pixels is 0.22 for bikes, 0.17 for people, and 0.9 for cars. Thus this dataset is more complex dataset to learn detectors from, but of more interest because it better reflects the real world complexity. Table 1 reports the number of images and objects in each class, 380 images are available for background class .

Dataset	Images	Objects
Bikes	365	511
Cars	420	770
Persons	311	785
Total	1096	2066

Table 1. Number of images and objects in each class in the GRAZ02 dataset.

5.2 Experimental Settings

Our RF-LNS is trained with varying amounts (10%,50% and 90% respectively) of randomly selected training data. All images not selected for the training split were put into the test split. For the 10% training data experiments, 10% of images were selected randomly with the remainder used for testing. This was repeated 20 times. For the 50% training data experiments, stratified 5 × 2 fold cross validation was used. Each cross validation selected 50% of the dataset for training and tested the classifiers on the remaining 50%; the test and training sets were then exchanged and the classifiers retrained and retested. This process was repeated 5 times. Finally, for the 90% training data situation, stratified 1 × 10 fold cross validation was performed, with the dataset divided into ten randomly selected, equally sized subsets, with each subset being used in turn for testing after the classifiers were trained on the remaining nine subsets.

6. Performances

GRAZ02 images contain only one object category per image so the recognition task can be seen as a binary classification problem: bikes vs. background (i.e., non-bikes), people vs. background, and car vs. background. Generalization performances in these object recognition experiments were estimated by statistic measure; the Area Under the ROC Curve (AUC) to measure the classifiers performance. AUC measures of classifier performance that is independent of the threshold, meaning it summarizes how true positive and false positive rates

[1] available at `http://www.emt.tugraz.at/~pinz/data/`

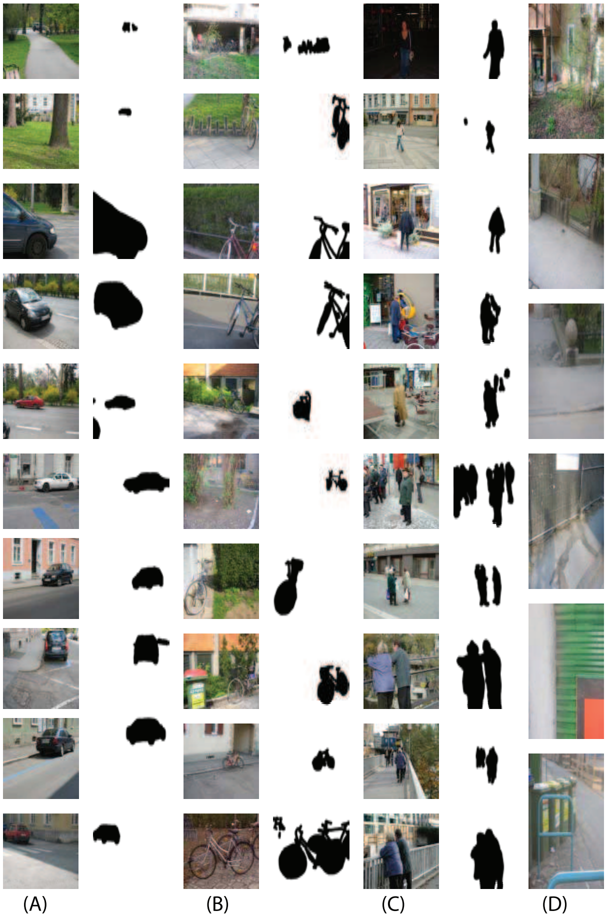


Fig. 5. Examples from GRAZ02 dataset (11) for four different categories: A) cars and ground truth, B) bikes and ground truth, C) persons and ground truth, and D) background.

change as the threshold gradually increases from 0.0 to 1.0, i.e., it does not summarize accuracy. An ideal perfect classifier has an AUC of 1.0 and a random classifier has an AUC of 0.5.

6.1 Finite Precision Analysis

The primary task here is to analyze the precision requirements for performing recognition. The RF-LNS precision was varied to ascertain optimal LNS precisions and compare them against the cost of using FP architectures. Tables 2, 3, and 4 give the mean AUC values across all runs to 2 decimal places for RF-LNS and training data amount combinations, for the bikes, cars and people datasets respectively. The performance of RF-LNS is reported with weight quantized with 4, 8, and 16 bits, and for different decision tree depths, from depth = 3 to depth

= 7. For example a figure of 85% means that 85% of object images were correctly classified but 15% of the background images were incorrectly classified (i.e. thought to be foreground). For RF-LNS to maintain acceptable performance, 16 bits of precision are sufficient for all GRAZ02 categories, even when only 10% training examples are used. Such low precision required by RF-LNS makes it competitive with FP arithmetic for our generic object recognition application.

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	D=3	D=4	D=5	D=6	D=7	D=3	h=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=7
10%	0.79	0.79	0.77	0.81	0.81	0.81	0.81	0.80	0.83	0.83	0.83	0.83	0.81	0.84	0.83
50%	0.86	0.86	0.82	0.81	0.83	0.88	0.89	0.85	0.88	0.86	0.90	0.90	0.86	0.89	0.89
90%	0.80	0.81	0.81	0.83	0.88	0.87	0.87	0.87	0.88	0.90	0.90	0.91	0.90	0.90	0.90

Table 2. Mean AUC performance of RF-LNS on the Bikes vs. Background dataset, by amount of training data. Performance of RF-LNS is reported for different Depths (D).

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	D=3	D=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=7
10%	0.66	0.70	0.70	0.75	0.71	0.68	0.73	0.73	0.76	0.73	0.71	0.75	0.75	0.77	0.75
50%	0.77	0.78	0.77	0.77	0.79	0.79	0.80	0.79	0.81	0.81	0.81	0.80	0.81	0.82	0.83
90%	0.77	0.75	0.75	0.73	0.79	0.81	0.81	0.78	0.78	0.82	0.83	0.83	0.81	0.80	0.85

Table 3. Mean AUC performance of RF-LNS on the Cars vs. Background dataset, by amount of training data. Performance of RF-LNS is reported for different Depths (D).

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	D=3	D=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=7
10%	0.83	0.73	0.77	0.77	0.79	0.77	0.74	0.80	0.79	0.81	0.80	0.78	0.81	0.81	0.82
50%	0.79	0.80	0.79	0.78	0.83	0.81	0.83	0.83	0.80	0.84	0.85	0.86	0.85	0.82	0.85
90%	0.80	0.80	0.81	0.78	0.83	0.81	0.82	0.82	0.80	0.86	0.88	0.86	0.83	0.83	0.87

Table 4. Mean AUC performance of RF-LNS on the Persons vs. Background dataset, by amount of training data. Performance of RF-LNS is reported for different Depths (D).

6.2 Efficiency and Hardware area

The efficiency of RF-LNS classifier is evaluated in terms of the number of slices. This is simply equivalent to hardware area required to achieve acceptable performance. Table 5 shows number of slice used by RF-LNS classifier as compared with 10- and 20-bit fixed-point (FX) implementations. The number of slices is reported for different Tree Unit for each dataset. RF-LNS takes almost the same number of slices as 10-bit FX but less than one-half of 20-bit FX implementation. This is interesting because 10-bit FX implementation has been widely recognized for not acceptable performance, particularly for recognition problem. Our design also achieved high speed clock rate processing. For the 1-bit RF-LNS, the power dissipation is small, and the area usage on FPGA is less than 2 percents.

7. Conclusions and Future Works

Efficient hardware implementations of machine-learning techniques yield a variety of advantages over software solutions: increased processing speed, and reliability as well as reduced cost and complexity. In this paper RF technique is modified so that classification is performed by LNS arithmetic. The model is applied for generic object recognition task, it shows that at low precision the RF-LNS hardware has significant area savings compared to the fixed-point alternative. With these characteristics, RF-LNS may be a good way for designing a real-time low power object recognition systems. Our future goals include further exploring precision requirements for hardware RF-LNS, noise analysis to determine the robustness of the hardware classifier and expanding LNS hardware architectures to other machine learning algorithms.

Dataset	Tree Units	16-bit LNS	10-bit FX	20-bit FX
Bikes	3	315	219	576
	4	498	407	713
	5	611	622	878
	6	823	835	1103
	7	1010	974	1345
Cars	3	277	283	603
	4	397	476	783
	5	536	694	866
	6	784	943	1002
	7	989	1287	1311
Persons	3	336	318	409
	4	534	535	657
	5	765	689	845
	6	878	926	1127
	7	1123	1158	1287

Table 5. Slices used for different tree units for each dataset.

8. References

[1] L. Breiman, “Random Forests,” *Machine Learning*, 45(1):5-32, 2001.

[2] F. Moomsmann, B. Triggs, and F. Jurie. “Fast discriminative visual codebooks using randomized clustering forests,” In *Proc. NIPS* 2006.

[3] J. Winn and A. Criminisi. “Object class recognition at a glance,” In *Proc. CVPR*, 2006.

[4] H. Elgawi Osman, “A binary Classification and Online Vision,” In *Proc. IJCNN*, 2009. pp.1142-1148

[5] A.Bosch, A.Zisserman, X.Munoz, “Image Classification Using Random Forests and Ferns,” *ICCV*, pp.1-8, 2007.

[6] J. Shotton, M. Johnson, R. Cipolla, “Semantic Texton Forests for Image Categorization and Segmentation,” In *Proc. CVPR*, pp.1-8 2008.

[7] F. Schroff, A. Criminisi, and A. Zisserman, “Object Class Segmentation using Random Forests,” In *Proc. BMVC* 2008.

[8] Y. Amit and D. Geman. “Shape quantization and recognition with randomized trees,” *Neural Computation* 9(7):15451588, 1997.

[9] V. Lepetit, P. Lagger, and P. Fua. “Randomized trees for real-time keypoint recognition,” In *Proc. CVPR*, 2005.

[10] H. Elgawi Osman, “Hardware-based solutions utilizing Random Forests for Object Recognition,” In *Proc. ICONIP*, Part II, LNCS 5507, pp. 760-767, 2008.

[11] A. Oplet, M. Fussenegger, A. Pinz and P. Auer. “Generic object recognition with boosting,” *TPAMI* 28(3):416-431, 2006.

[12] R. Genov and G. Cauwenberghs. “Kerneltron: Support Vector Machine in Silicon,” *IEEE Transactions on Neural Networks*, 14(5):1426-1434, 2003.

[13] R. Genov, S. Chakrabartty and G. Cauwenberghs. “Silicon Support Vector Machine with On-Line Learning,” *IJPRAI*, 17(3):385-404, 2003.

[14] M. Muselli and D. Liberati. “Binary Rule Generation via Hamming Clustering,” *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1258-1268, 2002.

[15] O. Tuzel, F. Porikli, and P. Meer. “Region covariance: A fast descriptor for detection and classification,” In *Proc. ECCV*, pp.589-600, 2006.

[16] A. Opelt. and Pinz A. “Object Localization with boosting and weak supervision for generic object recognition,” *In Kalvianen H. et al. (Eds.) SCIA 2005*, LNCS 3450, pp.862-871, 2005

An Intelligent System for Container Image Recognition using ART2-based Self-Organizing Supervised Learning Algorithm

Kwang-Baek Kim[1], Sungshin Kim[2] and Young Woon Woo[3]
[1]Silla University,
[2]Pusan National University,
[3]Dong-Eui University,
Korea

1. Introduction

Recently, the quantity of goods transported by sea has increased steadily since the cost of transportation by sea is lower than other transportation methods. Various automation methods are used for the speedy and accurate processing of transport containers in the harbor. The automation systems for transport container flow processing are classified into two types: the barcode processing system and the automatic recognition system of container identifiers based on image processing. However, these days the identifier recognition system based on images is more widely used in the harbors.

Identifiers of shipping containers are given in accordance with the terms of ISO standard, which consist of 4 code groups such as shipping company codes, container serial codes, check digit codes and container type codes (ISO-6346, 1995; Kim, 2003). And, only the first 11 identifier characters are prescribed in the ISO standard and shipping containers are able to be discriminated by automatically recognizing the first 11 characters. But, other features such as the foreground and background colors, the font type and the size of container identifiers, etc., vary from one container to another since the ISO standard doesn’t prescribes other features except code type (Kim, 2004; Nam et al., 2001). Since identifiers are printed on the surface of containers, shapes of identifiers are often impaired by the environmental factors during the transportation by sea. The damage to a container surface may lead to a distortion of shapes of identifier characters in a container image. So, the variations in the feature of container identifiers and noises make it quite difficult the extraction and recognition of identifiers using simple information like color values (Kim, 2004).

Generally, container identifiers have another feature that the color of characters is black or white. Considering such a feature, in a container image, all areas excepting areas with black or white colors are regarded as noises, and areas of identifiers and noises are discriminated by using a fuzzy-based noise detection method. Noise areas are replaced with a mean pixel value of the whole image area, and areas of identifiers are extracted and binarized by applying the edge detection by Sobel masking operation and the vertical and horizontal block extraction to the conversed image one by one. In the extracted areas, the color of identifiers is converted to black and one of background to white, and individual identifiers are extracted by using a 8-directional contour tacking algorithm. An ART2-based self-organizing supervised learning algorithm for the identifier recognition is proposed in this chapter, which creates nodes of the hidden layer by applying ART2 between the input layer and the hidden one and improves performance of learning by applying generalized delta learning and the Delta-bar-Delta algorithm (Vogl et al., 1998). Experiments using many images of shipping containers show that the presented identifier extraction method and the ART2-based supervised learning algorithm is more improved compared with the methods proposed previously.

2. The proposed container identifier recognition method

2.1 Extraction of container identifier areas

Due to the rugged surface shape of containers and noises vertically appeared by an external light, a failure may occur in the extraction of container identifier areas from a container image. To refine the failure problem, a novel method is proposed for extraction of identifier areas based on a fuzzy-based noise detection method.

In the proposed method, edges of identifiers are detected by applying Sobel masking operation to a grayscale image of the original image and extracts areas of identifiers using information on edges. Sobel masking operation is sensitive to noises so that it detects noises by an external light as edges. To remove an effect of noises in the edge detection, first, noise pixels are detected by a fuzzy method and replaced by the pixels with a mean gray value. Next, Applying Sobel masking to the noise-removed image, areas of container identifiers are separated from background areas.

2.2 Fuzzy-based noise detection

To remove noises by an external light, an container image is converted to a grayscale one and apply the membership function like Fig. 1 to each pixel of the grayscale image, deciding whether the pixel is a noise or not. In Fig. 1, C and E are categories being likely to belong to an area of identifiers, and D is the category being likely to be a noise. Eq. (1) shows the expression for the membership function of Fig. 1. The criterion to distinguish pixels of noise and non-noise using the degree of membership in the proposed method is given in Table 1.

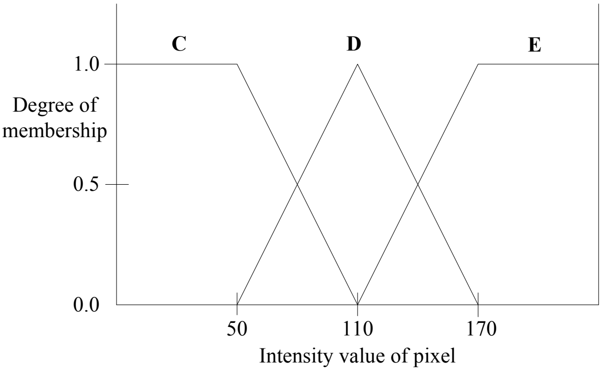


Fig. 1. Membership function(G) for gray-level pixels

$$\begin{aligned} &if\ (G \prec 50)\ or\ (G \succ 170)\ then\ u(G)=0 \\ &else\ if\ (G \succ 50)\ or\ (G \le 110)\ then\ u(G)=\frac{G-50}{110-50} \\ &else\ if\ (G \succ 110)\ or\ (G \le 170)\ then\ u(G)=\frac{110-G}{170-110} \end{aligned} \tag{1}$$

pixel of non-noise	u(G) < 0.42
pixel of noise	u(G) ≥ 0.42

Table 1. Criterion to distinguish pixels of noise and non-noise

To observe the effectiveness of the fuzzy-based noise detection, results of edge detection by Sobel masking are compared between the original image and the noise-removed image by the proposed method. Fig. 2 is the original container image, and Fig. 3 is the output image generated by applying only Sobel masking to a grayscale image of Fig. 2. Fig. 4 is results of edge detection obtained by applying the fuzzy-based noise removal and Sobel masking to Fig.2. First, the fuzzy-based noise detection method is applied to a grayscale image of the original image and pixels detected as noises are replaced with a mean gray value. Next, edges of container identifiers are detected by applying Sobel masking to the noise-removed image. As shown in Fig. 3, noise removal by the proposed fuzzy method generates more efficient results in the extraction of areas of identifiers.

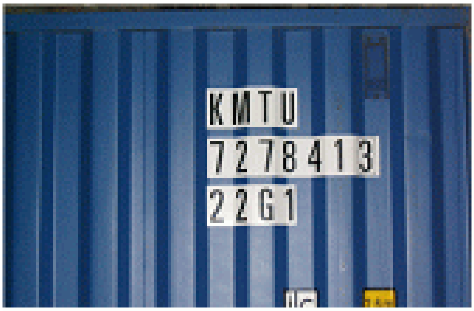


Fig. 2. An original container image

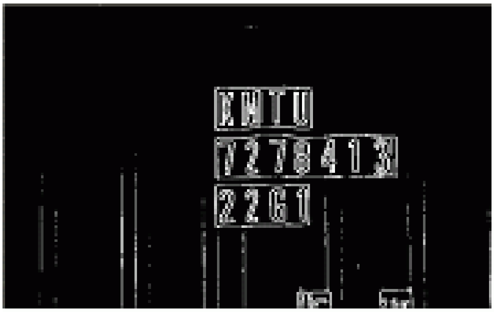


Fig. 3. Result of edge detection by only Sobel masking

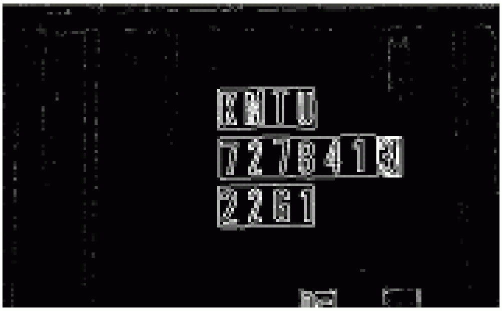


Fig. 4. Result of edge detection by fuzzy-based noise-removal and Sobel masking

2.3 Binarization of container identifier areas

Currently, the iterative binarization algorithm is mainly used in the preprocessing of pattern recognition. The iterative binarization algorithm, first, roughly determines an initial threshold, divides an input image to two pixel groups using the threshold, calculates a mean value for each pixel group, and sets the arithmetic mean of two mean values to a new threshold. And, the algorithm repeats the above processing until there is no variation of threshold value and sets the last value to the threshold value for binarization operation. In the case of a noise-removed container image, since the difference of intensity between the background and the identifiers is great, the iterative algorithm is able to provide a good threshold value.

2.4 Extraction of individual identifiers

Individual identifiers are extracted by applying the 8-directional contour tracking method (Chen & Hsu, 1989) to binarized areas of container identifiers. In the extraction process, the extraction of individual identifiers is successful in the case that the background color is a general color except white one like Fig. 5, and on the other hand, the extraction is failed in the case with white background color as shown in Fig. 6. In the binarization process, background pixels of a bright intensity are converted to black and identifier pixels of a dark intensity are converted to white. Since the contour tracking method detects edges of an area with black color, it can not detect edges of identifiers from target areas with white background. So a result of binarization process is reversed for identifier areas with white background. That is, background pixels are converted to white and identifier pixels to black. Fig. 7 shows that the pixel reversal lead to a success of edge detection in an identifier area with white background presented in Fig. 6.

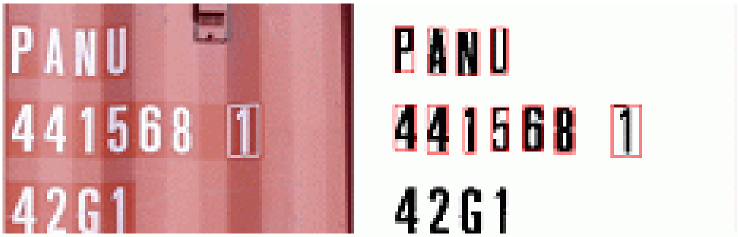


Fig. 5. Identifier area with a general color and successful results of edge extraction

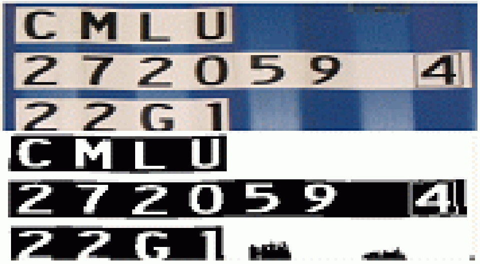


Fig. 6. Identifier area with white color and failed results of edge extraction

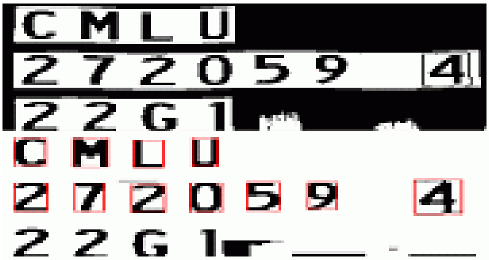


Fig. 7. Reversed binarized area of Fig. 6 and successful result of edge detection

The procedure of extracting individual identifiers using the 8-directional contour tracking method is as follow: P_i^r and P_i^c are pixels of horizontal and vertical directions being currently scanned in the identifier area, respectively, and P_i^{r+1} and P_i^{c+1} are pixels of the two directions being next scanned in the identifier area. And P_s^r and P_s^c are pixels of horizontal and vertical directions in the first mask of the 8-directional contour tracking.

Step 1. Initialize with Eq. (2) in order to apply the 8-neighborhood contour tracking algorithm to the identifier area, and find the pixel by applying tracking mask as shown in Fig. 8.

$$P_i^{r-1} = P_i^r, \quad P_i^{c-1} = P_i^c \tag{2}$$

Step 2. When a black pixel is found after applying the tracking mask in the current pixel, calculate the value of P_i^r and P_i^c as shown in Eq. (3)

$$P_i^r = \sum_{i=0}^{7} P_i^{r+1}, \quad P_i^c = \sum_{i=0}^{7} P_i^{c+1} \tag{3}$$

Step 3. For the 8 tracking masks, apply Eq. (4) to decide the next tracking mask.

$$\text{if } P_i^r = P_i^{r+1} \ \textit{and} \ P_i^c = P_i^{c+1} \text{ then rotates counter-clockwise.} \tag{4}$$

Step 4. Stop if P_i^r and P_i^c return back to P_s^r and P_s^c or go back to the Step 1 and repeat. If $\left|P_i^r - P_s^r\right| \le 1 \ \textit{and} \ \left|P_i^c - P_s^c\right| \le 1$ then break, else go back to Step 1.

Fig. 5 and Fig. 7 shows extraction results of individual identifiers by using the 8-directional contour tracking method.

6	5	4
7		3
0	1	2

EE

7	6	5
0		4
1	2	3

SE

0	7	6
1		5
2	3	4

SS

1	0	7
2		6
3	4	5

SW

2	1	0
3		7
4	5	6

WW

3	2	1
4		0
5	6	7

NW

4	3	2
5		1
6	7	0

NN

5	4	3
6		2
7	0	1

NE

Fig. 8. 8-diectional contour tracking masks

2.5 Recognition of container identifiers using ART2-based self-organizing supervised leaning algorithm

The error backpropagation algorithm uses gradient descent as the supervised learning rule to minimize the cost function defined in terms of the error value between the output value and the target one for a given input. Hence, the algorithm has the drawback that the convergence speed of learning is slower and the possibility of falling into the local minima is induced by the insufficient number of nodes in the hidden layer and the unsuitable initial connection weights. During the learning process, the algorithm uses credit assignment for propagating error value of the output layer's nodes backward to the nodes in the hidden layer. As a result, paralysis can be induced in the hidden layer. Generally, the recognition algorithms using the error backpropagation are plagued by the falling-off of recognition rate caused by the empirical determination of the number of hidden layer nodes and the credit assignment procedure(Kim & Yun, 1999).

Fuzzy C-Means-based RBF networks uses the fuzzy C-Means algorithm to generate the middle layer. It has a disadvantage of consuming too much time when applied to character recognition. In character recognition, a binary pattern is usually used as the input pattern. Thus, when the fuzzy C-Means algorithm is applied to the training pattern composed of 0 and 1, it is not only difficult to precisely classify input patterns but also takes a lot of training time compared to other clustering algorithms(Kim et al., 2005).

The ART2 architecture was evolved to perform learning for binary input patterns and also accommodate continuous valued components in input patterns(Carpenter et al., 1991). In the ART2 algorithm, connection weights are modified according to the calculation of mean values of all input patterns. Then the cluster center is calculated by adapting it to the new pattern.

However, the averaged mean value of the difference in input vector and connection weight is used for comparison with the vigilance factor, which leads to the possibility of an input pattern being classified to a similar cluster having different properties (Kim & Kim, 2004). This could happen particularly in cases where the pattern dimensionality is large and one feature drastically differs from the cluster center but its impact is minimized due to averaging all differences. When the traditional ART2 algorithm was applied to the recognition of container identifiers, it was observed that the recognition rate declined due to the classification of such different input patterns to the same cluster. Therefore, we propose a novel ART2-based hybrid network architecture where the middle layer neurons have RBF (Radial Basis Function) properties and the output layer neurons have a sigmoid function property.

An ART2-based self-organizing supervised learning algorithm for the recognition of container identifiers, is proposed in this chapter. First, a new leaning structure is applied between the input and the middle layers, which applies ART2 algorithm between the two layers, select a node with maximum output value as a winner node, and transmits the selected node to the middle layer. Next, generalized Delta learning algorithm and Delta-bar-Delta algorithm are applied in the learning between the middle and the output layers, improving the performance of learning. The proposed learning algorithm is summarized as follows:

1. The connection structure between the input and the middle layers is like ART2 algorithm and the output layer of ART2 becomes the middle layer of the proposed learning algorithm.
2. Nodes of the middle layer mean individual classes. Therefore, while the proposed algorithm has a fully-connected structure on the whole, it takes the winner node method that compares target vectors and output vectors and back-propagates a representative class and the connection weight.
3. The proposed algorithm performs the supervised learning by applying generalized Delta learning as the learning structure between the middle and the output layers.
4. The proposed algorithm improves the performance of learning by applying Delta-bar-Delta algorithm to generalized Delta learning for the dynamical adjustment of a learning rate. When defining the case that the difference between the target vector and the output vector is less than 0.1 as an accuracy and the opposite case as an inaccuracy, Delta-bar-Delta algorithm is applied restrictively in the case that the number of accuracies is greater than or equal to inaccuracies with respect to total patterns. This prevents no progress or an oscillation of learning keeping almost constant level of error by early premature situation incurred by competition in the learning process.

The detailed description of ART2-based self-organizing supervised learning algorithm is like Fig. 9.

3. Performance evaluation

The proposed algorithm is implemented by using Microsoft Visual C++ 6.0 on the IBM-compatible Pentium-IV PC for performance evaluation. 79 container images with size of 640x480 are used in the experiments for extraction and recognition of container identifiers. In the extraction of identifier areas, the previously proposed method fails to extract in images containing noises vertically appearing by an external light and the rugged surface shape of containers. On the other hand, the proposed extraction method detects and removes noises by using a fuzzy method, improving the success rate of extraction compared with the previously proposed. The comparison of the success rate of identifier area extraction between the proposed method in this chapter and the previously proposed method is like Table 2.

For the experiment of identifier recognition, applying the 8-directional contour tracking method to 72 identifier areas extracted by the proposed extraction algorithm, 284 alphabetic characters and 500 numeric characters are extracted. The recognition experiments are performed with the FCM-based RBF network and the proposed ART2-based self-organizing supervised learning algorithm using extracted identifier characters and compared the recognition performance in Table 3.

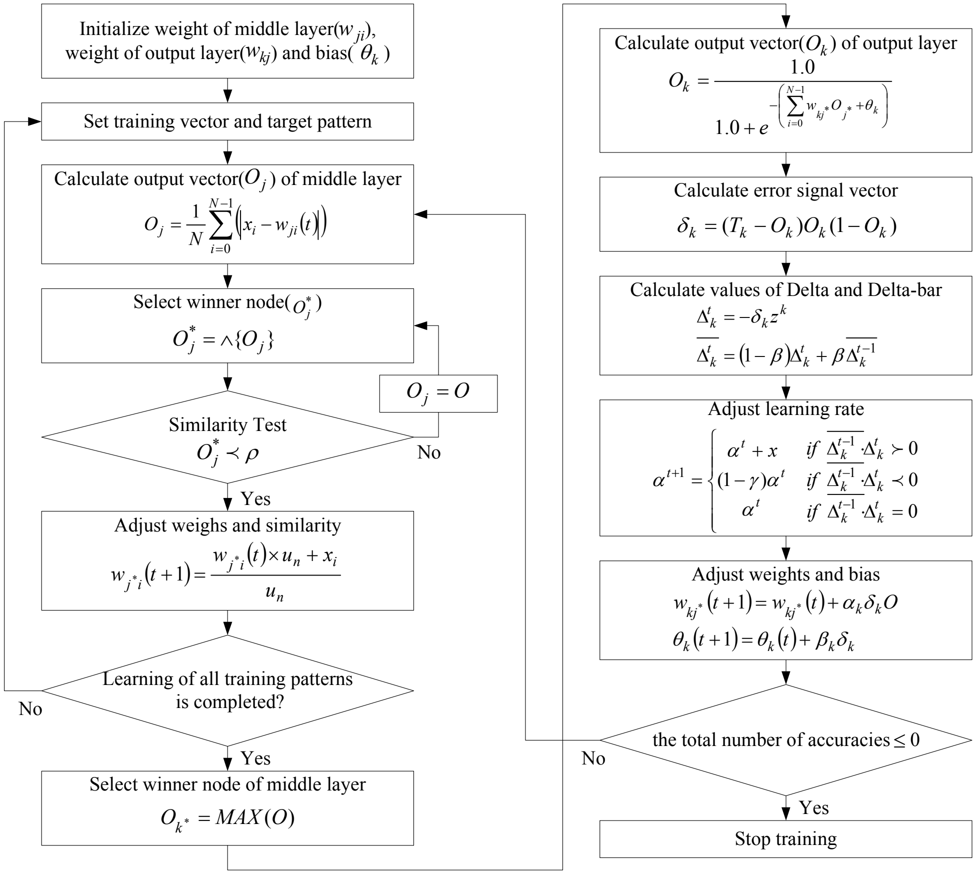

Initialize weight of middle layer(w_{ji}), weight of output layer(w_{kj}) and bias(θ_k)
Set training vector and target pattern
Calculate output vector(O_j) of middle layer
$O_j = \frac{1}{N}\sum_{i=0}^{N-1}\left(\left|x_i - w_{ji}(t)\right|\right)$
Select winner node(O_j^*)
$O_j^* = \wedge\{O_j\}$
$O_j = O$
Similarity Test
$O_j^* \prec \rho$
No
Yes
Adjust weighs and similarity
$w_{j^*i}(t+1) = \frac{w_{j^*i}(t)\times u_n + x_i}{u_n}$
Learning of all training patterns is completed?
No
Yes
Select winner node of middle layer
$O_{k^*} = MAX(O)$
Calculate output vector(O_k) of output layer
$O_k = \frac{1.0}{1.0 + e^{-\left(\sum_{i=0}^{N-1} w_{kj^*}O_{j^*} + \theta_k\right)}}$
Calculate error signal vector
$\delta_k = (T_k - O_k)O_k(1 - O_k)$
Calculate values of Delta and Delta-bar
$\Delta_k^t = -\delta_k z^k$
$\overline{\Delta_k^t} = (1-\beta)\Delta_k^t + \beta\overline{\Delta_k^{t-1}}$
Adjust learning rate
$\alpha^{t+1} = \begin{cases} \alpha^t + x & if\ \overline{\Delta_k^{t-1}}\cdot\Delta_k^t \succ 0 \\ (1-\gamma)\alpha^t & if\ \overline{\Delta_k^{t-1}}\cdot\Delta_k^t \prec 0 \\ \alpha^t & if\ \overline{\Delta_k^{t-1}}\cdot\Delta_k^t = 0 \end{cases}$
Adjust weights and bias
$w_{kj^*}(t+1) = w_{kj^*}(t) + \alpha_k\delta_k O$
$\theta_k(t+1) = \theta_k(t) + \beta_k\delta_k$
the total number of accuracies ≤ 0
No
Yes
Stop training

Fig. 9. ART2-based self-organizing supervised learning algorithm

	Previously-proposed method	Proposed method in this chapter
Success rate	55/79(69.6%)	72/79(91.1%)

Table 2. Comparison of the success rate of identifier area extraction

In the experiment of identifier recognition, the learning rate and the momentum are set to 0.4 and 0.3 for the two recognition algorithms, respectively. And, for ART2 algorithm generating nodes of the middle layer in the proposed algorithm, vigilance variables of two character types are set to 0.4.

When comparing the number of nodes of the middle layer between the two algorithms, the proposed algorithm creates more nodes than FCM-based RBF network, but via the comparison of the number of Epochs, it is known that the number of iteration of learning in the proposed algorithm is less than FCM-based RBF network. That is, the proposed algorithm improves the performance of learning. Also, comparing the success rate of recognition, it is able to be known that the proposed algorithm improves the performance of recognition compared with FCM-based RBF network. Failures of recognition in the proposed algorithm are incurred by the damage of shapes of individual identifiers in original images and the information loss of identifiers in the binarzation process.

	FCM-based RBF network		ART2-base self-organizing supervised learning algorithm	
	# of Epoch	# of success of recognition	# of Epoch	# of success of recognition
Alphabetic Characters(284)	236	240 (84.5%)	221	280 (98.5%)
Numeric Characters(500)	161	422 (84.4%)	151	487 (97.4%)

Table 3. Evaluation of recognition performance

4. Conclusion

This chapter proposes an automatic recognition system of shipping container identifiers using fuzzy-based noise removal method and ART2-based self-organizing supervised learning algorithm. In the proposed method, after detecting and removing noises from an original image by using a fuzzy method, areas of identifiers are extracted. In detail, the performance of identifier area extraction is improved by removing noises incurring errors using a fuzzy method based on the feature that the color of container identifiers is white or black on the whole. And, individual identifiers are extracted by applying the 8-directional contour tracking method to extracted areas of identifiers. Experiments using 79 container images show that 72 areas of identifiers and 784 individual identifiers are extracted successfully and 767 identifiers among the extracted are recognized by the proposed recognition algorithm. Failures of recognition in the proposed algorithm are incurred by the damage of shapes of individual identifiers in original images and the information loss of identifiers in the binarzation process.

5. References

Chen, Y.S. & Hsu, W.H. (1989). A systematic approach for designing 2-subcycle and pseudo 1-Subcycle parallel thinning algorithms, *Pattern Recognition*, Vol.22, No.3, (1989) pp. 267-282

ISO-6346, (1995). Freight Containers-Coding -Identification and Marking

Carpenter, G.A., Grossberg, S.J., Reynolds H. (1991). ARTMAP: Supervised Real-time Learning and Classification of Nonstationary Data by a Self-organizing Neural Network, *Neural Networks*, Vol.4, (1991) pp.565-588

Kim, K.B., Yun, H.W. (1999). A Study on Recognition of Bronchogenic Cancer Cell Image Using a Physiological Fuzzy Neural Networks, *Japanese Journal of Medical Electronics and Biological Engineering*, Vol.13, No.1, (1999) pp.39-43

Kim, K.B. (2003). The Identifier Recognition from Shipping Container Image by Using Contour Tracking and Self-Generation Supervised Learning Algorithm Based on Enhanced ART1, *Journal of Intelligent Information Systems*, Vol.9, No.3, (2003) pp. 65-80

Kim, K.B. (2004). Recognition of Identifiers from Shipping Container Images using Fuzzy Binarization and Neural Network with Enhanced Learning Algorithm, *Applied Computational Intelligence*, (2004) pp. 215-221

Kim, K.B., Kim, C. K. (2004). Performance Improvement of RBF Network using ART2 Algorithm and Fuzzy Logic System, *Lecture Notes in Artificial Intelligence*, LNAI 3339, Springer, (2004) pp.853-860

Kim, K.B., Lee, D.U., Sim, K.B. (2005). Performance Improvement of Fuzzy RBF Networks, *Lecture Notes in Computer Science*, LNCS 3610, Springer, (2005) pp.237-244

Nam, M.Y., Lim, E.K., Heo, N.S. & Kim, K.B. (2001). A Study on Character Recognition of Container Image using Brightness Variation and Canny Edge, *Proceedings of Korea Multimedia Society*, pp. 111-115

Vogl, T.P., Mangis, J.K., Zigler, A.K., Zink, W.T. & Alkon, D.L. (1998). Accelerating the convergence of the backpropagation method, *Biological Cybernetics*, Vol.59, (1998) pp. 256-264

12

Data mining with skewed data

Manoel Fernando Alonso Gadi Grupo Santander, Abbey, Santander Analytics
Milton Keynes
United Kingdom

Alair Pereira do Lago
Depart. de Ciencia de Computacao, Inst. de Matematica e Estatistica Universidade de Sao Paulo
Brazil

Jörn Mehnen
Decision Engineering Centre, Cranfield University Cranfield, Bedfordshire MK43 0AL
United Kingdom

In this chapter, we explore difficulties one often encounters when applying machine learning techniques to real-world data, which frequently show skewness properties. A typical example from industry where skewed data is an intrinsic problem is fraud detection in finance data. In the following we provide examples, where appropriate, to facilitate the understanding of data mining of skewed data. The topics explored include but are not limited to: data preparation, data cleansing, missing values, characteristics construction, variable selection, data skewness, objective functions, bottom line expected prediction, limited resource situation, parametric optimisation, model robustness and model stability.

1. Introduction

In many contexts like in a new e-commerce website, fraud experts start investigation procedures only after a user makes a claim. Rather than working reactively, it would be better for the fraud expert to act proactively before a fraud takes place. In this e-commerce example, we are interested in classifying sellers into legal customers or fraudsters. If a seller is involved in a fraudulent transaction, his/her license to sell can be revoked by the e-business. Such a decision requires a degree of certainty, which comes with experience. In general, it is only after a fraud detection expert has dealt with enough complains and enough data that he/she acquired a global understanding of the fraud problem. Quite often, he/she is exposed to a huge number of cases in a short period of time. This is when automatic procedures, commonly computer based, can step in trying to reproduce expert procedures thus giving experts more time to deal with harder cases. Hence, one can learn from fraud experts and build a model for fraud. Such a model requires fraud evidences that are commonly present in fraudulent behavior. One of the difficulties of fraud detection is that fraudsters try to conceal themselves under a “normal” behavior. Moreover, fraudsters rapidly change their modus operandi once it is discovered. Many fraud evidences are illegal and justify a more drastic measure against the fraudster. However, a single observed indicator is often not strong enough to be considered a proof and needs to be evaluated as one variable among others. All variables taken together can indicate high probability of fraud. Many times, these variables appear in the literature by the name of characteristics or features. The design of these characteristics to be used in a model is called characteristics extraction or feature extraction.

2. Data preparation

2.1 Characteristics extraction

One of the most important tasks on data preparation is the conception of characteristics. Unfortunately, this depends very much on the application (See also the discussions in Section 4 and 5). For fraud modelling for instance, one starts from fraud expert experience, determine significant characteristics as fraud indicators, and evaluates them. In this evaluation, one is interested in measuring how well these characteristics:

- covers (is present in) the true fraud cases;
- and how clearly they discriminate fraud from non-fraud behavior.

In order to cover as many fraud cases as possible, one may verify how many of them are covered by the characteristics set. The discrimination power of any of these characteristics can be evaluated by their odds ratio. If the probability of the event (new characteristics) in each of two compared classes (fraud and non-fraud in our case) are p_f (first class) and p_n (second class), then the odds ratio is:

$$OR = \frac{p_f/(1-p_f)}{p_n/(1-p_n)} = \frac{p_f(1-p_n)}{p_n(1-p_f)}.$$

An odds ratio equals to 1 describes the characteristics as equally probable in both classes (fraud and non-fraud). The more this ratio is greater/less than 1, the more likely this characteristic is in the first/second class than in the other one.

2.2 Data cleansing

In many manuals on best practice in model development, a chapter on data consistency checks, or data cleansing, is present. The main reason for this is to avoid waisting all the effort applied in the model development stage, because of data inconsistency invalidating the dataset in use.

Here we understand *data consistency checks* as being a set of expert rules to check whether a characteristic follows an expected behaviour. These expert rules can be based on expert knowledge or common sense. For example, a common error when filling in the date-of-birth section in a credit card application form is to put the current year instead of the year the person was actually born. In most countries an under sixteen year old can not have a credit card. Therefore, an easy way of checking this inconsistency is to simply calculate the applicant's age and check if it falls within a valid range. With more information available, more complex checks can be applied, such as, e.g. matching name with gender or street name with post code. In some cases the model developer has access to reports stored in Management Information Systems (MIS). If that is the case, it is a highly recommended idea to calculate key population indicators and compare these to portfolio reports. For example, in a credit card industry environment one can check the volume of applications; accept rate, decline rate and take-up rate

and others. It can also be useful to check the univariate distribution of each variable including the percentage of outliers, missing and miscellaneous values.
Having identified the characteristics that contain errors, the next step is to somehow fix the inconsistencies or minimise their impact in the final model. Here we list, in the form of questions, some good practices in *data cleansing* used by the industry that can sometimes improve model performance, increase generalisation power and finally, but no less important, make models less vulnerable to fraud and faults.

1. Is it possible to fix the errors by running some codes on the dataset? Sometimes wrong values have a one-to-one mapping to the correct values. Therefore, the best strategy is to make the change in the development dataset and to carry on with the development. It is important that these errors are fixed for the population the model will be applied to as well. This is because both developing and applying populations must be consistent, otherwise fixing the inconsistency would worsen the model performance rather than improving it;
2. Is a small number of attributes[1] (less than 5%) impacting only few rows (less than 5%)? In this case, one can do a bivariate analysis to determine if it is possible to separate these values into a default (or fault) group. Another option is to drop the rows. However, this tactic might turn out to be risky (see section about missing values);
3. Is the information value of the problematic attribute(s) greater than for the other attributes combined? Consider dropping this characteristic and demand fixing;
4. Is it possible to allow outliers? Simply dropping them might be valid if there are few or there are invalid values. Change their values to the appropriate boundary could also be valid. For example, if an acceptable range for *yearly income* is [1,000;100,000] MU[2] and an applicant has a yearly income of 200,000 MU then it should be changed to 100,000 MU. This approach is often referred to as truncated or censored modelling Schneider (1986).
5. Finally, in an industry environment, when an MIS is available, one can check for the acceptance rate or number of rows to be similar to the reports? It is very common for datasets to be corrupted after transferring them from a Mainframe to Unix or Windows machines.

3. Data skewness

A dataset for modelling is perfectly balanced when the percentage of occurrence of each class is $100/n$, where n is the number of classes. If one or more classes differ significantly from the others, this dataset is called skewed or unbalanced. Dealing with skewed data can be very tricky. In the following sections we explore, based on our experiments and literature reviews, some problems that can appear when dealing with skewed data. Among other things, the following sections will explain the need for stratified sampling, how to handle missing values carefully and how to define an objective function that takes the different costs for each class into account.

[1] possible values of a given characteristic
[2] MU = Monetary Units.

3.1 Missing values

Missing values are of little importance when dealing with balanced data, but can become extremely harmful or beneficial when dealing with skewed data. See how the example below looks harmful at first glance, but indeed exposes a very powerful characteristic.

Table 1 shows an example of a characteristic called Transaction Amount. By looking at the first line of the table one may conclude that the number of missing values is small (1.98%) and decide not to investigate any further. Breaking it down into fraudulent and legitimate transactions, one can see that 269 (32.5%) data items whose values are missings are frauds, which is nearly 9 times bigger than the overall fraud rate in our dataset (1,559/41,707 = 3.74% see Table 2).

Population	# transaction	# missing	% missing
Total	41707	825	1.98%
Fraud	1559	269	17.26%
Legitimate	40148	556	1.39%

Table 1. Fraud/legitimate distribution

Investigating even further, by analysing the fraud rates by ranges as shown in table 2, one can see that the characteristic being analysed really helps to predict fraud; on the top of this, missing values seem to be the most powerful attribute for this characteristic.

Trans. amount	# frauds	# trans.	Fraud rate
Missings	269	825	32.61%
0,100	139	14639	0.95%
101,500	235	10359	2.27%
501,1000	432	8978	4.81%
1001,5000	338	4834	6.99%
5001,+inf	146	2072	7.05%
Total	1559	41707	3.74%

Table 2. Transaction amount bivariate

When developing models with balanced data, in most cases one can argue that it is good practice to avoid giving prediction to missing values (as a separate attribute or dummy), especially, if this attribute ends up with dominating the model. However, when it comes to unbalanced data, especially with fraud data, some specific value may have been intentionally used by the fraudster in order to bypass the system's protection. In this case, one possible explanation could be a system failure, where all international transaction are not being correctly currency converted when passed to the fraud prevention system. This loophole may have been found by some fraudster and exploited. Of course, this error would have passed unnoticed had one not paid attention to any missing or common values in the dataset.

4. Derived characteristics

New or derived characteristics construction is one of, if not the, most important part of modelling. Some important phenomena mapped in nature are easily explained using derived variables. For example, in elementary physics speed is a derived variable of space over time. In data mining, it is common to transform date of birth into age or, e.g., year of study into primary, secondary, degree, master, or doctorate. Myriad ways exist to generate derived characteristics. In the following we give three typical examples:

1. *Transformed characteristics:* transform characteristics to gain either simplicity or generalisation power. For example, date of birth into age, date of starting a relationship with a company into time on books, and years of education into education level;
2. *Time series characteristics:* a new characteristic built based on a group of historical months of a given characteristic. Examples are average balance of a bank account within the last 6 months, minimum balance of a bank account within the last 12 months, and maximum days in arrears[3] over the last 3 months;
3. *Interaction:* variable combining two or more different characteristics (of any type) in order to map interesting phenomena. For example, average credit limit utilization = average utilization / credit limit.

5. Categorisation (grouping)

Categorisation (discretising, binning or grouping) is any process that can be applied to a characteristic in order to turn it into categorical values Witten & Franku (2005). For example, let us suppose that the variable *age* ranges from 0 to 99 and all values within this interval are possible. A valid categorisation in this case could be:

1. category *1*: if age is between 1 and 17;
2. category *2*: if age is between 18 and 30;
3. category *3*: if age is between 31 and 50;
4. category *4*: if age is between 51 and 99.

Among others, there are three main reasons for categorising a characteristic: firstly, to increase generalisation power; secondly, to be able to apply certain types of methods, such as, e.g. a Generalised Linear Model[4] (GLM) Witten & Franku (2005), or a logistic regression using Weight of Evidence[5] (WoE) formulations Agterberg et al. (1993); thirdly, to add stability to the model by getting rid of small variations causing noise. Categorisation methods include:

1. *Equal width*: corresponds to breaking a characteristic into groups of equal width. In the *age* example we easily break age into 5 groups of 20 decimals in each: 0-19, 20-39, 40-59, 60-79, 80-99.
2. *Percentile*: this method corresponds to breaking the characteristic into groups of equal volume, or percentage, of occurrences. Note that in this case groups will have different widths. In some cases breaking a characteristic into many groups may not be possible because occurrences are concentrated. A possible algorithm in pseudo code to create percentile groups is:

```
Nc <- Number of categories to be created
Nr <- Number of rows
Size <- Nr/Nc
Band_start [0] <- Minimum (Value (characteristic[0..Nr]))
//Dataset needs to be sorted by the characteristic to be grouped
For j = 1 .. Nc {
For i = 1 .. Size {
```

[3] Arrears is a legal term for a type of debt which is overdue after missing an expected payment.

[4] One example of this formulation is logistic regression using dummy input variables

[5] This formulation replaces the original characteristic grouped attribute for its weight of evidence

```
Value_end <- Value(characteristic[i+Nc*Size])
}
Band_end[j] <- Value_end
Band_start[j+1] <- Value_end
}
```

3. *Bivariate grouping*: this method corresponds to using the target variable to find good breaking points for the ranges of each group. It is expected that, in doing so, groups created using a bivariate process have a lower drop in information value, whilst it can improve the generalisation by reducing the number of attributes. One can do this in a spreadsheet by recalculating the odds and information value every time one collapses neighbouring groups with either similar odds, non-monotonic odds or a too small population percentage.

Next, we present one possible process of grouping the characteristic *age* using a bivariate grouping analysis. For visual simplicity the process starts with groups of equal width, each containing 10 units (see Table 3). The process consists of eliminating intervals without monotonic odds, grouping similar odds and guaranteeing a minimal percentage of individuals in each group.

Age	goods	bads	Odds	%tot
0-9	100	4	25.00	1.83%
10-19	200	10	20.00	3.70%
20-29	600	50	12.00	11.46%
30-39	742	140	5.30	15.55%
40-49	828	160	5.18	17.42%
50-59	1000	333	3.00	23.50%
60-69	500	125	4.00	11.02%
70-79	300	80	3.75	6.70%
80-89	200	100	2.00	5.29%
90-99	100	100	1.00	3.53%
Total	4570	1102	4.15	100.00%

Table 3. Age bivariate step 1/4

Age	goods	bads	Odds	%tot
0-9	100	4	25.00	1.83%
10-19	200	10	20.00	3.70%
20-29	600	50	12.00	11.46%
30-39	742	140	5.30	15.55%
40-49	828	160	5.18	17.42%
50-79	**1800**	**538**	**3.35**	**41.22**%
80-89	200	100	2.00	5.29%
90-99	100	100	1.00	3.53%
Total	4570	1102	4.15	100.00%

Table 4. Age bivariate step 2/4

The result of the first step, eliminating intervals without monotonic odds can be seen in Table 4. Here bands 50-59 (odds of 3.00), 60-69 (odds of 4.00) and 70-79 (odds of 3.75) have been merged, as shown in boldface. One may notice that merging bands 50-59 and 60-69 would

Age	goods	bads	Odds	%tot
0-9	100	4	25.00	1.83%
10-19	200	10	20.00	3.70%
20-29	600	50	12.00	11.46%
30-49	**1575**	**300**	**5.23**	**32.97%**
50-79	1800	538	3.35	41.22%
80-89	200	100	2.00	5.29%
90-99	100	100	1.00	3.53%
Total	4570	1102	4.15	100.00%

Table 5. Age bivariate step 3/4

result in a group with odds of 3.28; hence resulting in the need to merge with band 70-79 to yield monotonic odds.

By using, for example, 0.20 as the minimum allowed odds difference, Table 5 presents the result of step two where bands 30-39 (odds of 5.30) and 40-49 (odds of 5.18) have been merged. This is done to increase model stability. One may notice that odds retrieved from the development become expected odds in a future application of the model. Therefore, these values will vary around the expectation. By grouping these two close odds, one tries to avoid that a reversal in odds may happen by pure random variation.

Age	goods	bads	Odds	%tot
0-19	**300**	**14**	**21.43**	**5.54%**
20-29	600	50	12.00	11.46%
30-49	1575	300	5.23	32.97%
50-79	1800	538	3.35	41.22%
80-89	200	100	2.00	5.29%
90-99	100	100	1.00	3.53%
Total	4570	1102	4.15	100.00%

Table 6. Age bivariate step 4/4

For the final step, if we assume 2% to be be the minimum allowed percentage of the population in each group. This forces band 0-9 (1.83% of total) to be merged with one of its neighbours; in this particular case, there is only the option to merge with band 10-19. Table 6 shows the final result of the bivariate grouping process after all steps are finished.

6. Sampling

As computers become more and more powerful, sampling, to reduce the sample size for model development, seems to be losing attention and importance. However, when dealing with skewed data, sampling methods remain extremely important Chawla et al. (2004); Elkan (2001). Here we present two reasons to support this argument.

First, to help to ensure that no over-fitting happens in the development data, a sampling method can be used to break the original dataset into training and holdout samples. Furthermore, a stratified sampling can help guarantying that a desirable factor has similar percentage in both training and holdout samples. In our work Gadi et al. (2008b), for example, we executed a random sampling process to select multiple splits of 70% and 30%, as training and holdout samples. However, after evaluating the output datasets we decided to redo the sampling process using stratified sampling by fraud/legitimate flag.

Second, to improve the model prediction, one may apply an over- or under- sampling process to take the different cost between classes into account. Cost-sensitive procedure Elkan (2001) replicates (oversampling) the minority (fraud) class according to its cost in order to balance different costs for false positives and false negatives. In Gadi et al. (2008a) we achieved interesting results by applying a cost-sensitive procedure.
Two advantages of a good implementation of a cost-sensitive procedure are: first, it can enable changes in cut-off to the optimal cut-off, For example, in fraud detection, if the cost tells one, a cost-sensitive procedure will consider a transaction with as little as 8% of probability of fraud as a potential fraud to be investigated; second, if the cost-sensitive procedure considers cost per transaction, such an algorithm may be able to optimise decisions by considering the product [probability of event] x [value at risk], and decide on investigating those transactions in which this product is bigger.

7. Characteristics selection

Characteristics selection, also known as feature selection, variable selection, feature reduction, attribute selection or variable subset selection, is commonly used in machine learning and statistical techniques to select a subset of relevant characteristics for the building of more robust models Witten & Franku (2005).
Decision trees do characteristics selection as part of their training process when selecting only the most powerful characteristics in each subpopulation, leaving out all weak or highly correlated characteristics. Bayesian nets link different characteristics by cause and effect rules, leaving out non-correlated characteristics Charniak (1991). Logistic Regression does not use any intrinsic strategy for removing weak characteristics; however, in most implementations methods such as forward, backward and stepwise are always available. In our tests, we have applied a common approach in the bank industry that is to consider only those characteristics with information value greater than a given percentage threshold.

8. Objective functions

When defining an objective function, in order to compare different models, we found in our experiments that two facts are especially important:

1. We have noticed that academia and industry speak in different languages. In the academic world, measures such as Kolmogorov Smirnov (KS) Chakravarti et al. (1967) or Receiver Operating Characteristic (ROC curve) Green & Swets (1966) are the most common; in industry, on the other hand, rates are more commonly used. In the fraud detection area for example it is common to find measures such as hit rate (confidence) and detection rate (cover). Hit rate and detection rate are two different dimensions and they are not canonical. To optimise a problem with an objective having two outcomes is not a simple task Trautmann & Mehnen (2009). In our work in fraud detection we avoided this two-objective function by calculating one single outcome value: the total cost of fraud;

2. In an unbalanced environment it is common to find that not only the proportion between classes differs, but also the cost between classes. For example, in the fraud detection environment, the loss by fraud when a transaction is fraudulent is much bigger than the cost to call a customer to confirm whether he/she did or did not do the transaction.

9. Bottom line expected prediction

The problem of finding the best model can be computationally expensive, as there are many parameters involved in such a search. For this reason, it is very common for model developers to get satisfied with suboptimal models. A question equally difficult to answer, in general, is how far we are from an optimum. We do not intend to respond to this question here; what we want to address is a list of ways to help the model developer to estimate a minimum acceptable performance before getting close to the end of the model development. In our fraud analysis we found two good options for estimating a bottom line for expected suboptimal cost: a first option could be the cost resulting from a Naïve Bayes model. It is important to notice that Naïve Bayes does not need any grouping, characteristics selection or parameter tuning; a second option could be to consider the cost from a first "quick and dirty" model developed using the method chosen by the model developer.

10. Limited resource situation

Many real-world application present limited resource problems. This can make the decision of what is the best model different compared to a model without restrictions. In a hospital, for example, there may be a limited number of beds for patients; in a telephone costumer service facility, there may be a limited number of attendants; in the fraud detection world the number of people available to handle manual transactions is in general fixed; and the number of transactions each member of fraud detection can handle per day is also fixed due to practical reasons. In such applications, being aware of the capacity rate becomes very important. It is also extremely important for the model outcome to indicate the probability[6] of the event rather than providing a simple yes/no response. By having the outcome as a probability, models can be compared using for example, cutoffs that keep the selecting rate equal to the capacity rate. In fraud detection, comparing models detection rate and hit rate fixing for example 1000 transaction to be investigated.

11. Parametric optimisation

Once we have the data and the optimisation criteria, the following questions have to be answered:

> Which classification method is recommended for producing the best model for any given application?
>
> Which parameter set should be used?

For instance, we can apply classification methods such as: Neural Networks (NN), Bayesian Networks (BN), Naïve Bayes (NB), Artificial Immune Systems (AIS) and Decision Trees (DT), Support Vector Machines (SVM), Logistic Regression and others. In fact, there is not a final and unique answer to this first question. Support Vector Machines, for instance, is known to be very effective for data with a very large number of characteristics and is reported to perform well in categorisation problems in Information Retrieval. However, our experience with SVM on fraud data did not meet our expectations. For many parameter sets, the method did not even converge to a final model and this behaviour for unbalanced data is reported to not be uncommon.

[6] Or be possible to transform it into a probability.

In order to assess methods many factors can be used including the chosen optimisation criteria, scalability, time for classification and time spent in training, and sometimes more abstract criteria as time to understand how the method works. Most of the time, when a method is published, or when an implementation is done, the method depends on parameter choices that may influence the final results significantly. Default parameters, in general, are a good start. However, most of the time, they are far from producing the best model. This comprises with our experience with many methods in many different areas of Computer Science. This is particular true for classification problems with skewed data.

Quite often we see comparisons against known methods where the comparison is done by applying a special parameter variation strategy (sometimes a parameter optimisation) for the chosen method while not fairly conducing the same procedure for the other methods. In general, for the other methods, default parameters, or a parameter set published in some previous work is used. Therefore, it is not a surprise that the new proposed method wins. At a first glance, the usage of the default parameter set may seem to be fair and this bias is often reproduced in publications. However, using default sets can be biased by the original training set and, thus, not be fair.

Parameter optmisation takes time and is rarely conduced. For a fair comparison, we argue that one has to fine tune the parameters for all compared method. This can be done, for instance, via an exhaustive search of the parameter space if this search is affordable, or some kind of sampling like in Genetic Algorithm (GA)[7] (see Figure 1). Notice, that the final parameter set cannot be claimed to be optimal in this case.

Unfortunately, this sampling procedure is not as easy as one may suppose. There is not a single best universal optimisation algorithm for all problems (No Free Lunch theorem - Wolpert and Macready 1997 Wolpert & Macready (1997)). Even the genetic algorithm scheme as shown in Figure 1 might require parameter adjustment. According to our experience, we verified that a simple mistake in the probability distribution computation may drive the results to completely different and/or misleading results. A good genetic algorithm requires expertise, knowledge about the problem that should be optimised by the GA, an intelligent design, and resources. The more, the better. These considerations also imply that comparisons involving methods with suboptimal parameter sets depend very much on how well each parameter space sampling was conduced.

12. Robustness of parameters

After the parameter optimisation has been conducted, it can be advantageous or desirable to have the optimised parameters independent from the training set, i.e. they can be applied to different datasets of the same problem. In this case we can call this parameter set *robust*.

When the parameter are not robust, the optimisation process is not as strong as expected since the obtained optimised parameter set has no or little generalisation power. In this case, in our experiments, we found that it is a good approach to sacrifice some prediction power in order to gain robustness in the parameter set. Note that a procedure using n-fold cross validation could lead to a parameter set that is more independent from a dataset. However, we choose to present a different approach which also generates robust parameter sets with more control of what is happening during the process. This procedure is based on repeated sampling from the development dataset into training and holdout samples. Then, we applied parameter

[7] One could also use some kind of Monte Carlo, Grid sampling or Multiresolution alternatives.

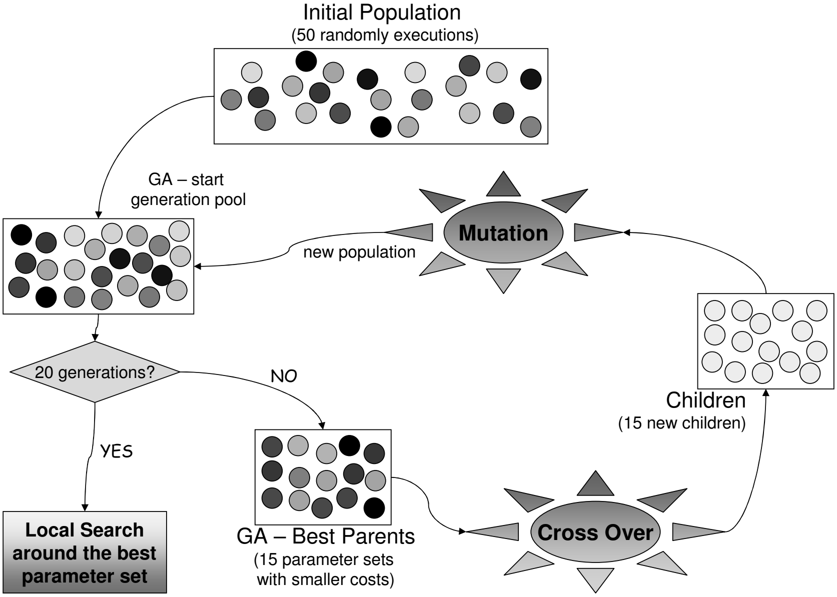


Fig. 1. Genetic Algorithm for parameters optimisation. We start with an initial pool of e.g. 50 random individuals having a certain fitness, followed by e.g. 20 Genetic Algorithm (GA) generations. Each GA generation combines two randomly selected candidates among the best e.g. 15 from previous generation. This combination performs: crossover, mutation, random change or no action for each parameter independently. As the generation goes by, the chance of no action increases. In the end, one may perform a local search around the optimised founded by GA optimisation. Retrieved from Gadi et al. Gadi et al. (2008b).

optimisation and choose the set of parameters which is the best in average over all splits at the same time.

In our work, in order to rewrite the optimisation function that should be used in a GA algorithm, we have used a visualization procedure with computed costs for many equally spaced parameter sets in the parameter space. After having defined a good optimisation function, due to time constraints, we did not proceed with another GA optimisation, but we reused our initial runs used in the visualization, with the following kind of *multiresolution optimisation* Kim & Zeigler (1996) (see Figure 2):

- we identified those parameters that have not changed, and we frozen these values for these respective parameters;
- with any other parameter, we screened the 20 best parameter sets for every split and identified a reasonable range;
- for all non-robust parameters, we chose an integer step *s* so the search space did not explode;

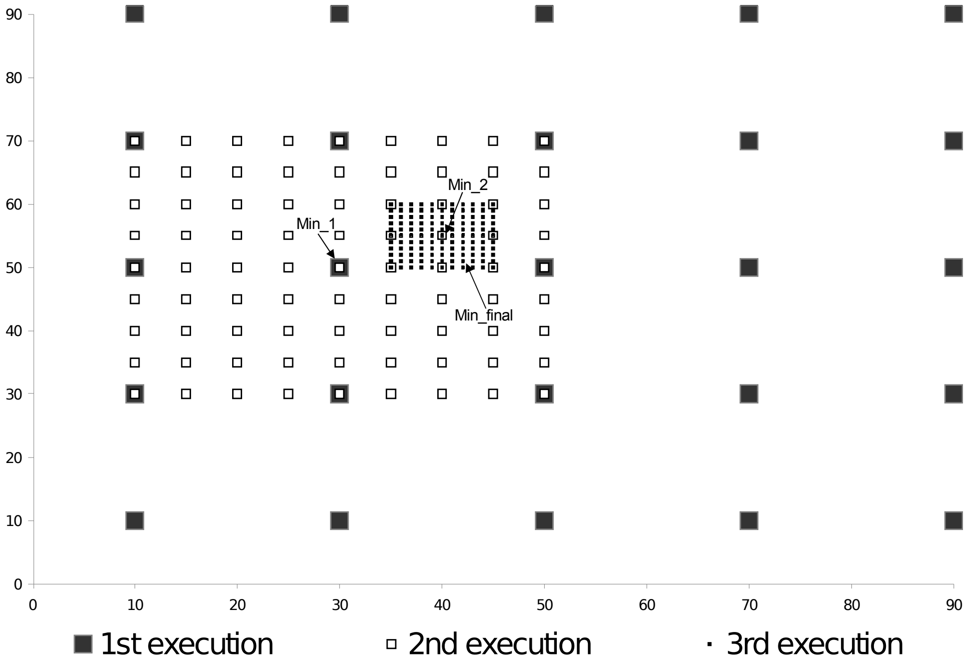


Fig. 2. An example of the multiresolution optimisation that was applied in order to find robust parameters. In this example one can see two parameters in the search space and three steps of this multiresolution optimisation. For the parameter represented in horizontal line, the search space in first step ranges from 10 to 90 with step size 20 and the minimum was found for 30. In the second step, the scale ranges from 10 to 50 with step size 5 and the minimum was found for 40. In third step, it ranges from 35 to 45, with step size 1, which is equivalent to a local exhaustive search in this neighborhood. Retrieved from Gadi et al. Gadi et al. (2008a).

- we evaluated the costs for all possible combinations according to the search space defined above and found the parameter set P that brings the minimum average cost among all the different used splits;
- if the parameter set P was at the border of the search space, we shifted this search space by one step in the direction of this border and repeated last step until we found this minimum P in the inner area of the search space;
- we zoomed the screening in on the neighborhood of P, refined steps *s*, and repeated the process from then on until no refinement was possible.

13. Model stability

In industry, generally the aim of modelling is to apply a model to a real situation and to generate profit, either by automating decision making where a model was not previously available or replacing old models by a new and improved one. For doing so, most model development processes rely on past information for their training. Therefore, it is very important to be able to assess whether or not a model is still fit for propose when it is in use, and to have a set of actions to expand the model's life span. In this section we explore advantages of using out-of-time samples, monitoring reports, stability by vintage, vintage selection and how to deal with different scales over time.

13.1 Out-of-time:

an Out-Of-Time sample (OOT) is any sample of the same phenomena used in the model development that is not in the development window[8], historic vintages or observation point selected for development. In most cases in reality a simple split of the development sample into training and testing data cannot identify a real over-fitting of the model Sobehart et al. (2000). Therefore, the most appropriated approach to identify this change is either to select a vintage or observation point posterior to the development window or select this previously to the development window. The second approach gives the extra advantage of using the most up-to-date information for the development.

13.2 Monitoring reports:

the previous action, OOT, should be best done before the actual model implementation; after that, it becomes important to evaluate whether the implemented model still delivers a good prediction. For this purpose, it is crucial to create a set of period based monitoring reports to track the model's performance and stability over time.

13.3 Stability by vintage:

stability by vintage corresponds to breaking the development sample down by time within the development window and evaluate the model's performance in all of the different periods within the data. For example, if one has information collected from January 08 to December 08, a good stability by vintage analysis would be to evaluate the model's performance over each month of 2008. This tends to increase the chance of a model to be stable after its implementation.

13.4 Vintage selection:

many phenomena found in nature, and even in human behaviour, repeat themselves year after year in a recurrent manner; this is known as *seasonality*. Choosing a development window from a very atypical month of the year can be very misleading; in credit cards, for example, choosing only December as the development time window can lead to overestimation of expected losses since this is the busiest time of the year. Two approaches intend to mitigate this problem. Both approaches are based on selecting the broadest development window possible. One common window size is 12 months, allowing the window to cover the whole year. Please notice, there is no need to fix the start of the window to any particular month. The first approach corresponds to simply develop the model with this pool of observation points; it is expected for the model to be an average model that will work throughout the year. A second approach is to introduce a characteristic indicating the "month of the year" the information was collected from, or any good combination of it, and then to develop the model. As a result, one would expect a model that adjusts better to each observation point in the development window.

[8] Here we understand development window as being the period from where the training samples were extracted. This can be hours, days, months, years, etc.

13.5 Different scale over time:

Another common problem applies to the situation where characteristic values fall outside the training sample boundaries or some unknown attributes occur. To reduce the impact of this problem, one can always leave the groups with the smallest and biggest boundaries as negative infinite and positive infinite, respectively, for example, changing [0,10];[11,20];[21,30] to $]-\infty$,10];[11,20];[21,$+\infty$[. Furthermore, undefined values could always be assigned to a default group. For example, if for a numeric characteristic a non-numeric value ocurrs it could be assigned to a default group.

14. Final Remarks

This work provided a brief introduction to pratical problem solving for machine learning with skewed data sets. Classification methods are generally not designed to cope with skewed data, thus, various action have to be taken when dealing with imbalanced data sets. For a reader looking for more information about the field we can recommend a nice editorial by Chawla et al. **?** and three conference proceedings Chawla et al. (2003); Dietterich et al. (2000); Japkowicz (2000). In addition, good algorithm examples can be found in Weka Witten & Franku (2005) and SAS Delwiche & Slaughter (2008).

Perhaps, most solutions that deal with skewed data do some sort of sampling (e.g: with undersampling, oversampling, cost sensitive training Elkan (2001), etc.). These contributions are effective Gadi et al. (2008a) and quite well known nowadays.

This text provides recommendations for practitioners who are facing data mining problems due to skewed data.

Details on the experiments can be found at Gadi et al. (2008b) and Gadi et al. (2008a), which presents an application of Artificial Immune Systems on credit card fraud detection.

Finally, another subject explored in this work was the importance of parametric optimization for chosing a good classification method for skewed data. We also suggested a proceedure for parametric optimization.

15. References

Agterberg, F. P., Bonham-Carter, G. F., Cheng, Q. & Wright, D. F. (1993). Weights of evidence modeling and weighted logistic regression for mineral potential mapping, pp. 13–32.

Chakravarti, I. M., Laha, R. G. & Roy, J. (1967). *Handbook of Methods of Applied Statistics*, Vol. I, John Wiley and Sons, USE.

Charniak, E. (1991). Bayesians networks without tears, *AI Magazine* pp. 50 – 63.

Chawla, N. V., Japkowicz, N. & Kotcz, A. (2004). Special issue on learning from imbalanced data sets, *SIGKDD Explorations* **6**(1): 1–6.

Chawla, N. V., Japkowicz, N. & Kotcz, A. (eds) (2003). *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Data Sets*.

Delwiche, L. & Slaughter, S. (2008). *The Little SAS Book: A Primer*, SAS Publishing.

Dietterich, T., Margineantu, D., Provost, F. & Turney, P. (eds) (2000). *Proceedings of the ICML'2000 Workshop on Cost-Sensitive Learning*.

Elkan, C. (2001). The foundations of cost-sensitive learning, *IJCAI*, pp. 973–978.
URL: *citeseer.ist.psu.edu/elkan01foundations.html*

Gadi, M. F. A., Wang, X. & Lago, A. P. d. (2008a). Comparison with parametric optimization in credit card fraud detection, *ICMLA '08: Proceedings of the 2008 Seventh International*

Conference on Machine Learning and Applications, IEEE Computer Society, Washington, DC, USA, pp. 279–285.

Gadi, M. F., Wang, X. & Lago, A. P. (2008b). Credit card fraud detection with artificial immune system, *ICARIS '08: Proceedings of the 7th international conference on Artificial Immune Systems*, Springer-Verlag, Berlin, Heidelberg, pp. 119–131.

Green, D. M. & Swets, J. A. (1966). *Signal Detection Theory and Psychophysics*, John Wiley and Sons.
URL: *http://www.amazon.co.uk/exec/obidos/ASIN/B000WSLQ76/citeulike00-21*

Japkowicz, N. (ed.) (2000). *Proceedings of the AAAI'2000 Workshop on Learning from Imbalanced Data Sets*. AAAI Tech Report WS-00-05.

Kim, J. & Zeigler, B. P. (1996). A framework for multiresolution optimization in a parallel/distributed environment: simulation of hierarchical gas, *J. Parallel Distrib. Comput.* **32**(1): 90–102.

Schneider, H. (1986). *Truncated and censored samples from normal populations*, Marcel Dekker, Inc., New York, NY, USA.

Sobehart, J., Keenan, S. & Stein, R. (2000). Validation methodologies for default risk models, pp. 51–56.
URL: *http://www.moodyskmv.com/research/files/wp/p51p56.pdf*

Trautmann, H. & Mehnen, J. (2009). Preference-based pareto optimization in certain and noisy environments, *Engineering Optimization* **41**: 23–38.

Witten, I. H. & Franku, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*, Elsevier.

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization, *Evolutionary Computation, IEEE Transactions on* **1**(1): 67–82.
URL: *http://dx.doi.org/10.1109/4235.585893*

13

Scaling up instance selection algorithms by dividing-and-conquering

Aida de Haro-García, Juan Antonio Romero del Castillo
and Nicolás García-Pedrajas
University of Córdoba
Spain

1. Introduction

The overwhelming amount of data that is available nowadays in any field of research poses new problems for machine learning methods. This huge amount of data makes most of the existing algorithms inapplicable to many real-world problems. Two approaches have been used to deal with this problem: scaling up machine learning algorithms and data reduction. Nevertheless, scaling up a certain algorithm is not always feasible. On the other hand, data reduction consists of removing from the data missing, redundant and/or erroneous data to get a tractable amount of data. The most common methods for data reduction are instance selection and feature selection.

However, these algorithms for data reduction have the same scaling problem they are trying to solve. For example, in the best case, most existing instance selection algorithms are $O\left(n^2\right)$, n being the number of instances. For huge problems, with hundreds of thousands or even millions of instances, these methods are not applicable. The same happens with feature selection algorithms.

The alternative is scaling up the machine learning algorithm itself. In the best case, this is an arduous task, and in the worst case and impossible one. In this chapter we present a new paradigm for scaling up machine learning algorithms based on the philosophy of divide-and-conquer. One natural way of scaling up a certain algorithm is dividing the original problem into several simpler subproblems and applying the algorithm separately to each subproblem. In this way we might scale up instance selection dividing the original dataset into several disjoint subsets and performing the instance selection process separately on each subset. However, this method does not work well, as the application of the algorithm to a subset suffers from the partial knowledge it has of the dataset. However, if we join this divide-and-conquer approach with the basis of the construction of ensembles of classifiers, the combination of weak learners into a strong one, we obtain a very powerful and fast method, applicable to almost any machine learning algorithm. This method can be applied in different ways. In this chapter we propose two algorithms, recursive divide-and-conquer and democratization, that are able to achieve very good performance and a dramatic reduction in the execution time of the instance selection algorithms.

We will describe these methods and we will show how they can achieve very good results when applied to instance selection. Furthermore, the methodology is applicable to other machine learning algorithms, such as feature selection and cluster analysis.

Instance selection (Liu & Motoda, 2002) consists of choosing a subset of the total available data to achieve the original purpose of the data mining application as if the whole data were used. Different variants of instance selection exist. Many of the approaches are based on some form of sampling (Cochran, 1997) (Kivinen & Mannila, 1994). There are other more modern methods that are based on different principles, such as, Modified Selective Subset (MSS) (Barandela et al., 2005), entropy-based instance selection (Son & Kim, 2006), Intelligent Multiobjective Evolutionary Algorithm (IMOEA) (Chen et al., 2005), and LVQPRU method (Li et al., 2005).

The problem of instance selection for instance based learning can be defined as (Brighton & Mellish, 2002) "the isolation of the smallest set of instances that enable us to predict the class of a query instance with the same (or higher) accuracy than the original set". It has been shown that different groups of learning algorithms need different instance selectors in order to suit their learning/search bias (Brodley, 1995). This may render many instance selection algorithm useless, if their philosophy of design is not suitable to the problem at hand.

We can distinguish two main models of instance selection (Cano et al., 2003): instance selection as a method for prototype selection for algorithms based on prototypes (such as *k*-Nearest Neighbors) and instance selection for obtaining the training set for a learning algorithm that uses this training set (such as decision trees or neural networks). This chapter is devoted to the former methods.

Regarding complexity, in the best case, most existing instance selection algorithms are of efficiency $O\left(n^2\right)$, *n* being the number of instances. For huge problems, with hundreds of thousands or even millions of instances, these methods are not applicable. Trying to develop algorithms with a lower efficiency order is likely to be a fruitless search. Obtaining the nearest neighbor of a given instance is $O(n)$. To test whether removing an instance affects the accuracy of the nearest neighbor rule, we must measure the effect on the other instances of the absence of the removed one. Measuring this effect involves recalculating, directly or indirectly, the nearest neighbors of the instances. The result is a process of $O\left(n^2\right)$. In this way, the attempt to develop algorithms of an efficiency order below this bound is not very promising.

Thus, the alternative is reducing the size *n* of the set to which instance selection algorithms are applied. In the construction of ensembles of classifiers the problem of learning from huge datasets has been approached by means of learning many classifiers from small disjoint subsets (Chawla et al., 2004). In that paper, the authors showed that it is also possible to learn an ensemble of classifiers from random disjoint partitions of a dataset, and combine predictions from all those classifiers to achieve high classification accuracies. They applied their method to huge datasets with very good results. Furthermore, the usefulness of applying instance selection to disjoint subsets has also been shown in (García-Pedrajas et al., 2009). In that work, a cooperative evolutionary algorithm was used. The training set was divided into several disjoint subsets and an evolutionary algorithm was performed on each subset of instances. The fitness of the individuals was evaluated only taking into account the instances in the subset. To account for the global view needed by the algorithm a global population was used. This method is scalable to medium/large problems but cannot be applied to huge problems. Zhu & Wu (2006) also used disjoint subsets in a method for ranking representative instances.

Following this idea, we will present in this chapter two approaches for scaling up instance selection algorithms that are based on a divide-and-conquer approach. The presented methods are able to achieve very good performance with a drastic reduction in the time needed for the execution of the algorithms. The general idea underlying this work is dividing the original dataset into subsets and performing the instance selection process in each subset separately. Then, we must find a method for combining the separate applications of the instance selection algorithm to a final global result.

The rest of this paper is organized as follows: Section 2 revised some related work; Section 3 describes in depth our proposal; Section 4 shows the experiments performed with our methods; and finally Section 5 states the conclusions of our work.

2. Related work

As stated in the previous section, scaling up instance selection algorithms is a very relevant issue. The usefulness of applying instance selection to disjoint subsets has also been shown in (García-Pedrajas et al., 2009). In this work a cooperative evolutionary algorithm is used. Several evolutionary algorithms are performed on disjoint subsets of instances and a global population is used to account for the global view. This method is scalable to medium/large problems but cannot be applied to huge problems.

There are not many previous works that have dealt with instance selection for huge problems. Cano et al. (2005) proposed an evolutionary stratified approach for large problems. Although the algorithm shows very good performance, it is still too computationally expensive for huge datasets. Kim & Oommen (2004) proposed a method based on a recursive application of instance selection to smaller datasets.

In a recent paper, De Haro-García and García-Pedrajas (2009) showed that the application of a recursive divide-and-conquer approach is able to achieve a good performance while attaining a dramatic reduction in the execution time of the instance selection process.

3. Scaling up instance selection algorithms using divide-and-conquer philosophy

As stated in the previous section, scaling up instance selection algorithms is a very relevant issue. In this section we will discuss our proposals based on a divide-and-conquer approach. The two methods aim a general objective of scaling up instance selection algorithms. However, individually they have two different aims. The first one, based on recursively using the principle of divide-and-conquer, has as main goal the reduction of the time needed by the instance selection process. In this way, it trades time for performance, allowing a small increase on the testing error achieved by the algorithms. The second one, based on combining the divide-and-conquer approach with principles from ensembles of classifiers, has as special objective reducing the time of the algorithms, but keeping their performance as much as possible.

3.1 Recursive divide-and-conquer approach

As we have said, in the best case, most existing instance selection algorithms are of efficiency $O\left(n^2\right)$. For huge problems, with hundreds of thousands or even millions of instances, these methods are not applicable. Trying to develop algorithms with a lower efficiency order is likely to be a fruitless search.

Following the divide-and-conquer philosophy, we can develop a methodology based on applying the instance selection algorithm to subsets of the whole training set. A simple approach consists of using a stratified random sampling (Liu & Motoda, 2002) (Cano et al., 2005), where the original dataset is divided into many disjoint subsets, and then apply instance selection over each subset independently. However, due to the fact that to select the nearest neighbor of an instance we need to know the whole dataset, this method is not likely to produce good results. In fact, in practice its performance is poor. However, the divide-and-conquer principle of this method is an interesting idea for scaling up instance selection algorithms. Furthermore, divide-and-conquer methodology has the additional advantage that we can adapt the size of the subproblems to the available resources.

Following this philosophy we start performing a partition of the dataset and then applying the instance selection algorithm to every subset independently. This step is able to be performed fast and obtains good results in terms of testing error. However, its results in terms of storage reduction are poor. To avoid this drawback we apply this method recursively. After an instance selection step is performed the remaining instances are rejoined to obtain again subsets of approximately the same size and the instance selection process is repeated. Fig. 1 shows and outline of the process.

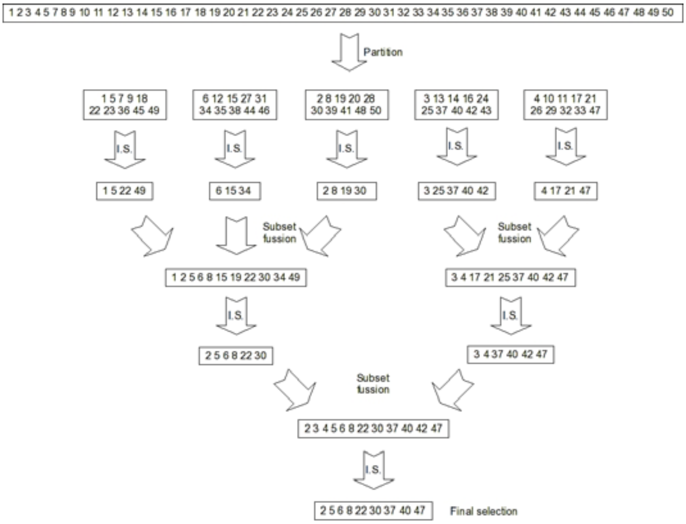


Fig. 1. Outline of recursive divide-and-conquer instance selection method

This method is applicable to any instance selection algorithm, as the instance selection algorithm is a parameter of the method. More formally, first, our method divides the whole training set, T, into disjoint subsets, t_i, of size s such as $T = \cup t_i$. s is the only parameter of the algorithm. In this study the dataset is randomly partitioned, although other methods may be devised. Then, the instance selection algorithm of our choice is performed over every subset independently. The selected instances in each subset are joined again. With this new training set constructed with the selected instances, the process is repeated until a certain stop criterion is fulfilled. The process of combining the instances selected by the execution of the instance selection algorithm over each dataset can be performed in different ways. We can just repeat the partition process as in the original dataset. However, as the first partition is performed we can take advantage of this performed task. In this way, instead of repeating the partitioning process, we join together the subsets of selected instances until new subsets of approximately size *s* are obtained. The detailed process is shown in Fig. 2.

```
Data: A training set $T=\{\boxed{x}_1, y_1 \boxed{,}\square , \boxed{x}_n, y_n \boxed{\}}\square$, and subset size $s$.
Result: The reduced training set $S \square T$ .

$S=T$
divide instances into disjoint subsets $t_i : T=\square t_i$ of size $s$

repeat until stop criterion

  for each subset $t_i$ do
    apply instance selection algorithm to $t_i$ to obtain $s_i \square t_i$
    remove from $S$ the instances removed from $t_i$
  end for

  fusion subsets $s_i$ to obtain new subsets $t_j$ of size $s$

end repeat

return $S$
```

Fig. 2. Recursive divide-and-conquer instance selection algorithm

The stop criterion may be obtained in different ways. We can have a goal in terms of testing error or reduction of storage and stop the algorithm when that goal is achieved. However, to avoid the necessity of setting any additional parameter, we obtain the stop criterion by means of cross-validation. We apply the algorithm using a cross-validation setup and obtain the number of steps before the testing error starts to grow. This number of steps gives the stopping criterion.

3.2 Democratic instance selection

The above method is very fast as it will be shown in the experimental results. However, it has the drawback of worsening the testing error achieved by some algorithms for certain problems. To improve the results of our approach in this aspect, we have developed a second method we called *democratic* instance selection. The method is also based on the general divide-and-conquer approach but including ideas from ensembles of classifiers.

Democratic instance selection is based on repeating several rounds of a fast instance selection process. Each round on its own would not be able to achieve a good performance. However, the combination of several rounds using a voting scheme is able to match the performance of an instance selection algorithm applied to the whole dataset with a large reduction in the time of the algorithm. Thus, in a different setup from the case of ensembles of classifiers, we can consider our method a form of "ensembling" instance selection.

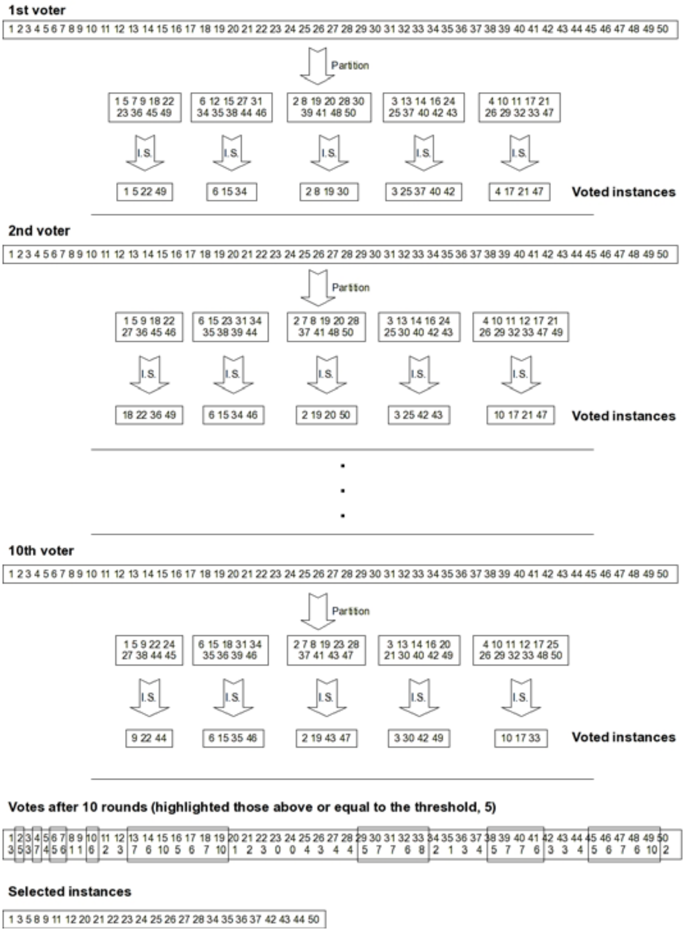


Fig. 3. Outline of democratic instance selection method

In classification, several weak learners are combined into an ensemble which is able to improve the performance of any of the weak learners isolated (García-Pedrajas et al., 2007). In our method, the instance selection algorithm applied to a partition into disjoint subsets of the original dataset can be considered a *weak instance selector*, as it has a partial view of the dataset. The combination of these weak selectors using a voting scheme is similar to the combination of different learners in an ensemble using a voting scheme. Fig. 3 shows a general outline of the method.

An important issue in our method is determining the number of votes needed to remove an instance from the training set. Preliminary experiments showed that this number highly depends on the specific dataset. Thus, it is not possible to set a general pre-established value usable in any dataset. On the contrary, we need a way of selecting this value directly from the dataset in run time.

A first natural choice would be the use of a cross-validation procedure. However, this method is very time consuming. A second choice is estimating the best value for the number of votes from the effect on the training set. This latter method is the one we have chosen. The election of the number of votes must take into account two different criteria: training error, ε_t , and storage, or memory, requirements m . Both values must be minimized as much as possible. Our method of choosing the number of votes needed to remove an instance is based on obtaining the threshold number of votes, v , that minimizes a fitness criterion, $f(v)$, which is a combination of these two values:

$$f(v) = a\varepsilon_t(v) + (1-a)m(v), \tag{1}$$

where α is a value in the interval *[0, 1]* which measures the relative relevance of both values. In general, the minimization of the error is more important than storage reduction, as we prefer a lesser error even if the reduction is smaller. Thus, we have used a value of $\alpha = 0.75$. Different values can be used if the researcher is more interested in reduction than in error. *m* is measured as the percentage of instances retained, and ε_t is the training error. However, estimating the training error is time consuming if we have large datasets. To avoid this problem the training error is estimated using only a small percentage of the whole dataset, which is 1% for medium and large datasets, and 0.1% for huge datasets.

```
Data: A training set $T = \{(x_{1,}y_1),\ldots,(x_n, y_n)\}$, subset size s, and number of rounds r.
Result: The set of selected instances $S \subset T$.

for i = 1 to r do
    divide instances into disjoint subsets $t_i : \cup t_i = T$ of size s
    for each $t_i$ do
        apply instance selection algorithm to $t_i$
        store votes of removed instances from $t_i$
    end for
```

```
obtain threshold of votes, $v$ , to remove an instance
$S=T$
remove from $S$ all instances with a number of votes above or equal to $v$

return $S$
```

Fig. 4. Democratic instance selection algorithm

More formally, the process is the following: We perform *r* rounds of the algorithm and store the number of votes received by each instance. Then, we must obtain the threshold number of votes, *v*, to remove an instance. This value must be $v \in [1,r]$. We calculate the criterion $f(v)$ (eq. 1) for all the possible threshold values from 1 to *r*, and assign v to the value which minimizes the criterion. After that, we perform the instance selection removing the instances whose number of votes is above or equal to the obtained threshold *v*. Fig. 4 shows the steps of this algorithm.

4. Experimental setup and results

In order to make a comprehensive comparison between the standard algorithms and our proposal we have selected a set of 30 problems from the UCI Machine Learning Repository (Hettich et al., 1998). A summary of these data sets is shown in Table 1. We have selected datasets with, at least, 1000 instances. For estimating the storage reduction and generalization error we used a *k*-fold cross-validation method. In this method the available data is divided into *k* approximately equal subsets. Then, the method is learned *k* times, using, in turn, each one of the *k* subsets as testing set, and the remaining *k-1* subsets as training set. The estimated error is the average testing error of the *k* subsets. A fairly standard value for *k* is *k* = 10.

4.1 Evaluating instance selection methods

The evaluation of a certain instance selection algorithm is not a trivial task. We can distinguish two basic approaches: direct and indirect evaluation (Liu & Motoda, 2002). Direct evaluation evaluates a certain algorithm based exclusively on the data. The objective is to measure at which extent the selected instances reflect the information present in the original data. Some proposed measures are entropy (Cover & Thomas, 1991), moments (Smith, 1998), and histograms (Chaudhuri et al., 1998).

Indirect methods evaluate the effect of the instance selection algorithm on the task at hand. So, if we are interested in classification we evaluate the performance of the used classifier when using the reduced set obtained after instance selection as learning set.

		Features				
Data set	Instances	Real	Binary	Nominal	Classes	1-NN error
abalone	4177	7	-	1	29	0.8034
adult	48842	6	1	7	2	0.2005
car	1728	-	-	6	4	0.1581
gene	3175	-	-	60	3	0.2767
german	1000	6	3	11	2	0.3120
hypothyroid	3772	7	20	2	4	0.0692
isolet	7797	617	-	-	26	0.1443
krkopt	28056	6	-	-	18	0.4356
kr vs. kp	3196	-	34	2	2	0.0828
letter	20000	16	-	-	26	0.0454
magic04	19020	10	-	-	2	0.2084
mfeat-fac	2000	216	-	-	10	0.0350
mfeat-fou	2000	76	-	-	10	0.2080
mfeat-kar	2000	64	-	-	10	0.0435
mfeat-mor	2000	6	-	-	10	0.2925
mfeat-pix	2000	240	-	-	10	0.0270
mfeat-zer	2000	47	-	-	10	0.2140
nursery	12960	-	1	7	5	0.2502
optdigits	5620	64	-	-	10	0.0256
page-blocks	5473	10	-	-	5	0.0369
pendigits	10992	16	-	-	10	0.0066
phoneme	5404	5	-	-	2	0.0952
satimage	6435	36	-	-	6	0.0939
segment	2310	19	-	-	7	0.0398
shuttle	58000	9	-	-	7	0.0010
sick	3772	7	20	2	2	0.0430
texture	5500	40	-	-	11	0.0105
waveform	5000	40	-	-	3	0.2860
yeast	1484	8	-	-	10	0.4879

Table 1. Summary of datasets used in our experiments

Therefore, when evaluating instance selection algorithms for instance learning, the most usual way of evaluation is estimating the performance of the algorithms on a set of benchmark problems. In those problems several criteria can be considered, such as (Wilson & Martínez, 2000): storage reduction, generalization accuracy, noise tolerance, and learning speed. Speed considerations are difficult to measure, as we are evaluating not only an algorithm but also a certain implementation. However, as the main aim of our work is scaling up instance selection algorithms, execution time is a basic issue. To allow a fair comparison, we have performed all the experiments in the same machine, a bi-processor computer with two Intel Xeon QuadCore at 1.60GHz.

One of the advantages of our approach is that it can be applied to any kind of instance selection method. As the instance selection method to apply is just a parameter of the algorithm, there is no restriction in the algorithm selected. In the experiments we have used several of the most widely used instance selection methods.

In order to obtain an accurate view of the usefulness of our method, we must select some of the most widely used instance selection algorithms. We have chosen to test our model using several of the most successful state-of-the-art algorithms. Initially, we used the algorithm ICF (Brighton & Mellish, 2002). ICF (Iterative Case Filtering) is based on the concepts of *coverage* and *reachability* of an instance *c*, which are defined as follows:

$$\text{Coverage}(c) = c' \in T : c \in LocalSet\left(c'\right)$$

$$\text{Reachable}(c) = c' \in T : c' \in LocalSet\left(c\right)$$

The local-set of a case *c* is defined as "the set of cases contained in the largest hypersphere centered on *c* such that only cases in the same class as *c* are contained in the hypersphere" (Brighton & Mellish, 2002) so the hypersphere is bounded by the first instance of different class. The coverage set of an instance includes the instances that have this as one of their neighbors and the reachable set is formed by the instances that are neighbors to this instance. The algorithm is based on repeatedly applying a deleting rule to the set of retained instances until no more instances fulfill the deleting rule.

In addition to this method, it is worth mentioning Reduced Nearest Neighbor (RNN) rule (Gates, 1972). This method is extremely simple, but it also shows an impressive performance in terms of storage reduction. In fact, it is the best of the methods used in these experiments in reducing storage requirements, as will be shown in the next section. However, it has a serious drawback, its computational complexity. Among the standard methods used this is the one that shows a worst scalability, taking several hundreds hours in the worst case. Therefore, RNN is the perfect target for our methodology, an instance selection method highly efficient but with a serious scalability problem. So we have also tested our approach using RNN, as base instance selection method.

The same parameters were used for the standard version of every algorithm and its application within our methodology. All the standard methods have no relevant parameters, the only value we must set is *k*, the number of nearest neighbors. Both, for ICF and RNN, we used *k* = 3 neighbors. This is a fairly standard value (Cano et al., 2003). Our method has two parameters: subset size, *s*, for both methods, and number of rounds, *r*, for the democratic approach. Regarding subset size we must use a value large enough to allow for a meaningful application of the instance selection algorithm on the subset, and small enough to allow a fast execution, as the time used by our method grows with *s*. As a compromise value we have chosen *s* = 100. For the number of rounds we have chosen a small value to allow for a fast execution, *r* = 10. The application of our recursive divide-and-conquer method with a certain instance selection algorithm X will be named RECURIS.X and the democratic approach named DEMOIS.X.

4.2 Recursive divide-and-conquer approach

In this section we show the results using the recursive approach. First, we compare the proposed approach against standard instance selection methods in terms of testing error and storage requirements. In the next section we will show execution time results.

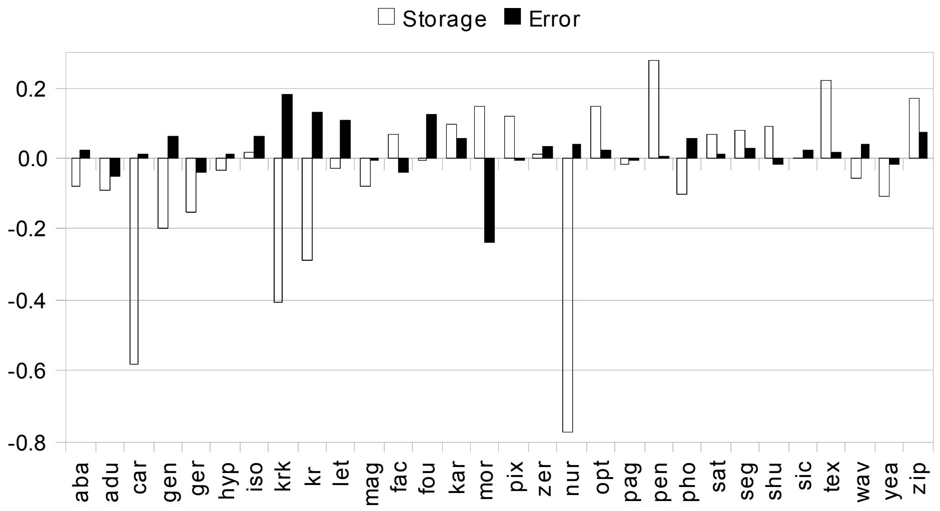


Fig. 5. Results of standard ICF method and its recursive counterpart for testing error and storage requirements

Fig. 5 shows the results comparing standard ICF and its recursive counterpart. The figure (as well as the following ones) shows for each dataset the difference between the standard method and our approach, a negative value meaning a better results of our proposal. The figure shows that in terms of storage reduction our method is better in general, achieving for some datasets, namely car, gene, german, krkopt, krvskp and nursery, significant improvements over the standard method. In terms of testing error RECURIS.ICF is slightly worse than standard ICF.

Fig. 6 shows the results for RNN as base instance selection method. This a very good test of our approach, as RNN is able to achieve very good results in terms of storage reduction while keeping testing error in moderate bounds. However, RNN has a big problem of scalability. The results show that our method is able to mostly keep the good performance of RNN in terms of storage requirements, although with a general worst behavior. However, this is compensated by a better testing error for most datasets.

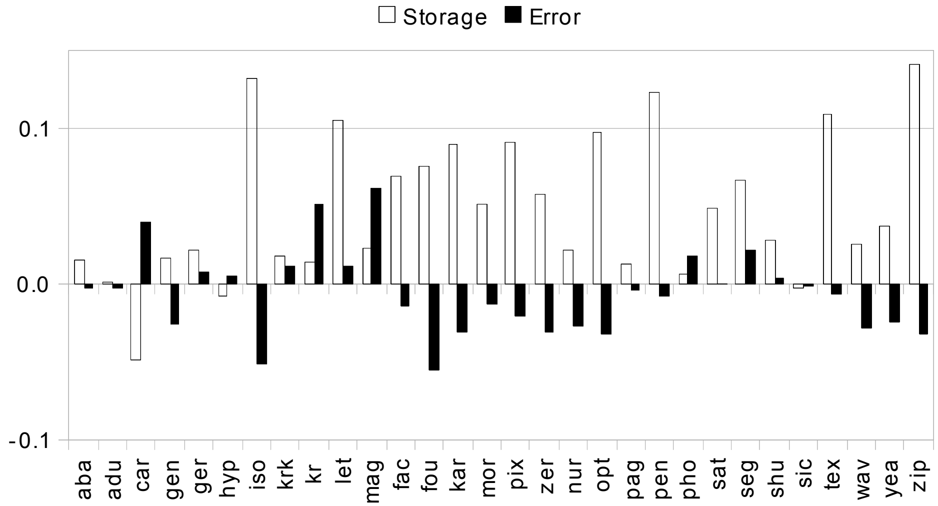


Fig. 6. Results of standard RNN and its recursive counterpart in terms of testing error and storage requirements

As an alternative to these standard methods, genetic algorithms have been applied to instance selection, considering this task to be a search problem. The application is easy and straightforward. Each individual is a binary vector that codes a certain sample of the training set. The evaluation is usually made considering both data reduction and classification accuracy. Examples of applications of genetic algorithms to instance selection can be found in (Kuncheva, 1995), (Ishibuchi & Nakashima, 2000) and (Reeves & Bush, 2001). Cano et al. (2003) performed a comprehensive comparison of the performance of different evolutionary algorithms for instance selection. They compared a generational genetic algorithm (Goldberg, 1989), a steady-state genetic algorithm (Whitley, 1989), a CHC genetic algorithm (Eshelman, 1990), and a population based incremental learning algorithm (Baluja, 1994). They found that evolutionary based methods were able to outperform classical algorithms in both classification accuracy and data reduction. Among the evolutionary algorithms, CHC was able to achieve the best overall performance.

In evolutionary computation, a population (set) of individuals (solutions to the problem faced) are codified following a code similar to the genetic code of plants and animals. This population of solutions is evolved (modified) over a certain number of generations (iterations) until the defined stop criterion is fulfilled. Each individual is assigned a real value that measures its ability to solve the problem, which is called its *fitness*.

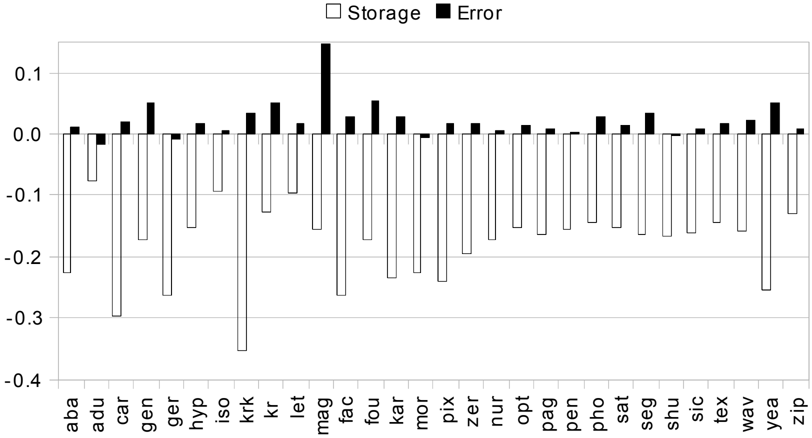


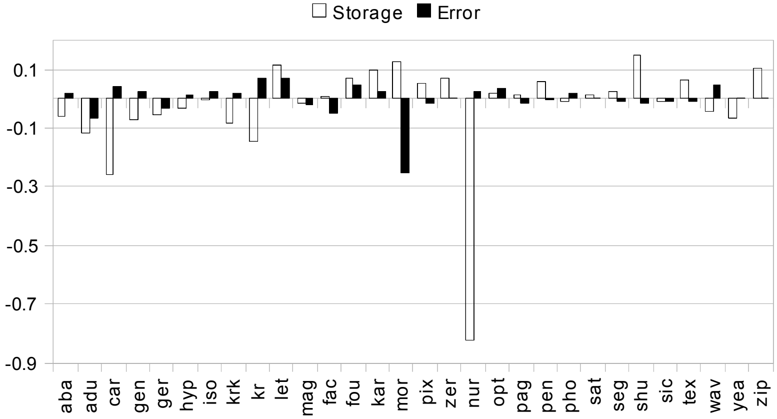
Fig. 7. Results of standard CHC and its recursive counterpart in terms of testing error and storage requirements

In each iteration new solutions are obtained combining two or more individuals (crossover operator) or randomly modifying one individual (mutation operator). After applying these two operators a subset of individuals is selected to survive to the next generation, either by sampling the current individuals with a probability proportional to their fitness, or by selecting the best ones (elitism). The repeated processes of crossover, mutation and selection are able to obtain increasingly better solutions for many problems of Artificial Intelligence.

Nevertheless, the major problem addressed when applying genetic algorithms to instance selection is the scaling of the algorithm. As the number of instances grows, the time needed for the genetic algorithm to reach a good solution increases exponentially, making it totally useless for large problems. As we are concerned with this problem, we have used as fifth instance selection method a genetic algorithm using CHC methodology. The execution time of CHC is clearly longer than the time spent by ICF, so it gives us a good benchmark to test our methodology on an algorithm that, as RNN, has a big scalability problem.

For CHC, see Fig. 7, the results show that the recursive approach is able to improve the results of the standard algorithm in terms of storage requirements but the error is worse than when using the whole dataset. However, the achieved storage reduction is relevant, and our method is clearly worse than standard CHC only in magic04 problem.

An interesting side result is the problem of scalability of CHC algorithm, which is more marked for this algorithm than for the previous ones. In other works, (Cano et al., 2003) (García-Pedrajas et al., 2009), CHC algorithm was compared with standard methods in small to medium problems. For those problems, the performance of CHC was better than the performance of other methods. However, as the datasets are larger, the scalability problem of CHC manifests itself. In our set of problems, CHC clearly performs worse than ICF and RNN in terms of storage reduction. We must take into account that for CHC we need a bit in the chromosome for each instance in the dataset. This means that for large problems, such as adult, krkopt, letter, magic or shuttle, the chromosome has more than 10000 bits, making the convergence of the algorithm problematic. Thus, CHC is, together with RNN, an excellent example of the applicability of our approach.



4.3 Democratic approach

In this section we show the results using the democratic approach. Results for ICF and DEMOIS.ICF are plotted in Fig. 8.

In terms of testing error, DEMOIS.ICF is able to match the results of ICF for most of the datasets. In terms of storage reduction the average performance of both algorithms is similar, with a remarkably good performance of DEMOIS.ICF for nursery and car datasets.

The next experiment is conducted using as base instance selection algorithm RNN. The results are plotted in Fig. 9. As we stated in the previous section, this is a perfect example of the potentialities of our approach. In our experiments RNN showed the best performance in terms of storage reduction. However, the algorithm has a very serious problem of scalability. As an extreme example, for adult problem it took more than 500 hours per experiment. This scalability problem prevents is application in those problems where it would be most useful.

Fig. 8. Results of standard ICF method and its democratic counterpart for testing error and storage requirements

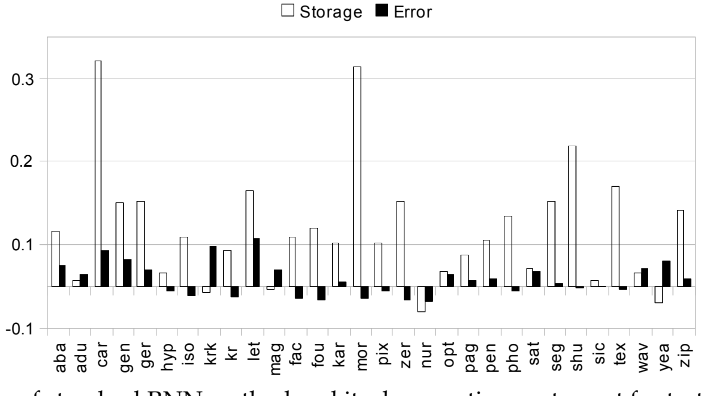


Fig. 9. Results of standard RNN method and its democratic counterpart for testing error and storage requirements

The figure shows how DEMOIS.RNN is able to solve the scalability problem of RNN. In terms of testing error, it is able to achieve a similar performance as standard RNN. In terms of storage reduction our algorithm performs worse than RNN. However, the performance of DEMOIS.RNN is still very good, in fact, better than any other of the previous algorithms. So, our approach is able to scale RNN to complex problems, improving its results in terms of testing error, but with a small worsening of the storage reduction. In terms of execution time the results are remarkable, the reduction of the time consumed by the selection process is large, with the extreme example of the two most time consuming datasets, adult and krkopt, where the speed-up is more than a hundred times (see next section).

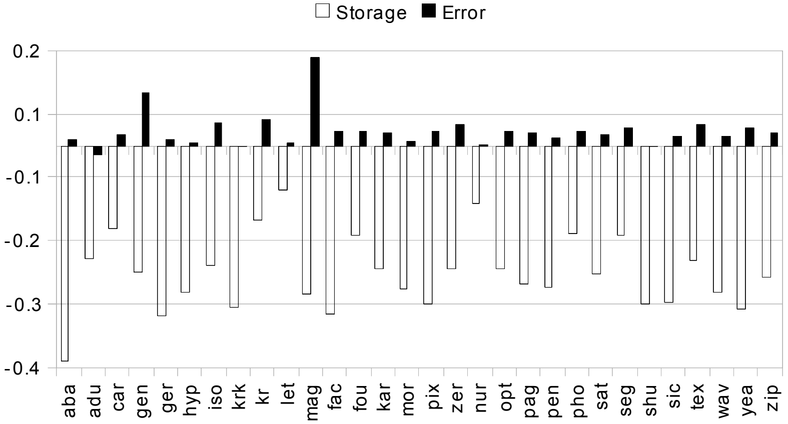


Fig. 10. Results of standard CHC method and its democratic counterpart for testing error and storage requirements

Fig. 10 plots the results of CHC algorithm. For this method, the scaling up of CHC provided by DEMOIS.CHC is evident not only in terms of running time, with a large reduction in all 30 datasets, but also in terms of storage reduction. DEMOIS.CHC is able to improve the reduction of CHC in all 30 datasets, with an average improvement of more than 20%, from an average storage of CHC of 31.83% to an average storage of 11.58%. The bad side effect is a worse testing error, which is however not very marked and compensated by the improvement in running time and storage reduction. As a summary, for CHC the results show that the democratic approach is able to improve the results of the standard algorithm in terms of storage requirements but the error is worse than when using the whole dataset. However, as it was the case for the recursive approach, there is a clear gaining in storage reduction with a moderately worse testing error.

4.4 Time

As we have estated our main aim is the scaling up of instance selection algorithms. In the previous sections we have shown that our methodology is able to match the performance of standard instance selection algorithms. In this section we show the results of execution time spent by each algorithm, showing a dramatic advantage of our approach. Fig. 11, 12 and 13 show the execution time of ICF, RNN and CHC methods respectively. The figures show execution time, in seconds, plotted against problem size.

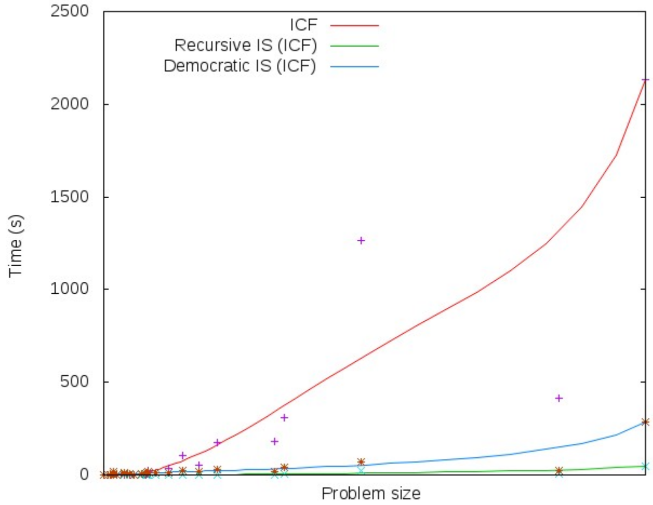


Fig. 11. Execution time using ICF as base instance selection algorithm

All the three figures show the excellent behavior of the two described methods. Both behave almost linearly as the problem size grows. On the other hand, ICF shows it is a quadratic complexity method and RNN and CHC behave far worse.

From a theoretical point of view the two algorithms presented in this chapter are of linear complexity. For the recursive approach we divide the dataset into n_s subsets of size *s*. Then, we apply the instance selection algorithm to each subset. The time needed for performing the selection in each subset will be fixed as the size of each subset is always *s*, regardless the number of instances of the datasets. More instances means a larger n_s . Thus, the complexity of each step of the recursive algorithm will be linear as n_s depends linearly on *n*, the size of the dataset. The algorithm performs a few of these steps before reaching the stopping criterion, and thus the whole method is of linear complexity.

The democratic approach also divides the dataset into partitions of disjoint subsets of size *s*. Thus, the chosen instance selection algorithm is always applied to a subset of fixed size, *s*, which is independent from the actual size of the dataset. The complexity of this application of the algorithm depends on the base instance selection algorithm we are using, but will always be small, as the size *s*is always small. Let *K* be the number of operations needed by the instance selection algorithm to perform its task in a dataset of size *s*. For a dataset of *n* instances we must perform this instance selection process once for each subset, that is *n/s* times, spending a time proportional to *(n/s)K*. The total time needed by the algorithm to perform *r* rounds will be proportional to *r(n/s)K*, which is linear in the number of instances, as *K* is a constant value.

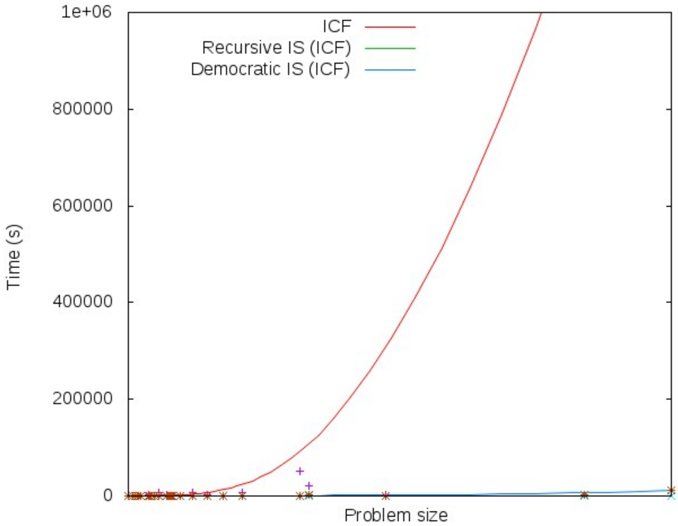


Fig. 12. Execution time using RNN as base instance selection algorithm.

Thus, the gaining in execution time would be greater as the size of the datasets is larger. If the complexity of the instance selection algorithm is greater, the reduction of the execution will be even better. The method has the additional advantage of allowing an easy parallel implementation. As the application of the instance selection algorithm to each subset is independent from all the remaining subsets, all the subsets can be processed at the same time, even for different rounds of votes. Also, the communication between the nodes of the parallel execution is small.

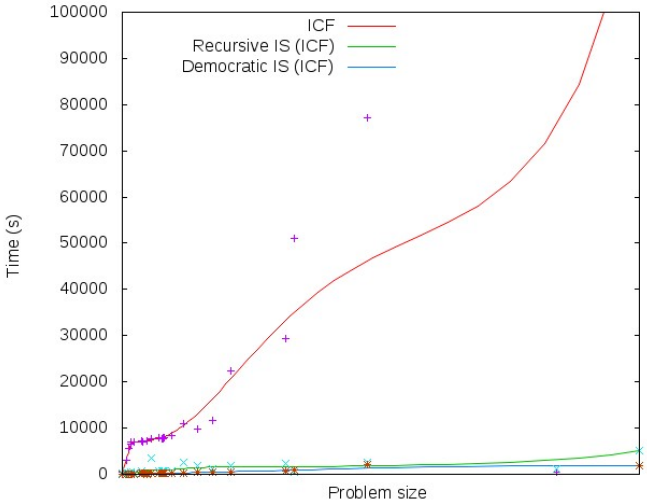


Fig. 13. Execution time using CHC as base instance selection algorithm

An additional process completes the method, the determination of the number of votes. Regarding the determination of the number of votes, the process can be made in different ways. If we consider all the training instances, the cost of this step would be $O\left(n^2\right)$.

However, to keep the complexity linear we use a random subset of the training set for determining the number of votes, with a limit on the maximum size of this subset that is fixed for any dataset. In this way, from medium to large datasets we use the 10% of the training set, for huge problems the 1%, and the percentage is further reduced as the size of the dataset grows. In fact, we have experimentally verified that we can consider any reasonable bound[1] in the number of instances without damaging the performance of the algorithm. Using a small percentage does not harm the estimation of the threshold of votes. With this method the complexity of this step is $O(1)$ as the number of instances used is bounded regardless the size of the dataset.

Finally, we consider the partition of the dataset apart from the algorithm as many different partition methods can be devised. The performed random partition is of complexity $O(n)$.

7. Conclusions

In this chapter we have shown two new methods for scaling up instance selection algorithms. These methods are applicable to any instance selection method without any modification. The methods consist of a recursive procedure, where the dataset is partitioned into disjoint subsets, an instance selection algorithm is applied to each subset, and then the selected instances are rejoined to repeat the process, and a democratic approach where several rounds of approximate instance selection are performed and the result is obtained by a voting scheme.

Using three well-known instance selection algorithms, ICF, RNN and a CHC genetic algorithm, we have shown that our method is able to match the performance of the original algorithms with a considerable reduction in execution time. In terms of reduction of storage requirements, our approach is even better than the use of the original instance selection algorithm over the whole dataset. Additionally, our method is straightforwardly parallelizable without modifications.

The proposed methods allow the application of instance selection algorithms to almost any problem size. The behavior is linear in the number of instances as it has been shown both theoretically and experimentally.

Furthermore, this philosophy can be extended to other learning algorithms such as feature selection or clustering, which means it is a powerful tool for scaling up machine learning algorithms.

8. References

Barandela, R., Ferri, F. J. & Sánchez, J. S. (2005). Decision boundary preserving prototype selection for nearest neighbor classification. *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 6, 787-806.

[1] This *reasonable* bound can be from a few hundreds to a few thousands, even for huge datasets.

Baluja, S. (1996). Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, vol.* 26, no. 3, 450–463.

Brighton, H. & Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, vol. 6, 153-172.

Brodley, C. E. (1995). Recursive automatic bias selection for classifier construction. *Machine Learning*, vol. 20, no. 1/2, 63–94.

Cano, J. R., Herrera, F. & Lozano, M. (2003). Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study. *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, 561-575.

Cano, J. R., Herrera, F. & Lozano, M. (2005). Stratification for scaling up evolutionary prototype selection. *Pattern Recognition Letters,* vol. 26, no. 7, 953–963.

Chaudhuri, S., Motwani, R. & Narasayya, V. (1998). Random sampling for histogram construction: How much is enough?. In: *Proceedings of ACM SIGMOD, International Conference on Management of Data,* ACM Press, Haas, L., and Tiwary, A., (Eds.), 436-447.

Chawla, N. W., Hall, L. O., Bowyer, K. W. & and Kegelmeyer, W. P. (2004). Learning Ensembles from Bites: A Scalable and Accurate Approach. *Journal of Machine Learning Research*, vol. 5, 421–451.

Chen, J. H., Chen, H. M. & Ho, S. Y. (2005). Design of nearest neighbor classifiers: multi-objective approach. *International Journal of Approximate Reasoning,* vol. 40, no. 1-2, 3–22.

Cochran, W. (1977). *Sampling techniques*, John Wiley & Sons, New York, USA.

Cover, T. M. & Thomas, J. A. (1991). *Elements of Information Theory*, John Wiley & Sons, Inc.

De Haro-García, A. & García-Pedrajas, N. (2009). A divide-and-conquer recursive approach for scaling up instance selection algorithms. *Data Mining and Knowledge Discovery*, vol. 18, 392-418.

Eshelman, L. J. (1990). *The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination*, Morgan Kauffman.

García-Pedrajas, N., García-Osorio, C. & Fyfe, C. (2007): Nonlinear Boosting Projections for Ensemble Construction. *Journal of Machine Learning Research*, vol. 8, 1–33.

García-Pedrajas, N., Romero del Castillo, J. A. & Ortiz-Boyer, D. (2009). A cooperative coevolutionary algorithm for instance selection for instance-based learning, *Machine Learning*, in press.

Gates, G. W. (1977). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, vol. 18, no. 3, 431-433.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning, Addison—Wesley, Reading, USA.

Hettich, S., Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences.

Ishibuchi, H. & Nakashima, T. (2000). Pattern and Feature Selection by Genetic Algorithms in Nearest Neighbor Classification. Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 4, no. 2, 138-145.

Kim, S.-W. & Oommen, B. J. (2004). Enhancing Prototype Reduction Schemes With Recursion: A Method Applicable for "Large" Data Sets. *IEEE Transactions on Systems, Man, and Cybernetics--Part B: Cybernetics*, vol. 34, no. 3, 1384–1397.

Kivinen, J. & Mannila, H. (1994). The power of sampling in knowledge discovery, In: *Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, ACM Press, 77–85.

Kuncheva, L. (1995). Editing for the *k*-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters*, vol. 16, 809-814.

Li, J., Manry, M. T., Yu, C. & Wilson, D. R. (2005). Prototype classifier design with pruning, *International Journal of Artificial Intelligence Tools*, vol. 14, no. 1-2, 261–280.

Liu, H. & Motoda, H. (2002). On issues of instance selection. *Data Mining and Knowledge Discovery*, vol. 6, 115-130.

Reeves, C. R. & Bush, D. R. (2001). Using genetic algorithms for training data selection in RBF networks. In: I*nstances Selection and Construction for Data Mining*, Liu, H. & Motoda, (Ed.), 339-356, Kluwer, Norwell, USA.

Smith, P. (1998). *Into Statistics*, Springer-Verlag, Berlin, Germany.

Son, S. H. & Kim, J. Y. (2006). Data reduction for instance-based learning using entropy-based partitioning, In: Proceedings *of the International Conference on Computational Science and Its Applications - ICCSA 2006*, Springer, 590–599.

Whitley, D (1986). The GENITOR Algorithm and Selective Pressure, In: Proceedings of the 3rd International Conference on Genetic Algorithms, 116–121, Morgan Kaufmann.

Wilson, D. R. & Martínez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, vol. 38, 257-286.

Zhu, X., and Wu, X. (2006). Scalable representative instance selection and ranking. In: *Proceedings of the 18th International Conference on Patter Recognition (ICPR'06)*, IEEE Computer Society, 352 – 355.

Ant Colony Optimization

Benlian Xu, Jihong Zhu and Qinlan Chen
Changshu Institute of Technology
China

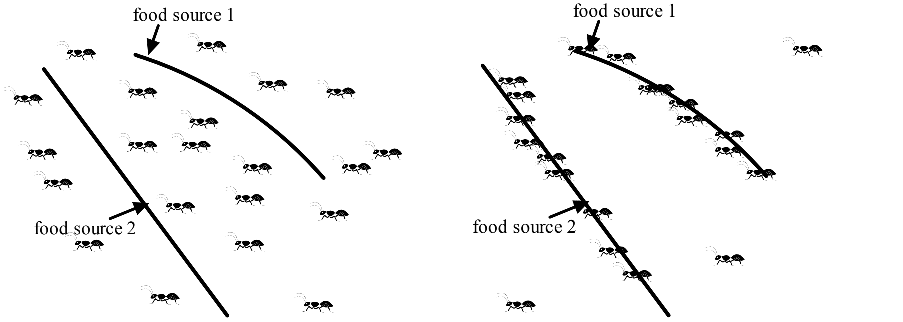
1. Introduction

Swarm intelligence is a relatively novel approach to problem solving that takes inspiration from the social behaviors of insects and of other animals. In particular, ants have inspired a number of methods and techniques among which the most studied and the most successful one is the ant colony optimization.

Ant colony optimization (ACO) algorithm, a novel population-based and meta-heuristic approach, was recently proposed by Dorigo et al. to solve several discrete optimization problems (Dorigo, 1996, 1997). The general ACO algorithm mimics the way real ants find the shortest route between a food source and their nest. The ants communicate with one another by means of pheromone trails and exchange information indirectly about which path should be followed. Paths with higher pheromone levels will more likely be chosen and thus reinforced later, while the pheromone intensity of paths that are not chosen is decreased by evaporation. This form of indirect communication is known as stigmergy, and provides the ant colony shortest-path finding capabilities. The first algorithm following the principles of the ACO meta-heuristic is the Ant System (AS) (Dorigo,1996), where ants iteratively construct solutions and add pheromone to the paths corresponding to these solutions. Path selection is a stochastic procedure based on two parameters, the pheromone and heuristic values, which will be detailed in the following section in this chapter. The pheromone value gives an indication of the number of ants that chose the trail recently, while the heuristic value is problem-dependent and it has different forms for different cases.

Due to the fact that the general ACO can be easily extended to deal with other optimization problems, its several variants has been proposed as well, such as Ant Colony System (Dorigo,1997), rank-based Ant System (Bullnheimer,1999), and Elitist Ant System (Dorigo,1996) . And the above variants of ACO have been applied to a variety of different problems, such as vehicle routing (Montemanni,2005), scheduling (Blum,2005), and travelling salesman problem (Stützle,2000). Recently, ants have also entered the data mining domain, addressing both the clustering (Kanade,2007), and classification task (Martens et al.,2007).

This chapter will focus on another application of ACO to track initiation in the target tracking field. To the best of our knowledge, there are few reports on the track initiation using the ACO. But in the real world, it is observed that there is a case in which almost all ants are inclined to gather around the food sources in the form of line or curve. Fig. 1 shows the evolution process of ants searching for foods. Initially, all ants are distributed randomly in the plane as in Fig.1 (a), and a few hours later we find that most of ants gather together around the food sources as shown in Fig.1 (b). Taking inspiration from such phenomenon, we may regard these linear or curvy food sources as tentative tracks to be initialized, and the corresponding ant model is established from the optimal aspect to solve the problem of multiple track initiation.



(a) Initial distribution of ants (b) The distribution of ants a few hours later

Fig. 1. The evolution process of ant search for foods

The remainder of this chapter is structured as follows. First, in section 2, the widely used ant system and its successors are introduced. Section 3 gives the new application of ACO to the track initiation problem, and the system of ants of different tasks is modeled to coincide with the problem. The performance comparison of ACO-based techniques for track initiation is carried out and analysized in Section 4. Finally, some conclusions are drawn.

2. Ant System and Its Direct Successors

2.1 Ant System

Initially, three different versions of AS were developed (Dorigo et al., 1991), namely ant-density, ant-quantity, and ant-cycle. In the ant-density and ant-quantity versions the ants updated the pheromone directly after a move from one city to an adjacent city, while in the ant-cycle version the pheromone update was only done after all the ants had constructed the tours and the amount of pheromone deposited by each ant was set to be a function of the tour quality.

The two main phases of the AS algorithm constitute the ants' solution construction and the pheromone update. In AS, a good way to initialize the pheromone trails is to set them to a value slightly higher than the expected amount of pheromone deposited by the ants in one iteration. The reason for this choice is that if the initial pheromone values are too low, then the search is quickly biased by the first tours generated by the ants, which in general leads toward the exploration of inferior zones of the search space. On the other side, if the initial pheromone values are too high, then many iterations are lost waiting until pheromone evaporation reduces enough pheromone values, so that pheromone added by ants can start to bias the search.

Tour Construction

In AS, m (artificial) ants incrementally build a tour of the TSP. Initially, ants are put on randomly chosen cities. At each construction step, ant k applies a probabilistic action choice rule, called random proportional rule, to decide which city to visit next. In particular, the probability with which ant k , located at city i , chooses to go to city j is

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}}, \quad if \ j \in N_i^k, \tag{1}$$

where $\eta_{ij} = 1/d_{ij}$ is a heuristic value that is computed in advance, α and β are two parameters which determine the relative importance of the pheromone trail and the heuristic information, and N_i^k is the set of cities that ant k has not visited so far. By this probabilistic rule, the probability of choosing the arc (i, j) may increase with the bigger value of the associated pheromone trail τ_{ij} and of the heuristic information value η_{ij}. The role of the parameters α and β is described as below. If $\alpha = 0$, the closest cities are more likely to be selected: this corresponds to a classic stochastic greedy algorithm. If $\beta = 0$, it means that the pheromone is used alone, without any heuristic bias. This generally leads to rather poor results and, in particular, for values of $\alpha > 1$ it leads to earlier stagnation situation, that is, a situation in which all the ants follow the same path and construct the same tour, which, in general, is strongly suboptimal.

Each ant maintains a memory which records the cities already visited. And moreover, this memory is used to define the feasible neighbourhood N_i^k in the construction rule given by equation (1). In addition, such a memory allows ant k both to compute the length of the tour T^k it generated and to retrace the path to deposit pheromone for upcoming global pheromone update.

Concerning solution construction, there are two different ways of implementing it: parallel and sequential solution construction. In the parallel implementation, at each construction step all ants move from their current city to the next one, while in the sequential implementation an ant builds a complete tour before the next one starts to build another one. In the AS case, both choices for the implementation of the tour construction are equivalent in the sense that they do not significantly influence the algorithm's behaviour.

Update of Pheromone Trails

After all the ants have constructed their tours, the pheromone trails are updated. This is done by first lowering the pheromone value on all arcs by a factor, and then adding an amount of pheromone on the arcs the ants have crossed in their tours. Pheromone evaporation is implemented by the following law

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij}, \quad \forall (i, j) \in L \tag{2}$$

where $0 < \rho < 1$ is the pheromone evaporation rate. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to "forget" bad decisions previously taken. In fact, if an arc is not chosen by the ants, its associated pheromone value decreases exponentially with the number of iterations. After evaporation, all ants deposit pheromone on the arcs they have crossed in their tour:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}, \forall(i,j)\in L \tag{3}$$

where $\Delta\tau_{ij}^{k}$ is the amount of pheromone ant k deposits on the arcs it has visited. It is defined as follows:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^{k} & \textit{if arc } (i,j) \textit{ belongs to } T^{k} \\ 0 & \textit{otherw} \end{cases} \tag{4}$$

where C^{k}, the length of the tour T^{k} travelled by ant k, is computed as the sum of the lengths of the arcs belonging to T^{k}. By means of equation (4), the shorter an ant's tour is, the more pheromone the arcs belonging to this tour receive. In general, arcs that are used by many ants and which are part of short tours, receive more pheromone and are, therefore, more likely to be chosen by ants in the following iterations of the algorithm.

2.2. Elitist Ant System

The elitist strategy for Ant System (EAS) (Dorigo,1996) is, in principle, to provide a strong additional reinforcement to the arcs belonging to the best tour found since the start of the algorithm. Note that this additional feedback to the best-so-far tour is another example of a daemon action of the ACO meta-heuristics.

Update of Pheromone Trails

The additional reinforcement of tour T^{bs} is achieved by adding a quantity e/C^{bs} to its arcs, where e is a parameter that defines the weight given to the best-so-far tour T^{bs}, and C^{bs} is its length. Thus, equation (3) for the pheromone deposit becomes

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs} \tag{5}$$

where $\Delta\tau_{ij}^{k}$ is defined as in equation (4) and $\Delta\tau_{ij}^{bs}$ is defined as follows:

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/C^{bs} & \textit{if arc } (i,j) \textit{ belongs to } T^{bs} \\ 0 & \textit{otherw} \end{cases} \tag{6}$$

Note that in EAS the pheromone evaporation is implemented as in AS.

2.3. Rank-Based Ant System

Another improvement over AS (Bullnheimer,1999) is the rank-based version of AS (AS_{rank}). In AS_{rank} each ant deposits an amount of pheromone that decreases with its rank. Additionally, as in EAS, the best-so-far ant always receives the largest amount of pheromone in each iteration.

Update of Pheromone Trails

Before updating the pheromone trails, the ants are sorted by increasing tour length and the quantity of pheromone an ant deposits is weighted according to the rank of the ant. In each iteration, assume that total W best-ranked ants are considered, and only the $(W-1)$ best-ranked ants and the ant that produced the best-so-far tour are allowed to deposit pheromone. The best-so-far tour gives the strongest feedback with weight w; the r th best ant of the current iteration contributes to pheromone updating with the value $1/C^r$ multiplied by a weight given by max $\{0, W-r\}$. Thus, the AS_{rank} pheromone update rule is

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{r=1}^{W-1} (W-r)\Delta\tau_{ij}^r + w\Delta\tau_{ij}^{bs} \tag{7}$$

where $\Delta\tau_{ij}^r = 1/C^r$ and $\Delta\tau_{ij}^{bs} = 1/C^{bs}$.

2.4 *Max- Min* Ant System

Max-Min Ant System (MMAS) (St ü tzle & Hoos, 2000) introduces some main modifications with respect to AS. First, it strongly exploits the best tours found: only either the iteration-best ant, that is, the ant that produced the best tour in the current iteration, or the best-so-far ant is allowed to deposit pheromone. Unfortunately, such a strategy may lead to a stagnation situation in which all ants follow the same tour, because of the excessive growth of pheromone trails on arcs of a good, although suboptimal, tour. To counteract this effect, a second modification introduced by MMAS is that it limits the possible range of pheromone trail values to the interval $[\tau_{\min}, \tau_{\max}]$. Second, the pheromone trails are initialized to the upper pheromone trail limit, which, together with a small pheromone evaporation rate, increases the exploration of tours at the start of the search. Finally, in MMAS, pheromone trails are reinitialized each time the system approaches stagnation or when no improved tour has been generated for a certain number of consecutive iterations.

Update of Pheromone Trails

After all ants have constructed a tour, pheromones are updated by applying evaporation as in AS, followed by the deposit of new pheromone as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \tag{8}$$

where $\Delta\tau_{ij}^{best} = 1/C^{best}$. The ant which is allowed to add pheromone may be either the best-so-far, in which case $\Delta\tau_{ij}^{best} = 1/C^{bs}$, or the iteration-best, in which case $\Delta\tau_{ij}^{best} = 1/C^{ib}$, where C^{ib} is the length of the iteration-best tour. In general, in MMAS implementations both the iteration-best and the best-so-far update rules are used in an alternate way. Obviously, the choice of the relative frequency with which the two pheromone update rules are applied has an influence on how greedy the search is: When pheromone updates are always performed by the best-so-far ant, the search focuses very quickly around T^{bs}, whereas when it is the iteration-best ant that updates pheromones, then the number of arcs that receive pheromone is larger and the search is less directed.

Pheromone Trail Limits

In MMAS, lower and upper limits $\tau_{\min}$ and $\tau_{\max}$ on the possible pheromone values on any arc are imposed in order to avoid earlier searching stagnation. In particular, the imposed pheromone trail limits have the effect of limiting the probability p_{ij} of selecting a city j when an ant is in city i to the interval $[p_{\min}, p_{\max}]$, with $0 < p_{\min} \le p_{ij} \le p_{\max} \le 1$. Only when an ant k has just one single possible choice for the next city, that is $\left|N_i^k\right| = 1$, we have $p_{\min} = p_{\max} = 1$.

It is easy to show that, in the long run, the upper pheromone trail limit on any arc is bounded by $1/\rho C^*$, where C^* is the length of the optimal tour. Based on this result, MMAS uses an estimate of this value, $1/\rho C^{bs}$, to define $\tau_{\max}$: each time a new best-so-far tour is found, the value of $\tau_{\max}$ is updated. The lower pheromone trail limit is set to $\tau_{\min} = \tau_{\max}/\kappa$, where κ is a parameter (Stützle & Hoos, 2000).

Pheromone Trail Initialization and Re-initialization

At the start of the algorithm, the initial pheromone trails are set to an estimate of the upper pheromone trail limit. This way of initializing the pheromone trails, in combination with a small pheromone evaporation parameter, causes a slow increase in the relative difference in the pheromone trail levels, so that the initial search phase of MMAS is very explorative.

Note that, in MMAS, pheromone trails are occasionally re-initialized. Pheromone trail re-initialization is typically triggered when the algorithm approaches the stagnation behaviour or if for a given number of algorithm iterations no improved tour is found.

3. ACO for Track Initiation of Bearings-only multi-target tracking

3.1 Problem Presentation

Bearings-only multi-target tracking (BO-MTT) (Nardone, 1984 ; Dogancay, 2004, 2005) in a bistatic system can be described as: given a time history of noise-corrupted bearing measurements from two observers, the objective is to obtain optimum estimation of the positions, velocities and accelerations of all targets. Generally, the whole process of target tracking includes track initiation, track maintenance and track deletion. To the best of our knowledge, however, many reported literature mainly focused on the track maintenance, i.e. target tracking, without considering the track initiation process, after the motion of each target is modelled. Actually, track initiation plays an important role in evaluating the performance of subsequent target tracking, and improperly initiated tracks may either lead to target loss or the increase of consumption of limited resources.

In the case of multi-sensor-multi-target BOT, for instance, two-sensor-two-target BOT at a given scan, four Line of Sights (LOSs) are available alone to determine which LOS belongs to some target of interest. Usually, such a problem can also be dealt with the general track initiation techniques widely used in the radar tracking field through intersecting these LOSs to obtain a group of candidates of true targets' position points. However, such an operation will result in some intersections including both the true "target positions" and the virtual "target position" called "ghost", as shown in Fig.2. These "ghosts", in fact, do not belong to any target (denoted by position points 3 and 4). Due to this fact, the origin uncertainty of obtained position candidates should be discriminated and this issue forms the topic of this section. In addition, such a problem becomes harder to handle in the presence of clutter.

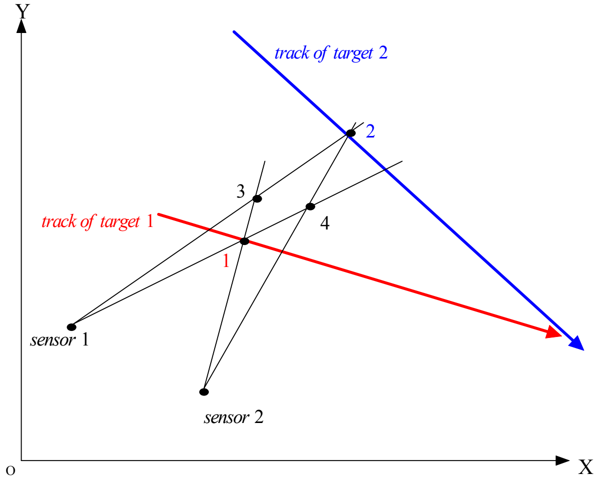


Fig. 2. The generated "ghosts" in case of two-sensor-two-target BOT

3.2 Motive

In the image detection field, the Hough transform (H-T) has been recognized as a robust technique for line or curve detection and also have been largely applied by scientific community (Bhattacharya, 2002; Shapiro, 2005). The basic idea of H-T is to transform a point (x, y) in the Cartesian coordinate system onto a curve in the (ρ, θ) parameter space, which is formulated as

$$\rho = x \cos\theta + y \sin\theta \tag{9}$$

where ρ is the distance from the line through (x, y) to the origin, and θ is the angle to the normal with the x axis. The angle θ varies from 0^0 to 180^0, while the ρ may be either positive or negative.

So, it is observed that, if a set of points in the Cartesian coordinate lie on the same line, all curves each corresponding to a point must intersect at a same point denoted by (ρ_0, θ_0) in the parameter space. Inspired by this phenomenon, the H-T technique can be utilized to initialize the track of target which makes a uniform rectilinear motion.

3.3 Solution to Multi-Target Track Initiation by ACO

In this section, we will investigate the problem of multi-target track initiation. First, a objective function is presented to describe the property of the multi-target track initiation. Second, a novel ACO algorithm, called different tasks of ants, is modelled to initiate the tracks of interest.

As noted before, if there are n curves in the parameter space, at most C_n^2 intersections are obtained in general. However, in a real tracking scenario, these curves will not strictly intersect the point but several points distributed in the parameter space due to the existence of measurement error. Even so, these points are still distributed in a small region, and thus such a small area could be deemed as an objective function to be optimized.

For the case of two given tracks, the corresponding intersections in the parameter space are plotted in Fig.3, and for the upper left expanded subfigure, which corresponds to target 1, the minimum and maximum values of θ could be obtained and then denoted by $\theta_{\min}$ and $\theta_{\max}$, respectively. Similarly, the related minimum and the maximum values of ρ are also found and denoted by $\rho_{\min}$ and $\rho_{\max}$, respectively.

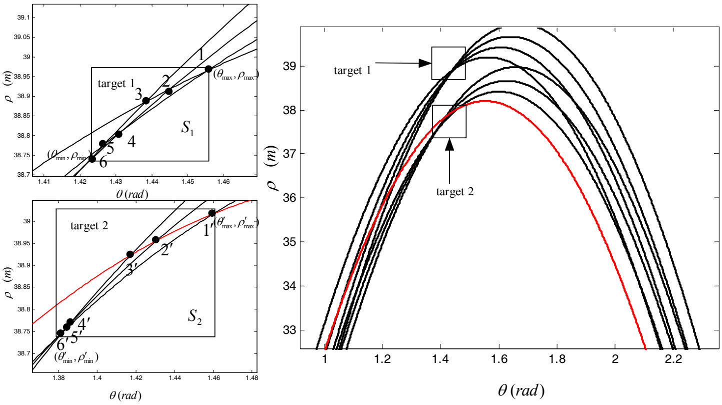


Fig. 3. A case of determination of objective function in the parameter space

As a result, two rectangular blocks are formed and the area of each is calculated as

$$S_i \triangleq (\theta^i_{\max} - \theta^i_{\min}) \bullet (\rho^i_{\max} - \rho^i_{\min}), \tag{10}$$

and the objective function J is defined as

$$\begin{aligned} & J = \min \sum_{r=1}^{M} S_{r(r_1 - r_2 - r_3 - r_4)} \\ & s.t \quad r_k \neq m_k \qquad \forall r_1 - r_2 - r_3 - r_4, m_1 - m_2 - m_3 - m_4 \in \Theta, \\ & \qquad k = 1, \dots, 4; \end{aligned} \tag{11}$$

where $r_1 - r_2 - r_3 - r_4$ or $m_1 - m_2 - m_3 - m_4$ is the possible track in the track space Θ, M is the number of tracks to be initialized.

Afterwards, the ants of different tasks will be investigated, and it has the following characteristics:

1) The number of tasks is equal to the one of tracks to be initiated, or equal to the one of targets of interest.
2) The traditional ACO algorithm builds solutions in an incremental way, but the proposed system of different tasks of ants builds solutions in parallel way. Especially, in the proposed system of ants of different tasks, the thought of both collaboration and competition between ants is considered and introduced. For instance, ants of the same task search for foods in a collaborative way, while ants of different tasks will compete with each other during establishing solutions.

3) Ants of the same task are dedicated to finding their best solution, and a set of all best solutions found by ants of different tasks constitute the solutions to Eq. (11) we describe.
4) In the system of ants of different tasks, the search space depends not only on the measurement returns at the next scan but also on the prior knowledge of target motion.

The determination of search space

In the case of bearings-only two-sensor-M-target tracking, the sampling data of the first four scans are utilized sequentially to initiate tracks, and then total four search spaces, i.e., Ω_1, Ω_2, Ω_3, and Ω_4, are obtained sequentially. Suppose that the prior knowledge about target motion, such as the minimum and maximum velocities denoted by $v_{\min}$ and $v_{\max}$ respectively, is known and then utilized to construct an annular region whose inner and outer radiuses are determined by $r_1 = \| \boldsymbol{v}_{\min} \| \bullet T$ and $r_2 = \| \boldsymbol{v}_{\max} \| \bullet T$, respectively, where T denotes the sampling interval. For instance, if an ant is now located at position i in Ω_1, then the ant will visit the next position located in the shadow section covered by both the annular region and Ω_2, which is denoted by $\tilde{\Omega}_2^i$ in Fig.4.

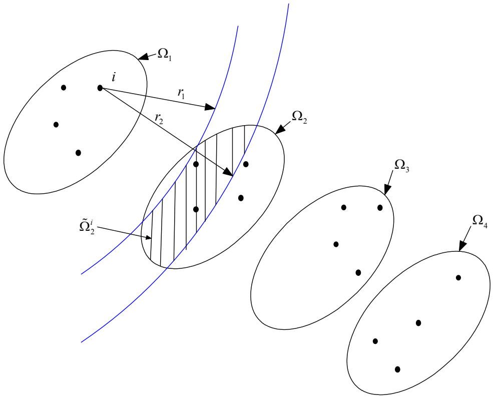


Fig. 4. The determination of search spaces

Track Candidate Construction Using the Ants of Different Tasks

Initially, $\tilde{M}$ ants of different tasks are placed randomly on position candidates in the first search space Ω_1, then each ant of a given task visits probabilistically the position candidate in the next search space. Suppose that an ant of a given task s is now located at position i in $\tilde{\Omega}_{\tilde{i}}^i$ ($1 \le \tilde{i} \le 3$), then the ant will visit position j in the next search space by applying the following probabilistic formula:

$$j = \begin{cases} \underset{j \in \tilde{\Omega}^{i}_{\tilde{i}+1}}{\arg\max} \left\{ \left[\dfrac{\tau^{s}_{i,j}}{\tau_{i,j}} \right]^{\alpha} \bullet \left[\eta_{i,j} \right]^{\beta} \bullet \left[\dfrac{1}{\tau_{i,j} - \tau^{s}_{i,j}} \right]^{\gamma} \right\} & \text{if } q \le q_0 \quad , \\ \bar{J} & \text{otherwise} \end{cases} \tag{12}$$

and $\bar{J}$ is a random variable selected according to the following probability distribution

$$P(j) = \begin{cases} \dfrac{\left[\dfrac{\tau^{s}_{i,j}}{\tau_{i,j}} \right]^{\alpha} \bullet \left[\eta_{i,j} \right]^{\beta} \bullet \left[\dfrac{1}{\tau_{i,j} - \tau^{s}_{i,j}} \right]^{\gamma}}{\sum_{l \in \tilde{\Omega}^{i}_{\tilde{i}+1}} \left[\dfrac{\tau^{s}_{i,l}}{\tau_{i,l}} \right]^{\alpha} \bullet \left[\eta_{i,l} \right]^{\beta} \bullet \left[\dfrac{1}{\tau_{i,l} - \tau^{s}_{i,l}} \right]^{\gamma}} & \text{if } j \in \tilde{\Omega}^{i}_{\tilde{i}+1} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where $\tau^{s}_{i,j}$ denotes the pheromone amount deposited by ants of task s on trail (i, j), $\tau_{i,j}$ is the total pheromone amount deposited by all ants of different tasks on trail (i, j), γ shows the repulsion on the foreign pheromones left on the trail (i, j), q is a random number uniformly distributed between 0 and 1, and q_0 is a parameter which determines the relative importance of the exploitation of good solutions versus the exploration of search spaces.

According to the search spaces discussed above, Fig. 5 plots the process of how the heuristic value is calculated from search spaces Ω_1 to Ω_2, namely, if an ant will move from positions i to j, the corresponding heuristic value can be defined as

$$\eta_{i,j} = \exp\left(-\frac{(d_{i,j} - r_0)^2}{2(r_2 - r_1)^2} \right), \tag{14}$$

where $d_{i,j}$ denotes the distance between positions i and j, and r_0 is equal to $(r_2 - r_1)/2$. Note that if position j falls out of $\tilde{\Omega}^{i}_{2}$, we set $\eta_{i,j} = 0$, and the search failure is declared for the current ant.

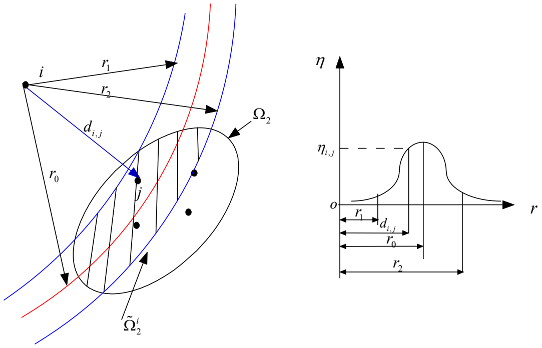


Fig. 5. The calculation of heuristic value

Update of Pheromone

The pheromone update is performed in two phases, namely, local update and global update. While building a solution, if an ant of task s carries out the transition from positions i to j, then the pheromone level of the corresponding trail is changed in the following way:

$$\tau_{i,j}^{s} \leftarrow (1-\rho)\tau_{i,j}^{s} + \tau_{0}^{s} \text{ ,} \tag{15}$$

where τ_{0}^{s} is the initial pheromone level of ants of task s.

Once all ants of different tasks at a given iteration have visited four candidate positions each from different sampling indices, the pheromone amount on each established track will be updated globally. Here, we use the best-so-far-solution found by ants of the same task, i.e. the best solution found from the start of the algorithm run, to update the corresponding pheromone trail. We adopt the following rule

$$\tau_{i,j}^{s} \leftarrow (1-\rho)\tau_{i,j}^{s} + \sum_{k=1}^{p} \Delta\tau_{i,j}^{s,k} \text{ .} \tag{16}$$

where $\Delta\tau_{i,j}^{s,k}$ is the pheromone amount that ant k of task s deposits on the trail (i,j) it has traveled at the current iteration, and p is the number of ants. In the case of bearings-only multi-sensor-multi-target tracking, $\Delta\tau_{i,j}^{s,k}$ is set to a constant.

4. A Comparison of ACO-Based Methods for Track Initiation

4.1 The Problem

Two cases are investigated here, namely two and three tracks' initiation problems. For each scenario, the performance of track initiation is investigated both in clutter-free environments and in clutter environments, respectively.

Two fixed sensors used to measure the targets' bearings are located at $(0,0)$ and $(18km,0)$ respectively in a surveillance region. The standard deviation of the bearing measurements for each sensor is taken as 0.1^{0}, and the sampling interval is set to be $T=10s$. The case in which each target makes a uniform rectilinear motion is considered, and the initial state of each target is illustrated in Table 1.

Scenarios	Targets	x (km)	y (km)	$\dot{x}$ (m/s)	$\dot{y}$ (m/s)
1	1	60	30	50	-100
	2	80	60	150	-150
2	1	60	30	50	-100
	2	80	60	150	-150
	3	60	50	80	-120

Table 1. The initial position and velocity of each target in the two considered scenarios

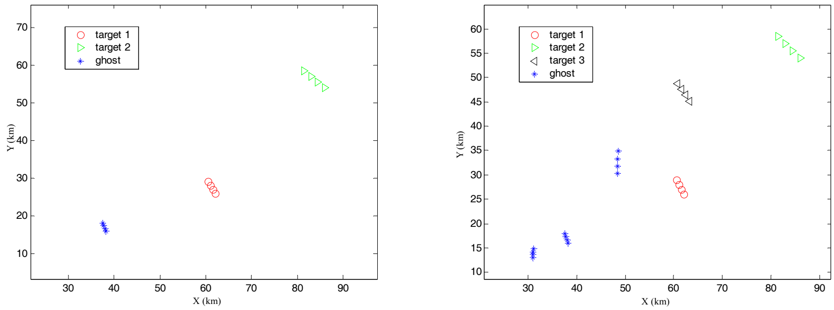


Fig. 6. The target position candidates in a “clutter-free” environment (left: Scenario 1, right: Scenario 2)

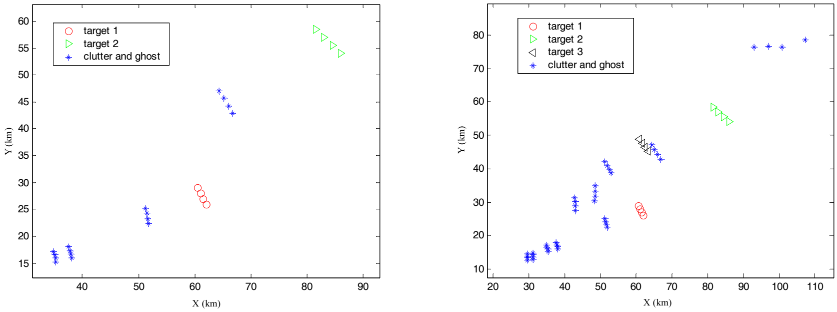


Fig. 7. The target position candidates in clutter environments (left: Scenario 1, right: Scenario 2)

Figs.6 and 7 depict a part of position candidates obtained by intersecting LOSs at each scan, and our object is to discriminate the true “positions” of each target of interest. Here, we use two ACO-based techniques, namely the Ant System (called the traditional ACO) and the system of ants of different tasks (called the proposed ACO).

Other parameters related to the two ACO-based methods are illustrated in Table 2

Parameter Value		Parameter Value	
α	0.01	$\Delta\tau$	0.03
β	0.2	$\tilde{M}$	$3M^2$
γ	2	$\lvert v_{\min} \rvert$	$100m/s$
ξ	0.8	$\lvert v_{\max} \rvert$	$400m/s$
q_0	0.7	$\lvert a_{\max} \rvert$	$15m/s^2$
τ_0	0.05	N_θ	50

Table 2. The Parameter Settings for ACO-related Methods

4.2 Evaluation Indices

Two performance indices are introduced to evaluate the system of ants of different tasks, i.e.

The probability of false track initiation: assuming $\tilde{N}$ Monte-Carlo runs are performed, we define the probability of false track initiation as

$$F \triangleq \sum_{i=1}^{\tilde{N}} f_i \Big/ \sum_{i=1}^{\tilde{N}} n_i \, , \tag{17}$$

where f_i denotes the number of false initiated tracks at the i th Monte-Carlo run, and n_i is the total number of initiated tracks.

The probability of correct initiation of at least j tracks: if at least j ($1 \le j \le M$) tracks are initiated correctly, its corresponding probability is

$$C_j \triangleq \sum_{i=1}^{\tilde{N}} l_{ij} \Big/ \tilde{N} \, , \tag{18}$$

where l_{ij} is a binary variable and defined as

$$l_{ij} = \begin{cases} 1 & \text{if at least j tracks are initiated correctly} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

at the i th Monte-Carlo run.

4.3 Results

All results in Tables 3 to 6 are averaged over 10,000 Monte-Carlo runs. According to the evaluation indices we introduce, the traditional ACO algorithm performs as well as the proposed one, as illustrated in Tables 3 and 4, in clutter-free environments. However, in the presence of clutter, the proposed ACO algorithm shows a significant improvement over the traditional one with respect to the probability of false track initiation, as shown in Tables 5 and 6.

Evaluation indices		The traditional ACO	The proposed ACO
Pro. of false track initiation (F)		0.0001	0.0002
Pro. of correct initiation of at least j tracks(C_j)	C_1	1.0000	1.0000
	C_2	0.9998	0.9997

Table 3. Performance comparison for two-track-initiation problem in clutter-free environments

Evaluation indices		The traditional ACO	The proposed ACO
Pro. of false track initiation (F)		0.0048	0.0046
Pro. of correct initiation of at least j tracks(C_j)	C_1	1.0000	1.0000
	C_2	1.0000	1.0000
	C_3	0.9857	0.9861

Table 4. Performance comparison for three-track-initiation problem in clutter-free environments

Evaluation indices		The traditional ACO	The proposed ACO
Pro. of false track initiation (F)		0.0348	0.0107
Pro. of correct initiation of at least j tracks(C_j)	C_1	1.0000	1.0000
	C_2	0.9787	0.9997

Table 5. Performance comparison for two-track-initiation problem in clutter environments

Evaluation indices		The traditional ACO	The proposed ACO
Pro. of false track initiation (F)		0.0672	0.0380
Pro. of correct initiation of at least j tracks(C_j)	C_1	1.0000	1.0000
	C_2	0.9594	1.0000
	C_3	0.9267	0.9861

Table 6. Performance comparison for three-track-initiation problem in clutter environments

Among 10,000 Monte-Carlo runs, only the cases of all tracks being initiated successfully are investigated and called effective runs later. For the objectivity of comparison, we select the worst case, in which the maximum running time for each ACO algorithm is evaluated, from the effective runs.

Fig. 8 depicts the trends of objective function evolution with the increasing number of iterations in scenario 2. Compared with the traditional ACO algorithm, the proposed one requires fewer iterations for convergence in clutter-free or clutter environments. According to Tables 3 and 4, although the performance of the traditional ACO algorithm is comparable to that of the proposed one, we find that the proposed ACO one seems more practical due to less running time needed. Figs. 9 and 10 depict varying curves of pheromone on the true targets' tracks, it is observed that the amount of pheromone on each "true" track increases in a moderate way, which means most ants prefer choosing these tracks and regarded them as optimal solutions.

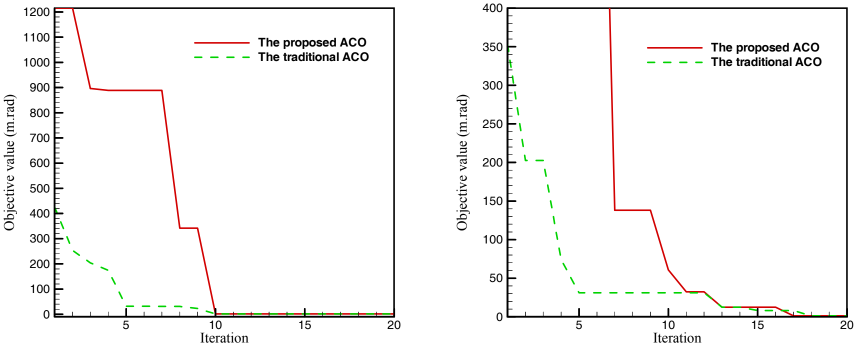


Fig. 8. Objective function curves (left: In clutter-free environments; right: In clutter environments)

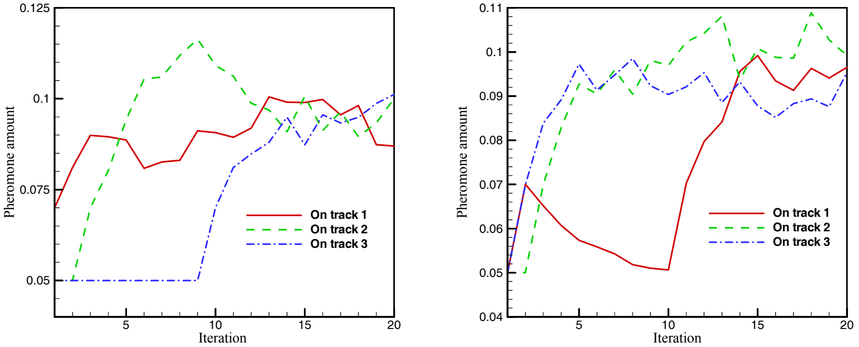


Fig. 9. Pheromone curves in clutter-free environments (left: The proposed ACO; right: The traditional ACO)

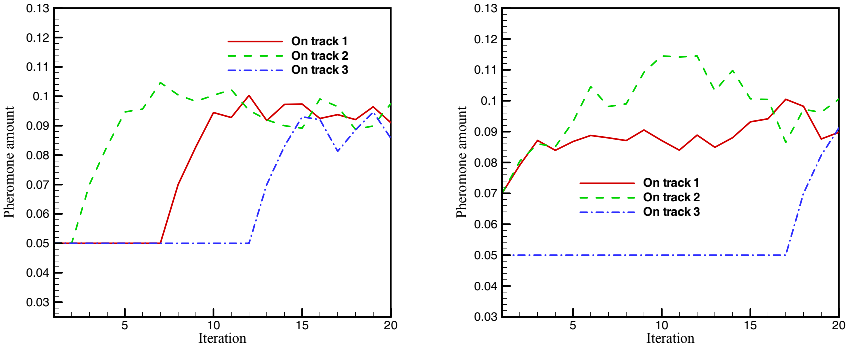


Fig. 10. Pheromone curves in clutter environments (left: The proposed ACO; right: The traditional ACO)

5. Conclusion

This chapter mainly aims to introduce some widely used ACO algorithms and their origins, such as the AS, EAS, MMAS, and so on. It is found that all concerns are focused on the pheromone update strategy. Some uses the best-so-far-ant or the iteration-best ant independently/interactively to update the trail that ants travelled. Meanwhile, the update law may differ a bit for different ACO algorithms. Among the four ACO algorithms, two versions have received great popularities in various applications, i.e. AS and MMAS. Another contribution in this chapter is the extension of the general ACO algorithm to the system of ants of different tasks, and its behaviour is modelled and implemented in the track initiation problems. Simulation results are also presented to show the effectiveness of the novel ACO algorithm. According to the example presented in this chapter, we believe that the general framework of AS can be modified to solve various optimal or non-optimal problems.

6. References

B. Bullnheimer; R. F. Hartl & C. Strauss. (1999). A new rank based version of the ant system: A computational study, *Central Eur. J. Oper. Res. Econ.*, Vol. 7, No. 1, 25–38, ISSN 1435-246X.

C. Blum. (2005). Beam-ACO—hybridizing ant colony optimization with beam search: An application to open shop scheduling, *Comput. Oper. Res.*, Vol. 32, No. 6, 1565–1591, ISSN 0305-0548.

David Martens; Manu De Backer & Raf Haesen. (2007). Classification With Ant Colony Optimization, *IEEE Trans. on Evolutional Computation*, Vol. 11, No. 5, October 2007, 651-665, ISSN 1089-778X.

Kutluyll Dogancay. (2004). On the bias of linear least squares algorithm for passive target localization, Signal Processing, Vol. 84, No. 3, 475-486, ISSN 0165-1684.

Kutluyll Dogancay. (2005). Bearings-only target localization using total least squares, *Signal Processing*, Vol. 85, No. 9,1695-1710, ISSN 0165-1684.

M. Dorigo; V. Maniezzo & A. Colorni. (1991). Positive Feedback as a Search Strategy, Technical Report 91–016, Politecnico di Milano, Milano, Italy.

M. Dorigo; V. Maniezzo & A. Colorni. (1996). The ant system: optimization by a colony of cooperating agents, *IEEE Trans. on System, Man, and Cybernetics-part B,* Vol.26, No. 1, 29-42, ISSN 1083-4419.

M. Dorigo & L. M. Gambardella. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Trans. on Evolutional Computation*, Vol.1, No. 1, 53-66, ISSN 1089-778X.

P. Bhattacharya; A. Rosenfeld & I. Weiss. (2002). Point-to-line mappings as Hough transforms, *Pattern Recognition Letters*, Vol. 23, No. 4, 1705-1710, ISSN 0167-8655.

Parag M. Kanade & Lawrence O. Hall. (2007). Fuzzy Ants and Clustering, *IEEE Trans. on System, Man, and Cybernetics-part A,* Vol. 37, No. 5, September 2007, 758-769,ISSN 1083-4427.

R. Montemanni; L. M. Gambardella; A. E. Rizzoli & A. Donati. (2005). Ant colony system for a dynamic vehicle routing problem, *J. Combinatorial Optim.*, Vol. 10, No. 4, 327–343, ISSN 1573-2886.

S.C. Nardone; A.G. Lindgren & K.F. Gong. (1984). Fundamental properties and performance of conventional bearings-only target motion analysis, *IEEE Transactions on Aerospace* and Electronic Systems, Vol. 29, No. 9, 775-787, ISSN 0018-9251.

T. Stützle & H. H. Hoos. (2000). $\mathcal{MAX}$-$\mathcal{MIN}$ ant system, *Future generation computer systems,* Vol.16, 889-914, ISSN 0167-739X.

V. Shapiro. (2006). Accuracy of the straight line Hough transform: the non-voting approach, *Computer Vision and Image Understanding*, Vol. 103, No. 1, 1-21, ISSN 1077-3142.

15

Mahalanobis Support Vector Machines Made Fast and Robust

Xunkai Wei[†‡], Yinghong Li[‡], Dong Liu[†], Liguang Zhan[†]
[†]*Beijing Aeronautical Engineering Technology Research Center*
[‡]*Air Force Engineering University*
CHINA

1. Introduction

As is known to us, common Euclidean distance based SVMs are easily influenced by outliers in given samples and might subsequently cause big prediction errors in testing processes. Therefore, many scholars propose various preprocessing methods such as whitening or normalizing the data to a sphere shape to remove the outliers and then call the routine SVM methods to build a more reasonable machine. However, since Euclidean distance is often sub-optimal especially in high dimension learning problem and might cause the learning machine fail due to the ill-conditioned Gram kernel, then it is necessary to find some more efficient and robust way to resolve the problem, which is the motivation of this chapter.

The Mahalanobis distance is superior to Euclidean Distance in handling with outliers and is widely used in statistics and machine learning area. Currently there are some methods in building SVMs combined with Mahalanobis distances. Some of them use it in the kernel and replace common kernel by a Mahalanobis one in SVMs. Some of them use it in preprocessing phase to remove the outlier first and then build SVMs using common methods. Others use it in the postprocessing phase to extract key support vectors for speedup and efficiency. Most of them achieve superior performances compared with SVM counterpart. However, it should be pointed out that the complexity of the combined algorithm is the most concerned factor in building such an algorithm.

As is known to us, none of them incorporates the Mahalanobis distance into models, which tradeoff the complexity and performance in the same algorithm meantime and make the algorithm more robust. The obvious feature of this new method is that there is no more necessary to remove the outlier first, since it is already considered and will be identified automatically in the model. It is also expected to improve and simplify the whole learning process efficiently.

One Class Classification (OCC) (Scholkopf, 2001) now becomes an active topic in machine learning domain. One Class Support Vector Machines (OCSVM) is firstly proposed via constructing a hyperplane in kernel feature space which separates the mapped patterns from the origin with maximum margin. Support vector domain description (SVDD) (Tax, 1999) is another popular OCC method, which seeks the minimum hypersphere that encloses all the data of the target class in a feature space. In this way, it finds the descriptive area that covers the data and excludes the superfluous space that results in false alarms existed in OCSVM.

However, although OCSVM does provide good representation for the classes of interest, it overlooks the discrimination issue between them. Moreover, the hypersphere model of SVDD is not flexible enough to give a tight description of the target class generally.

Therefore, in our previous works, we proposed two Mahalanobis distance based learning machine called QP-MELM and QP-MHLM respectively via solving their duals. However, as is suggested in (Löfberg, 2004), if both the primal form and dual form of an optimization problem are solvable, then the primal form is more commendable for approximation ability. Therefore, (Wei, 2007A) rewrote the MELM as a Second Order Cone Programming (SOCP) representable form and proposed a SOCP-MELM for class description. Applications to real world UCI benchmark datasets show promising results.

Recently, Wei etc al proposed a novel learning concept called enclosing machine learning (Wei, 2007D), which imitates the human being's cognition process, i.e. cognizing things of the same kind (To obtain a minimum bounding boundary for class description) and recognizing unknown things via point detection. Wei illustrated the concept using minimum volume enclosing ellipsoid learner for one class description and extended it to imbalanced data set classification. Except this, (Wang, 2005) and (Liu, 2006) proposed two SVDD based pattern classification algorithms (called SSPC and MEME respectively for simplicity) for imbalanced data set, which can also been classified to enclosing machine leaning's framework .

This chapter will be organized as follows. First, review of Mahalanobis distance\property and related learning methods will be briefed. Then, the new optimization models based on linear programming for Data Description, Classification incorporating Mahalanobis distance will be proposed. Third, benchmark datasets experiments for classification and regression will be investigated in detail. Finally, conclusions and discussions will be made.

2. The Mahalanobis Distance

2.1 Definitions

Let $\mathbf{X}$ be a $m \times N$ sample matrix containing N random observations $\mathbf{x}_i \in R^m, i = 1, 2, \cdots, N$. The sample mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ can be concisely expressed in terms of sample matrix.

$$\begin{cases} \boldsymbol{\mu} = \dfrac{1}{N}\mathbf{X1} \\ \boldsymbol{\Sigma} = \dfrac{1}{N}\mathbf{XX}^T - \dfrac{1}{N^2}\mathbf{X11}^T\mathbf{X}^T \end{cases} \tag{1}$$

where $\mathbf{1}$ is a $N-$dimensional all one vector.

If the covariance matrix is singular, it is difficult to calculate the inverse of $\boldsymbol{\Sigma}$. Instead, we can use the pseudoinverse $\boldsymbol{\Sigma}^+$ to approximate as $\boldsymbol{\Sigma}^+ = \mathbf{P}\boldsymbol{\Sigma}^{-1}\mathbf{P}^T$ using inverse of nonzero eigenvalues. This gives the minimum squared error approximation to the true solutions. It should be noted that pseudoinverse restricts inversion to the range of the operator, i.e. the subspace where it is not degenerate. This is often unavoidable in high dimensional feature spaces. If the covariance is real symmetric and positive semidefinite, then the covariance matrix can be decomposed as $\mathbf{\Sigma} = \mathbf{P}^T\mathbf{G}\mathbf{P}$, and thus $\mathbf{\Sigma}^{-1} = \mathbf{P}^T\mathbf{G}^{-1}\mathbf{P}$. Then the Mahalanobis distance from a sample $\mathbf{x}$ to the population $\mathbf{X}$ is

$$d^2(\mathbf{x},\mathbf{X}) = (\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \tag{2}$$

2.2 The Mahalanobis Distance in Kernel Feature Space

In implicit high-dimensional feature space defined by kernel functions, the Mahalanobis distance can be represented in terms of the dot products of data maps. Suppose $\mathbf{X}^\Phi, \boldsymbol{\mu}^\Phi, \mathbf{\Sigma}^\Phi$ are the sample matrix, mean vector, and covariance matrix in the feature space, respectively. The centered kernel matrix is defined as

$$\mathbf{K}_C = \mathbf{K} - \frac{1}{N}\mathbf{E}\mathbf{K} - \frac{1}{N}\mathbf{K}\mathbf{E} + \frac{1}{N^2}\mathbf{E}\mathbf{K}\mathbf{E} \tag{3}$$

where $\mathbf{E}$ is a $N \times N$ all one matrix, $\mathbf{K} = \left\{\Phi(\mathbf{x}_i)\bullet\Phi(\mathbf{x}_j) = k(\mathbf{x}_i,\mathbf{x}_j)\right\}_{i,j=1,\cdots,N}$ is a $N \times N$ symmetric matrix.

Using (1), we obtain

$$\mathbf{\Sigma}^\Phi = \mathbf{X}^{\Phi T}\mathbf{Z}^2\mathbf{X}^\Phi \tag{4}$$

where $\mathbf{Z} = (\frac{1}{N}(\mathbf{I} - \frac{1}{N}\mathbf{E}))^{\frac{1}{2}}$ is a $N \times N$ symmetric matrix, $\mathbf{I}$ is a $N \times N$ unit matrix.

The kernel Mahalanobis distance in the feature space can then be written as

$$\begin{aligned} d^2(\Phi(\mathbf{x}),\mathbf{X}^\Phi) &= (\Phi(\mathbf{x})-\boldsymbol{\mu}^\Phi)^T\mathbf{\Sigma}^{\Phi-1}(\Phi(\mathbf{x})-\boldsymbol{\mu}^\Phi) \\ &\overset{k(\bullet)}{=} (k(\mathbf{X},\mathbf{x}) - \frac{1}{N}\sum_{i=1}^{N}k(\mathbf{X},\mathbf{x}_i))^T \times (\mathbf{Z}\mathbf{M}^{-2}\mathbf{Z}) \\ &\times(k(\mathbf{X},\mathbf{x}) - \frac{1}{N}\sum_{i=1}^{N}k(\mathbf{X},\mathbf{x}_i)) \end{aligned} \tag{5}$$

where $\mathbf{K} = k(\mathbf{X}, \mathbf{X}^T)$, $\mathbf{x}_i$ is the i th sample of $\mathbf{X}$, $\mathbf{M} = \mathbf{ZKZ}$ is symmetric and semidefinite matrix and thus $\mathbf{M}^{-2}$ can be calculated via singular value decomposition, i.e. $\mathbf{M} = \mathbf{U\Lambda U}^T, \mathbf{M}^{-2} = \mathbf{U\Lambda}^{-2}\mathbf{U}^T$.

Using singular value decomposition method, we can easily conclude following theorem:
Theorem 1: Let the eigenstructures of the centered matrix $\mathbf{K}_C$ be $\mathbf{K}_C = \mathbf{Q}^T\mathbf{\Omega Q}$, then the covariance matrix $\mathbf{\Sigma}^\Phi$ can be diagonalized as follows:

$$\mathbf{\Sigma}^\Phi = (\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{QX}^{\Phi T})^T(\frac{1}{N}\mathbf{\Omega})(\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{QX}^{\Phi T}) \tag{6}$$

where N is the number of samples.
Proof: Recall that the covariance matrix $\mathbf{\Sigma}^\Phi$ in the feature space, and suppose $\mathbf{\Sigma}^\Phi$ could be decomposed via Singular Value Decomposition (SVD): $\mathbf{\Sigma}^\Phi = \mathbf{P}^T\mathbf{GP}$, we have

$$\begin{aligned}\mathbf{\Sigma}^\Phi &= \frac{1}{N}\sum_{i=1}^{N}(\Phi(\mathbf{x}_i) - \boldsymbol{\mu}^\Phi)(\Phi(\mathbf{x}_i) - \boldsymbol{\mu}^\Phi)^T \\ &= \frac{1}{N}\mathbf{X}^\Phi\mathbf{X}^{\Phi T} - \frac{1}{N^2}\mathbf{X}^{\Phi T}\mathbf{1}\mathbf{1}^T\mathbf{X}^\Phi = \mathbf{P}^T\mathbf{GP}\end{aligned} \tag{7}$$

Notice that the eigenvectors necessarily lie in the span of the centered data, thus $\mathbf{P}$ can be written as following linear combination

$$\mathbf{P} = \boldsymbol{\theta}(\mathbf{X}^{\Phi T} - \frac{1}{N}\mathbf{EX}^{\Phi T}) \tag{8}$$

where $\boldsymbol{\theta}$ is the coeffient matrix to be determined, $\mathbf{E}$ is a $N \times N$ all one matrix.
Multiplying (8) by $\mathbf{X}^{\Phi T} - \frac{1}{N}\mathbf{EX}^{\Phi T}$ from the left side, and by $\mathbf{P}^T$ from the right side, we have (after substituting $\mathbf{X}^{\Phi T}\mathbf{X}^\Phi = \mathbf{K}$)

$$\frac{1}{N}(\mathbf{K}_C)^2\boldsymbol{\theta}^T = \mathbf{K}_C\boldsymbol{\theta}^T\mathbf{G} \tag{9}$$

Multiplying (9) by pseudoinverse $\mathbf{K}_C^+$ from the left side, we have

$$\frac{1}{N}\mathbf{K}_C\boldsymbol{\theta}^T = \boldsymbol{\theta}^T\mathbf{G} \tag{10}$$

Since matrix $\mathbf{G}$ is diagonal, and the centered kernel matrix can be decomposed as

$$\mathbf{K}_C = \mathbf{Q}^T\boldsymbol{\Omega}\mathbf{Q} \tag{11}$$

We can obtain

$$\boldsymbol{\theta} = \mathbf{DQ} \tag{12}$$

$$\mathbf{G} = \frac{1}{N}\boldsymbol{\Omega} \tag{13}$$

where $\mathbf{D}$ is some diagonal matrix.
Since the eigenvectors of the covariance matrix are orthogonal, we have

$$\begin{aligned}\mathbf{I} &= \mathbf{PP}^T \\ &= \boldsymbol{\theta}(\mathbf{X}^{\Phi T} - \frac{1}{N}\mathbf{EX}^{\Phi T})(\mathbf{X}^{\Phi} - \frac{1}{N}\mathbf{X}^{\Phi}\mathbf{E})\boldsymbol{\theta}^T \\ &= \mathbf{DQK}_C\mathbf{Q}^T\mathbf{D} \\ &= \mathbf{D\Omega D}\end{aligned} \tag{14}$$

Therefore

$$\mathbf{D} = \boldsymbol{\Omega}^{-\frac{1}{2}} \tag{15}$$

Using the fact that

$$\begin{aligned}&\mathbf{K}_C = \mathbf{Q}^T\boldsymbol{\Omega}\mathbf{Q} \\ &\Rightarrow \mathbf{QK}_C = \boldsymbol{\Omega}\mathbf{Q} = \boldsymbol{\Omega}^{-1}\mathbf{QK}_C\mathbf{E} = \mathbf{QE} \\ &\Rightarrow \boldsymbol{\Omega}^{-1}\mathbf{Q}(\mathbf{K} - \frac{1}{N}\mathbf{EK} - \frac{1}{N}\mathbf{KE} + \frac{1}{N^2}\mathbf{EKE})\mathbf{E} = \mathbf{QE} \\ &\Rightarrow \boldsymbol{\Omega}^{-1}\mathbf{Q}(\mathbf{KE} - \frac{1}{N}\mathbf{EKE} - \mathbf{KE} + \frac{1}{N}\mathbf{EKE}) = \mathbf{QE} \\ &\Rightarrow \boldsymbol{\Omega}^{-1}\mathbf{Q0} = \mathbf{QE} = 0\end{aligned} \tag{16}$$

Thus from (8), using (12), (15) and (16), we have

$$\mathbf{P} = \mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Q}\mathbf{X}^{\Phi T} \tag{17}$$

Consequently, from (7), using (13), (17), we obtain

$$\mathbf{\Sigma}^{\Phi} = (\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Q}\mathbf{X}^{\Phi T})^{T}(\frac{1}{N}\mathbf{\Omega})(\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Q}\mathbf{X}^{\Phi T}) \tag{18}$$

And this ends the complete proof of Theorem 1.

According to Theorem 1, we can calculate the pseudoinverse $\mathbf{\Sigma}^{\Phi +}$ to approximate $\mathbf{\Sigma}^{\Phi -1}$ as

$$\mathbf{\Sigma}^{\Phi +} = N\mathbf{X}^{\Phi}\mathbf{Q}^{T}\mathbf{\Omega}^{-2}\mathbf{Q}\mathbf{X}^{\Phi T} \tag{19}$$

Thus (5) can be simplified as

$$\begin{aligned} d^{2}(\Phi(\mathbf{x}), \mathbf{X}^{\Phi}) = N(k(\mathbf{X}, \mathbf{x}) - \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{X}, \mathbf{x}_i))^{T} \\ \times(\mathbf{Q}^{T}\mathbf{\Omega}^{-2}\mathbf{Q}) \\ \times(k(\mathbf{X}, \mathbf{x}) - \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{X}, \mathbf{x}_i)) \end{aligned} \tag{20}$$

As we mentioned before, there often exists zero eigenvalues condition in high dimensional feature spaces, (7) is represented to approximate the true inverse of sample covariance matrix. Later, we will introduce another regularization method for avoiding the zero eigenvalues condition in Section 3.

3. Mahalanobis One Class SVMs

3.1 Mahalanobis One Class SVM

It is often beneficial to utilize the covariance matrix $\mathbf{\Sigma}$ and use the Mahalanobis distance instead. Given a set of unlabeled patterns $\{x_1, x_2, \cdots, x_N\}$, the OCSVM first maps them to the feature space $\mathbb{F}$ via a nonlinear map Φ. In the sequel, the OCSVM is obtained via solving

$$\min_{w,\xi_i\geq 0,\rho} \frac{1}{2}\mathbf{w}^T\mathbf{w}+\frac{1}{\nu N}\sum_{i=1}^{N}\xi_i-\rho \quad s.t.\begin{cases}\mathbf{w}^T x_i \geq \rho-\xi_i,\\ \xi_i \geq 0, i=1,2,\cdots,N.\end{cases} \tag{21}$$

where $\mathbf{w}^T x=\rho$ is the decision hyperplane, $\xi_i \geq 0$ is slack variable, N is the number of samples. Using kernel trick, the corresponding dual

$$\max_{\alpha_i\geq 0} \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} \quad s.t.\begin{cases}\mathbf{0}\leq\boldsymbol{\alpha}\leq\frac{1}{\nu N}\mathbf{1}\\ \boldsymbol{\alpha}\mathbf{1}=1.\end{cases} \tag{22}$$

is a quadratic convex optimization problem, where $\boldsymbol{\alpha}$ is the dual variable, $\mathbf{K}=\mathbf{X}^{\Phi T}\mathbf{X}^{\Phi}$ is a kernel matrix. By using the Mahalanobis distance metric instead of Euclidean distance metric, the primal now becomes:

$$\min_{w,\xi_i\geq 0,\rho} \frac{1}{2}\mathbf{w}^T\mathbf{\Sigma}^{-1}\mathbf{w}+\frac{1}{\nu N}\sum_{i=1}^{N}\xi_i-\rho \quad s.t.\begin{cases}\mathbf{w}^T\mathbf{\Sigma}^{-1} x_i \geq \rho-\xi_i,\\ \xi_i \geq 0, i=1,2,\cdots,N.\end{cases} \tag{23}$$

where $\mathbf{\Sigma}$ is the sample covariance matrix.

Using $\mathbf{w}=\mathbf{\Sigma}\mathbf{u}$, then the separation hyperplane is now $\mathbf{u}^T x=\rho$ and the distance to the origin becomes $\frac{\rho}{\sqrt{\mathbf{u}^T\mathbf{\Sigma}\mathbf{u}}}$. Therefore, (23) is equivalent to

$$\min_{\mathbf{u},\xi_i\geq 0,\rho} \frac{1}{2}\mathbf{u}^T\mathbf{\Sigma}\mathbf{u}+\frac{1}{\nu N}\sum_{i=1}^{N}\xi_i-\rho \quad s.t.\begin{cases}\mathbf{u}^T x_i \geq \rho-\xi_i,\\ \xi_i \geq 0, i=1,2,\cdots,N.\end{cases} \tag{24}$$

The dual is as follows using optimality conditions:

$$\max_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{X}^{\Phi T}\boldsymbol{\Sigma}^{-1}\mathbf{X}^{\Phi}\boldsymbol{\alpha}$$

$$s.t.\begin{cases} \mathbf{0} \le \boldsymbol{\alpha} \le \dfrac{1}{\nu N}\mathbf{1} \\ \boldsymbol{\alpha}^T\mathbf{1} = 1. \end{cases} \tag{25}$$

Note that the kernel trick is

$$\mathbf{X}^{\Phi T}\mathbf{X}^{\Phi} \overset{k(\bullet)}{=} \mathbf{K}, \tag{26}$$

Using the kernel trick and (21), the kernel Mahalanobis hyperplane learning machine can be written in kernel form as:

$$\max_{\boldsymbol{\alpha}} \frac{1}{2} N\boldsymbol{\alpha}^T \mathbf{K}\mathbf{Q}^T\boldsymbol{\Omega}^{-2}\mathbf{Q}\mathbf{K}\boldsymbol{\alpha}$$

$$s.t.\begin{cases} \mathbf{0} \le \boldsymbol{\alpha} \le \dfrac{1}{\nu N}\mathbf{1} \\ \boldsymbol{\alpha}^T\mathbf{1} = 1. \end{cases} \tag{27}$$

where the symbols are defined as previously noted.

We can easily conclude that (13) is a QP problem, thus can be efficiently solved via YALMIP[12].

As is mentioned before, there is uncertainty in the estimation of $\boldsymbol{\Sigma}$ in general. We can assume that $\boldsymbol{\Sigma}$ is only known to be within the set [5]

$$\{\boldsymbol{\Sigma} : \left\|\boldsymbol{\Sigma} - \boldsymbol{\Sigma}^0\right\|_F \le r\} \tag{28}$$

Suppose $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^0 + r\Delta\boldsymbol{\Sigma}$, then according to the Cauchy-Schwarz inequality [14], we get

$\mathbf{u}^T\Delta\boldsymbol{\Sigma}\mathbf{u} \le \|\mathbf{u}\|_2 \, \|\Delta\boldsymbol{\Sigma}\mathbf{u}\|_2 \le \|\mathbf{u}\|_2 \, \|\Delta\boldsymbol{\Sigma}\|_F \, \|\mathbf{u}\|_2 = \mathbf{u}^T\mathbf{u}$

This holds of compatibility of the Frobenius matrix norm and the Euclidean vector norm and because $\|\Delta\boldsymbol{\Sigma}\|_F \le 1$. For $\Delta\boldsymbol{\Sigma}$ the unity matrix, this upper bound is attained.

Thus we have

$$\begin{aligned}\max_{\mathbf{\Sigma}:\left\|\mathbf{\Sigma}-\mathbf{\Sigma}^0\right\|_F\le r}\mathbf{u}^T\mathbf{\Sigma}\mathbf{u} &= \mathbf{u}^T(\mathbf{\Sigma}^0 + r\max_{\|\Delta\mathbf{\Sigma}\|_F\le 1}\Delta\mathbf{\Sigma})\mathbf{u} \\ &= \mathbf{u}^T(\mathbf{\Sigma}^0 + r\mathbf{I})\mathbf{u}\end{aligned} \tag{29}$$

where $r > 0$ is fixed and $\left\|\bullet\right\|_F$ denotes the Frobenius norm, $\mathbf{\Sigma}^0$ is estimated via (1). Then the primal in (10) can be modified as

$$\begin{aligned}&\min_{\mathbf{u},\xi_i\ge 0,\rho}\max_{\mathbf{\Sigma}}\frac{1}{2}\mathbf{u}^T\mathbf{\Sigma}\mathbf{u}+\frac{1}{\nu N}\sum_{i=1}^{N}\xi_i-\rho \\ &s.t.\begin{cases}\mathbf{u}^T x_i\ge\rho-\xi_i, \\ \left\|\mathbf{\Sigma}-\mathbf{\Sigma}^0\right\|\le r, \\ \xi_i\ge 0, i=1,2,\cdots,N.\end{cases} \\ &\Rightarrow\min_{\mathbf{u},\xi_i\ge 0,\rho}\frac{1}{2}\mathbf{u}^T\mathbf{\Sigma}_r\mathbf{u}+\frac{1}{\nu N}\sum_{i=1}^{N}\xi_i-\rho \\ &s.t.\begin{cases}\mathbf{u}^T x_i\ge\rho-\xi_i, \\ \xi_i\ge 0, i=1,2,\cdots,N.\end{cases}\end{aligned} \tag{30}$$

where $\mathbf{\Sigma}_r = \mathbf{\Sigma}^0 + r\mathbf{I}$, now the sample covariance matrix is always nonsingular for $r > 0$. Actually, this is a regularization method.
The dual is now

$$\begin{aligned}&\max_{\boldsymbol{\alpha}}\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{X}^{\Phi T}\mathbf{\Sigma}_r^{-1}\mathbf{X}^{\Phi}\boldsymbol{\alpha} \\ &s.t.\begin{cases}\mathbf{0}\le\boldsymbol{\alpha}\le\dfrac{1}{\nu N}\mathbf{1} \\ \boldsymbol{\alpha}^T\mathbf{1}=1.\end{cases}\end{aligned} \tag{31}$$

By using the Woodbury formula [10]

$$(\mathbf{A}+\mathbf{BC})^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}(\mathbf{I}+\mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1} \tag{32}$$

and (4), we obtain

$$\begin{aligned}\mathbf{\Sigma}_r^{-1} &= (r\mathbf{I} + \mathbf{X}^{\Phi}\mathbf{Z}\mathbf{Z}\mathbf{X}^{\Phi T})^{-1} \\ &= \frac{1}{r}\mathbf{I} - \frac{1}{r}\mathbf{X}^{\Phi}\mathbf{Z}(r\mathbf{I} + \mathbf{Z}\mathbf{X}^{\Phi T}\mathbf{X}^{\Phi}\mathbf{Z})^{-1}\mathbf{Z}\mathbf{X}^{\Phi T}\end{aligned} \tag{33}$$

Using the kernel trick, (17) then becomes

$$\begin{aligned}&\max_{\boldsymbol{\alpha}} \frac{1}{2r}\boldsymbol{\alpha}^T(\mathbf{K} - \mathbf{K}\mathbf{Z}\mathbf{M}_r^{-1}\mathbf{Z}\mathbf{K})\boldsymbol{\alpha} \\ &s.t. \begin{cases}\mathbf{0} \le \boldsymbol{\alpha} \le \dfrac{1}{\nu N}\mathbf{1} \\ \boldsymbol{\alpha}^T\mathbf{1} = 1.\end{cases}\end{aligned} \tag{34}$$

where $\mathbf{M}_r = r\mathbf{I} + \mathbf{Z}\mathbf{K}\mathbf{Z}$. Again, we get a standard QP problem, and the inverse of the real symmetric and positive definite matrix $\mathbf{M}_r$ can be exactly estimated using stable and efficient eigenvalue decomposition method.

3.2 Mahalanobis Data Description Machine

Given a set of unlabeled patterns $\{x_1, x_2, \cdots, x_N\}$, the SVDD first maps them to the feature space $\mathbb{F}$ via a nonlinear map Φ. In the sequel, the SVDD is obtained via solving

$$\begin{aligned}&\min_{R,\xi_i \ge 0,\mathbf{c}} R^2 + C\sum_{i=1}^{N}\xi_i \\ &s.t. \begin{cases} d^2(\mathbf{c}, \mathbf{x}_i) \le R^2 + \xi_i, \\ \xi_i \ge 0, i = 1, 2, \cdots, N.\end{cases}\end{aligned} \tag{35}$$

where R is radius, $\xi_i \ge 0$ is slack variable, $\mathbf{c}$ is the center, $d(\bullet)$ is the given distance metric (the default is the Euclid norm) , C is a tradeoff that controls the size of sphere and the errors, N is the number of samples.

The corresponding dual

$$
\begin{aligned}
&\max_{\alpha_i \geq 0} \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\
&s.t. \begin{cases} 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, N \\ \sum_{i=1}^{N} \alpha_i = 1. \end{cases}
\end{aligned} \tag{36}
$$

is a convex optimization problem, where $\alpha_i \geq 0$ is dual variable, $k(\bullet)$ is a kernel function that satisfies Mercer condition.

By using the Mahalanobis distance metric instead of Euclidean distance metric, the primal now becomes:

$$
\begin{aligned}
&\min_{R, \xi_i \geq 0, \mathbf{c}} R^2 + C \sum_{i=1}^{N} \xi_i \\
&s.t. \begin{cases} (\mathbf{x}_i - \mathbf{c})^T \mathbf{\Sigma}^{-1} (\mathbf{x}_i - \mathbf{c}) \leq R^2 + \xi_i, \\ \xi_i \geq 0, i = 1, 2, \cdots, N. \end{cases}
\end{aligned} \tag{37}
$$

where $\mathbf{\Sigma}$ is the sample covariance matrix.

The dual is as follows using optimality conditions:

$$
\begin{aligned}
&\max_{\alpha_i \geq 0} \sum_{i=1}^{N} \alpha_i \mathbf{x}_i^T \mathbf{\Sigma}^{-1} \mathbf{x}_i - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{\Sigma}^{-1} \mathbf{x}_j \\
&s.t. \begin{cases} 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, N \\ \sum_{i=1}^{N} \alpha_i = 1. \end{cases}
\end{aligned} \tag{38}
$$

Using the kernel trick, and following equations,

$$
\mathbf{x}_i^T \mathbf{X} \overset{k(\bullet)}{=} k(\mathbf{x}_i, \mathbf{X}), \mathbf{c} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \,, \tag{39}
$$

the kernel Mahalanobis Ellipsoidal learning machine can be written in kernel form as:

$$\begin{aligned}\max_{\alpha_i \geq 0} &\sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{X})\mathbf{Q}^T \mathbf{\Omega}^{-2} \mathbf{Q} k(\mathbf{X}, \mathbf{x}_i)^T \\ &- \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{X})\mathbf{Q}^T \mathbf{\Omega}^{-2} \mathbf{Q} k(\mathbf{X}, \mathbf{x}_j) \\ s.t. &\begin{cases} 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, N \\ \sum_{i=1}^{N} \alpha_i = 1. \end{cases}\end{aligned} \tag{40}$$

Accordingly we can obtain the following Mahanlanobis distance of the sample $\mathbf{x}$ from the center $\mathbf{c}$ in the feature space:

$$\begin{aligned} d^2(\mathbf{x}^\Phi, \mathbf{c}^\Phi) &= N(\mathbf{x}^\Phi - \mathbf{c}^\Phi)^T \mathbf{X}^\Phi \mathbf{Q}^T \mathbf{\Omega}^{-2} \mathbf{Q} \mathbf{X}^{\Phi T} (\mathbf{x}^\Phi - \mathbf{c}^\Phi) \\ &= N(k(\mathbf{X}, \mathbf{x}) - \sum_{i=1}^{N} \alpha_i k(\mathbf{X}, \mathbf{x}_i))^T \mathbf{Q}^T \mathbf{\Omega}^{-2} \mathbf{Q} \\ &\quad \times (k(\mathbf{X}, \mathbf{x}) - \sum_{i=1}^{N} \alpha_i k(\mathbf{X}, \mathbf{x}_i)) \end{aligned} \tag{41}$$

The parameters R, ξ_i can be determined by the following relations via KKT conditions:

$$\left| \begin{array}{l} d^2(\mathbf{x}^\Phi, \mathbf{c}^\Phi) < R^2, \alpha_i = 0, \xi_i = 0 \\ d^2(\mathbf{x}^\Phi, \mathbf{c}^\Phi) = R^2, 0 < \alpha_i < C, \xi_i = 0 \\ d^2(\mathbf{x}^\Phi, \mathbf{c}^\Phi) = R^2 + \xi_i, \alpha_i = C, \xi_i > 0 \end{array} \right. \tag{42}$$

Again, we can assume that $\mathbf{\Sigma}$ is only known to be within the set

$$\{\mathbf{\Sigma} : \left\| \mathbf{\Sigma} - \mathbf{\Sigma}^0 \right\|_F \leq r\} \tag{43}$$

where $r > 0$ is fixed and called regularization constant and $\left\| \bullet \right\|_F$ denote s the Frobenius norm, $\mathbf{\Sigma}^0$ is estimated via (1).

Then the primal in (37) can be modified as

$$\min_{R,\xi_i\geq 0,\mathbf{c}} \max_{\mathbf{\Sigma}} R^2 + C\sum_{i=1}^{N}\xi_i$$
$$s.t.\begin{cases}(\mathbf{x}_i-\mathbf{c})^T\mathbf{\Sigma}^{-1}(\mathbf{x}_i-\mathbf{c})\leq R^2+\xi_i,\\ \xi_i\geq 0, i=1,2,\cdots,N,\\ \left\|\mathbf{\Sigma}-\mathbf{\Sigma}^0\right\|_F\leq r\end{cases} \tag{44}$$

Suppose $\mathbf{\Sigma}=\mathbf{\Sigma}^0+r\Delta\mathbf{\Sigma}$. Then for any given $\mathbf{v}$, according to the Cauchy-Schwarz inequality, we get

$$\mathbf{v}^T\Delta\mathbf{\Sigma}\mathbf{v}\leq\left\|\mathbf{v}\right\|_2\left\|\Delta\mathbf{\Sigma}\mathbf{v}\right\|_2\leq\left\|\mathbf{v}\right\|_2\left\|\Delta\mathbf{\Sigma}\right\|_F\left\|\mathbf{v}\right\|_2=\mathbf{v}^T\mathbf{v}$$

This holds of compatibility of the Frobenius matrix norm and the Euclidean vector norm and because $\left\|\Delta\mathbf{\Sigma}\right\|_F\leq 1$. For $\Delta\mathbf{\Sigma}$ the unity matrix, this upper bound is attained.

Thus for any given $\mathbf{v}$ and $\mathbf{\Sigma}^0$, we have

$$\begin{aligned}\min_{\mathbf{\Sigma}:\left\|\mathbf{\Sigma}-\mathbf{\Sigma}^0\right\|_F\leq r} \mathbf{v}^T\mathbf{\Sigma}^{-1}\mathbf{v}&=\mathbf{v}^T(\mathbf{\Sigma}^0+r\max_{\left\|\Delta\mathbf{\Sigma}\right\|_F\leq 1}\Delta\mathbf{\Sigma})^{-1}\mathbf{v}\\&=\mathbf{v}^T(\mathbf{\Sigma}^0+r\mathbf{I})^{-1}\mathbf{v}\end{aligned} \tag{45}$$

Therefore, (44) can be modified as

$$\min_{R,\xi_i\geq 0,\mathbf{c}} R^2 + C\sum_{i=1}^{N}\xi_i$$
$$s.t.\begin{cases}(\mathbf{x}_i-\mathbf{c})^T\mathbf{\Sigma}_r^{-1}(\mathbf{x}_i-\mathbf{c})\leq R^2+\xi_i,\\ \xi_i\geq 0, i=1,2,\cdots,N.\end{cases} \tag{46}$$

where $\mathbf{\Sigma}_r=\mathbf{\Sigma}^0+r\mathbf{I}$, now the sample covariance matrix is always nonsingular for $r>0$. Actually, this is a regularization method.

The dual is now

$$\max_{\alpha_i \geq 0} \sum_{i=1}^{N} \alpha_i \mathbf{x}_i^T \boldsymbol{\Sigma}_r^{-1} \mathbf{x}_i - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \mathbf{x}_i^T \boldsymbol{\Sigma}_r^{-1} \mathbf{x}_j$$
$$s.t. \begin{cases} 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, N \\ \sum_{i=1}^{N} \alpha_i = 1. \end{cases} \tag{47}$$

By using the Woodbury formula

$$(\mathbf{A} + \mathbf{BC})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I} + \mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1} \tag{48}$$

we obtain

$$\begin{aligned} \boldsymbol{\Sigma}_r^{-1} &= (r\mathbf{I} + \mathbf{X}^{\Phi}\mathbf{ZZX}^{\Phi T})^{-1} \\ &= \frac{1}{r}\mathbf{I} - \frac{1}{r}\mathbf{X}^{\Phi}\mathbf{Z}(r\mathbf{I} + \mathbf{ZX}^{\Phi T}\mathbf{X}^{\Phi}\mathbf{Z})^{-1}\mathbf{ZX}^{\Phi T} \end{aligned} \tag{49}$$

Using the kernel trick, (47) then becomes

$$\max_{\alpha_i \geq 0} \sum_{i=1}^{N} \alpha_i (k(\mathbf{x}_i, \mathbf{x}_i) - k(\mathbf{x}_i, \mathbf{X})\mathbf{ZM}_r^{-1}\mathbf{Z}k(\mathbf{X}, \mathbf{x}_i))$$
$$- \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j (k(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}_i, \mathbf{X})\mathbf{ZM}_r^{-1}\mathbf{Z}k(\mathbf{X}, \mathbf{x}_j)) \tag{50}$$
$$s.t. \begin{cases} 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, N \\ \sum_{i=1}^{N} \alpha_i = 1. \end{cases}$$

where $\mathbf{M}_r = r\mathbf{I} + \mathbf{ZKZ}$. Again, the inverse of the real symmetric and positive definite matrix $\mathbf{M}_r$ can be estimated via singular value decomposition.

Note that above mentioned models are both QP based, which might converge slowly in large dataset case. In order to further reduce the model complexity of above Mahalanobis Distance based SVM, here we introduce a new linear programming based model for ellipsoidal data description.

Let $\{\Phi(\mathbf{x}_i), i = 1, \cdots, N\}$ be the images of the samples in feature space through mapping Φ. We first center all the samples in feature space, and then we can build ellipsoidal machine centered at origin enclosing a majority of the imaged vectors. Then according to (5), the distance from any sample $\mathbf{x}$ to the origin can be kernelized as

$$d^2 = \varphi(x_i)\Sigma^{-1}\varphi(x_i)^T = \varphi(x_i)\Sigma^{+}\varphi(x_i)^T = \left\|\sqrt{N}\mathbf{\Omega}^{-1}\mathbf{Q}\mathbf{k}_i\right\|_2^2 \tag{51}$$

where $\mathbf{k}_i := (k(\mathbf{x}_1, \mathbf{x}_i), \cdots, k(\mathbf{x}_N, \mathbf{x}_i))$.

Therefore, now we can rewrite the primal form as:

$$\begin{aligned} &\min_{R^2, \xi_i \geq 0} R^2 + C\sum_{i=1}^{N} \xi_i \\ &s.t. \begin{cases} \left\|\sqrt{N}\mathbf{\Omega}^{-1}\mathbf{Q}\mathbf{k}_i\right\|_2^2 \leq R^2 + \xi_i, \\ \xi_i \geq 0, i = 1, 2, \cdots, N. \end{cases} \end{aligned} \tag{52}$$

The Lagrange function for the primal form will be as follows:

$$L(R^2, \xi_i, \alpha_i, \beta_i) = R^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \beta_i \xi_i - \sum_{i=1}^{N} \alpha_i (R^2 - \left\|\sqrt{N}\mathbf{\Omega}^{-1}\mathbf{Q}\mathbf{k}_i\right\|_2^2 + \xi_i) \tag{53}$$

where $\alpha_i, \beta_i \geq 0$ are Lagrange multipliers or the dual variables. According to the KKT conditions, and equating the partial derivatives of L with respect to R^2, ξ_i to zero yields:

$$\frac{\partial L}{\partial R^2} = 1 - \sum_{i=1}^{N} \alpha_i = 0 \tag{54}$$

$$\frac{\partial L}{\partial \xi_i} = C - \beta_i - \alpha_i = 0 \tag{55}$$

From (55), we get

$$\beta_i = C - \alpha_i \geq 0 \tag{56}$$

From (56) and using $\alpha_i, \beta_i \geq 0$, we can obtain

$$0 \leq \alpha_i \leq C \tag{57}$$

Using (57), and substituting (54), (55) into (53) results in the dual problem:

$$\min_{\alpha} -\sum_{i=1}^{N}\alpha_i \left\|\sqrt{N}\mathbf{\Omega}^{-1}\mathbf{Q}\mathbf{k}_i\right\|_2^2$$
$$s.t.\begin{cases}\sum_{i=1}^{N}\alpha_i = 1\\ 0 \le \alpha_i \le C\end{cases} \tag{58}$$

We can see that the optimization problem (58) is in a linear programming form. This LP form is superior to QP form in computational complexity.

From the solution of (58), the samples with $\alpha_i = 0$ will fall inside the ellipsoid. The samples with $\alpha_i > 0$ is called support vectors. Support vectors with $\alpha_i = C$ is called border support vectors. And the radius of the ellipsoid can be obtained using

$$R = \left\|\sqrt{N}\mathbf{\Omega}^{-1}\mathbf{Q}\mathbf{k}_{sv}\right\| \tag{59}$$

via using any border support vectors $\mathbf{x}_{sv}$.

Again, we assume that $\mathbf{\Sigma}$ is only known to be within the set $\{\mathbf{\Sigma} : \left\|\mathbf{\Sigma} - \mathbf{\Sigma}^0\right\|_F \le r\}$,where $r > 0$ is fixed and called regularization constant and $\| \ \|_F$ denotes the Frobenius norm. Then, the kernelized distance formula from any sample $\mathbf{x}$ to the origin is as follows:

$$d^2 = \varphi(x_i)\Sigma_r^{-1}\varphi(x_i)^T = \varphi(x_i)\Sigma_r^{+}\varphi(x_i)^T = \left\|(\frac{1}{N}\mathbf{\Omega} + r\mathbf{I})^{-\frac{1}{2}}\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Q}\mathbf{k}_i\right\|_2^2 \tag{59}$$

Therefore, the robust Mahalanobis data description in kernel form is

$$\min_{R^2,\xi_i \ge 0} R^2 + C\sum_{i=1}^{N}\xi_i$$
$$s.t.\begin{cases}\left\|(\frac{1}{N}\mathbf{\Omega} + r\mathbf{I})^{-\frac{1}{2}}\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Q}\mathbf{k}_i\right\|_2 \le R^2 + \xi_i,\\ \xi_i \ge 0, i = 1,2,\cdots,N.\end{cases} \tag{60}$$

where $\mathbf{k}_i := (k(x_1,x_i),k(x_2,x_i),\cdots,k(x_N,x_i))$.

And the dual form is

$$\min_{\alpha} -\sum_{i=1}^{N}\alpha_i \left\| (\frac{1}{N}\mathbf{\Omega}+r\mathbf{I})^{-\frac{1}{2}}\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Q}\mathbf{k}_i \right\|^2$$
$$s.t.\begin{cases}\sum_{i=1}^{N}\alpha_i = 1 \\ 0 \le \alpha_i \le C\end{cases} \tag{61}$$

Accordingly, we can get the ellipsoidal radius function in robust form for any sample $\mathbf{x}$,

$$f(\mathbf{x}) = \sqrt{N}\left\| (\frac{1}{N}\mathbf{\Omega}+r\mathbf{I})^{-\frac{1}{2}}\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Q}\mathbf{k}_{\mathbf{x}} \right\|_2 \tag{62}$$

4. Mahalanobis Classification SVMs

4.1 The main idea

Given a set of training data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i \in R^k, y_i \in \{1, -1\}$, $y_i = 1$ denotes target class, $y_i = -1$ denotes outlier class. Recall that SVM tries to find an optimum separation hyperplane by maximizing the margin as shown in Fig. 1(a). One can also try to find a hypersphere with a maximum separation shell which separates the target class from the outlier class shown in Fig.1 (b). Co-inspired by this idea, but instead of trying to find a sphere that provides a description of one class, for classification purposes, we would like to find a more compact and flexible hyper-ellipsoid that encloses all samples from one class (target class) but excludes all samples from the other class (outlier class). That is to say, we would like to find a hyper-ellipsoid $\mathrm{E}(R, \mathbf{c})$ that encloses all the target class samples and excludes all outlier class samples (see Fig.1 (c)). For the depicted example, we can see that our proposed MMEE method is tighter. Thus this can commendably reduce the risk of false alarms.

So as to guarantee the generalization performance, we also assume that hyper-ellipsoid centered at origin $\mathrm{E}(R, 0)$ separates the two classes with margin $2d$, i.e. it satisfies the following constraints

$$\begin{aligned} R^2 - \|\mathbf{x}_i\|_M^2 \ge d^2, \forall y_i{=}1 \\ \|\mathbf{x}_i\|_M^2 - R^2 \ge d^2, \forall y_i{=}\text{-}1 \end{aligned} \tag{63}$$

where d is the shortest distance from the hyper-ellipsoid to the closest target and outlier class samples, $\|z\|_M := z^T \mathbf{\Sigma}^{-1} z$ for any vector z. Note that the distance is now under Mahalanobis distance metric and d act s just as the margin of the SVM.

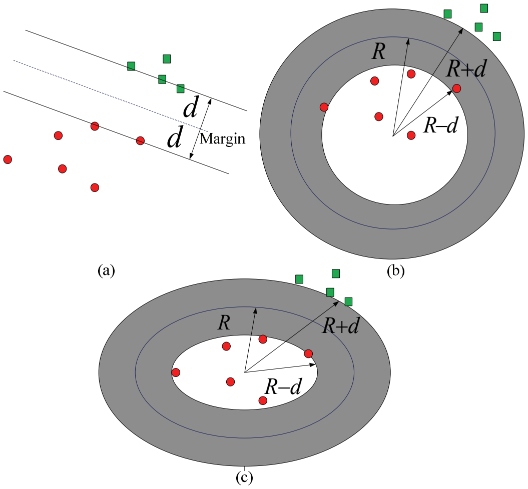


Fig. 1. Geometric illustrations of separation between two classes via different algorithms. (a) SVM. (b) Sphere shell separation (c) Ellipsoidal shell separation.

Obviously, there are many such hyper-ellipsoids which satisfy (63). An ideal criterion is to maximize the separation ratio $\frac{R+d}{R-d}$. But this objective is nonlinear and cannot be dealt with directly. Yet, it is easy to show that maximization of the separation ratio is equivalent to minimization of $\left(\frac{R}{d}\right)^2$. Using Taylor series formula, $\frac{R^2}{d^2}$ can be approximated as

$$\frac{R^2}{d^2} \approx \frac{R_0^2}{d_0^2} + \frac{1}{d_0^2}\left(R^2 - \frac{R_0^2}{d_0^2} d^2\right) \tag{64}$$

Now, the primal of Ellipsoidal shell separation in original space can be written as

$$\begin{aligned} &\min_{R^2, d^2} \; R^2 - \gamma d^2 \\ &s.t. \quad y_i \left(R^2 - \|\mathbf{x}_i\|_M^2\right) \geq d^2 \end{aligned} \tag{65}$$

where γ is a constant, which controls the ratio of the radius to the separation margin.

4.2 The Linear Programming Classification Machine

Introducing the kernel trick, the primal form can be rewritten as follows:

$$\begin{aligned} &\min_{R^2,d^2} \quad R^2 - \gamma d^2 \\ &s.t. \quad y_i\left(R^2 - \sqrt{N}\left\|\mathbf{\Omega}^{-1}\mathbf{Q}\mathbf{k}_i\right\|^2\right) \geq d^2 \end{aligned} \tag{66}$$

The robust kernelized primal version is

$$\begin{aligned} &\min_{R^2,d^2} R^2 - \gamma d^2 \\ &s.t. \ y_i\left(R^2 - \left\|(\frac{1}{N}\mathbf{\Omega} + r\mathbf{I})^{-\frac{1}{2}}\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Q}\mathbf{k}_i\right\|^2\right) \geq d^2 \end{aligned} \tag{67}$$

So as to allow misclassified samples, we introduce slack variables $\xi_i \geq 0$. Thus (66) can be modified as

$$\begin{aligned} &\min_{R^2,d^2,\xi_i} R^2 - \gamma d^2 + C\sum_{i=1}^{N}\xi_i \\ &s.t. \begin{cases} y_i\left(R^2 - \left\|\mathbf{x}_i\right\|_M^2\right) \geq d^2 - \xi_i \\ \xi_i \geq 0, i = 1, \cdots, N \end{cases} \end{aligned} \tag{68}$$

Accordingly, the kernelized primal version is

$$\begin{aligned} &\min_{R^2,d^2,\xi_i} R^2 - \gamma d^2 + C\sum_{i=1}^{N}\xi_i \\ &s.t. \begin{cases} y_i\left(R^2 - \sqrt{N}\left\|\mathbf{\Omega}^{-1}\mathbf{Q}\mathbf{k}_i\right\|^2\right) \geq d^2 - \xi_i \\ \xi_i \geq 0, i = 1, \cdots, N \end{cases} \end{aligned} \tag{69}$$

And the robust kernel version is

$$\min_{R^2,d^2,\xi_i} R^2 - \gamma d^2 + C\sum_{i=1}^{N}\xi_i$$
$$s.t.\begin{cases} y_i\left(R^2 - \left\|(\frac{1}{N}\mathbf{\Omega} + r\mathbf{I})^{-\frac{1}{2}}\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{Qk}_i\right\|^2\right) \geq d^2 - \xi_i \\ \xi_i \geq 0, i = 1,\cdots,N \end{cases} \tag{70}$$

In order to obtain the obtain the dual form, the for the primal form (69) will be as follows:

$$\begin{aligned} L(R^2,d^2,\xi_i,\alpha_i,\beta_i) = R^2 - \gamma d^2 + C\sum_{i=1}^{N}\xi_i - \\ \sum_{i=1}^{N}\alpha_i(y_i(R^2 - \sqrt{N}\left\|\mathbf{\Omega}^{-1}\mathbf{Qk}_i\right\|^2) - d^2 + \xi_i) - \sum_{i=1}^{N}\beta_i\xi_i \end{aligned} \tag{71}$$

According to the KKT conditions, we will get following equalities:

$$\frac{\partial L}{\partial R^2} = 1 - \sum_{i=1}^{N}\alpha_i y_i = 0 \tag{72}$$

$$\frac{\partial L}{\partial d^2} = -\gamma + \sum_{i=1}^{N}\alpha_i = 0 \tag{73}$$

$$\frac{\partial L}{\partial \xi_i} = C - \beta_i - \alpha_i = 0 \tag{74}$$

Using (72)-(74), the Lagrange function for the primal form (69) is simplified into following form:

$$\min_{\alpha} -\sum_{i=1}^{N} \alpha_i y_i \sqrt{N} \left\| \mathbf{\Omega}^{-1} \mathbf{Q}\mathbf{k}_i \right\|^2$$
$$s.t. \begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 1 \\ \sum_{i=1}^{N} \alpha_i = \gamma \\ 0 \le \alpha_i \le C \\ i = 1, \cdots, N \end{cases} \tag{75}$$

We can see that (75) is in a linear programming form. Therefore, we can easily solve it using any mature and stable LP solvers. And it is expected to be easily extended to large scale datasets.

Accordingly, we can also conclude the dual form for (70) as follows:

$$\min_{\alpha} -\sum_{i=1}^{N} \alpha_i y_i \left\| \left(\frac{1}{N}\mathbf{\Omega} + r\mathbf{I}\right)^{-\frac{1}{2}} \mathbf{\Omega}^{-\frac{1}{2}} \mathbf{Q}\mathbf{k}_i \right\|^2$$
$$s.t. \begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 1 \\ \sum_{i=1}^{N} \alpha_i = \gamma \\ 0 \le \alpha_i \le C \\ i = 1, \cdots, N \end{cases} \tag{76}$$

5. Applications

5.1 Mahalanobis One Class SVMs

We investigate the initial performances of our proposed Mahalanobis Ellipsoidal Learning Machine (MELM) using three real-world datasets (ionosphere, heart and sonar) from the UCI machine learning repository. To see how well the MELM algorithm performs with respect to other learning algorithms, we compared the OCSVM, SVDD and MOCSVM algorithm using Gaussian kernels $k(x, y) = \exp(-\gamma \|x - y\|^2)$ .As for the MOCSVM, we only use one single RBF kernel for performance comparison.

We treat each class as the "normal" data in separate experiments. We randomly choose 80% of points as training data and the rest 20% as testing data. We determined the optimal values of γ and C for RBF kernels by 5-fold cross validation. For the regularization constant, we set it as 0.01.

The datasets used and the results obtained by the four algorithms are summarized in Table 1. We can notice that the performance of our proposed Mahalanobis Ellipsoidal Learning Machine is competitive with or even better than the other approaches for the three datasets studied.

Dataset		SVDD	OCSVM	MOCSVM	MELM
ionosphe re	+1	66.02	65.26	66.05	68.98
	-1	69.13	68.96	70.99	75.82
heart	+1	70.13	70.01	69.96	71.19
	-1	71.11	70.23	71.78	75.37
sonar	+1	92.98	92.10	93.29	96.49
	-1	89.73	89.38	90.49	94.32

Table 1. Performance results of different algorithms for single class problems. Correctness ratio (%) is reported.

From Table 1, we also see that, on all 3 datasets, the results obtained by the Mahalanobis distance based learning algorithm are slightly better than the corresponding results of the other two Euclidean distance based methods.

5.2 Mahalanobis Classification Machine

We tested the new algorithm and compared it to standard SVMs using several real-world datasets from the UCI machine learning repository. The results of MMEE and other three algorithms SVM, SSPC and MEME depend on the values of the kernel parameter λ and the regularization parameter C. In addition, the performance of MMEE, SSPC and MMEE also depend on the constant that balances the volume and the margin. For simplicity, we set C to infinity for all four algorithms. Thus, we only considered hard margin MMEE, SSPC, MEME and SVM algorithms.

For all the datasets, we used the 5-fold cross-validation method to estimate the generalization error of the classifiers. In the 5-fold cross-validation process, we ensured that each training set and each testing set were the same for all algorithms, and the same Gaussian kernel $k(x, y) = \exp(-\lambda \|x - y\|^2)$ was used. On each dataset, the value of the kernel parameter λ for SVM was optimized to provide the best error rate using 5-fold cross-validation. As for MMEE, We investigate the robust version MMEE. And the regularization constant r was set as 0.03. For SSPC, MEME and MMEE, the kernel parameter λ and the constant that balances the volume and the margin were optimized using grid based 5-fold CV method.

The datasets used and the results obtained by all four algorithms are summarized in Table 1. As we can see, SSPC and MEME achieve the same or slightly better results than SVMs on all 5 datasets. But our proposed MMEE method shows promising performances. The accuracy is commendably higher than the other three methods in the datasets studied in this paper.

Dataset	SVM	SSPC	MEME	MMEE
Breast Cancer	4.26	4.26	4.25	4.05
Ionosphere	6.00	5.71	5.70	4.84
Liver	36.19	35.36	35.42.	33.89
Pima	35.13	34.90	34.91	28.56
Sonar	11.22	10.73	11.03	9.93

Table 2. Performance results of different algorithms. Error rate (%) is reported.

6. Conclusions

In this paper, we extended the support vector data description one class support vector machines via utilizing the sample covariance matrix information and using the Mahalanobis distance metric instead of Euclidean distance metric. The proposed Mahalanobis Ellipsoidal Learning Machine can be easily addressed as a robust optimization problem by introducing an uncertainty model into the estimation of sample covariance matrix. We propose a LP representable Mahalanobis Data Description Machine for one class classification. We also address a robust optimization problem by introducing an uncertainty model into the estimation of sample covariance matrix. The results of applications to the three UCI real world datasets show promising performances.

We also proposed a LP based Minimum Mahalanobis Enclosing Ellipsoid (MMEE) pattern classification algorithm for generally two class dataset classification. The MMEE method can be solved in kernel form of LP. We also address a robust optimization problem by introducing an uncertainty model into the estimation of sample covariance matrix. Initial applications to several UCI real world datasets show promising performances. The initial results show that the proposed methods own both good description and discrimination character for supervised learning problems. Moreover, the data description with non-hyperplane bounding decision boundary owns better discrimination performance than hyperplane counterpart in the context of supervising learning.

7. References

Abe, Shigeo (2005). Training of support vector machines with Mahalanobis kernels. In *Proceeding of ICANN 2005 Conference*, LNCS 3697, W. Duch et al. Eds. Springer-Verlag, Berlin Heidelberg, pp. 571–576

Blake, C.; Keogh, E., Merz, C. J. (1998). UCI repository of machine learning databases, University of California, Irvine, CA. Available at http://www. ics.uci.edu/~ mlearn/ML-Repository.html

Chapelle, O. (2007). Training a Support Vector Machine in the Primal. *Neural Computation*, Vol.19, 1155-1178

Lanckriet, G.; El Ghaoui, L., Jodan M. (2003). Robust novelty detection with single-class MPM. In *Advances in Neural Information Processing Systems 15*, S.Becker, S. Thrun, and K. Obermayor, Eds. MIT Press, Cambridge

Li, Yinghong; Wei, Xunkai. (2004). *Engineering applications of support vector machines,* Weapon Industry Press, Beijing, China

Liu, Y.; Zheng, Y.-F. (2006). Maximum Enclosing and Maximum Excluding Machine for Pattern Description and Discrimination. In: Proceeding of ICPR 2006 Conference. Vol. 3, 129-132

Löfberg, J. (2004). YALMIP: A Toolbox for Modeling and Optimization in MATLAB. In: *Proceedings of the CACSD Conference*

Ruiz, A.; Lopez-de-Teruel, P. E. (2001). Nonlinear Kernel-based Statistical Pattern Analysis. *IEEE Transactions on Neural Networks,* Vol.2, No.1, 16-32

Scholkopf, B.; Platt, J., Shawe-Taylor, J., Smola, A., Williamson, R. (2001). Estimating the Support of a High Dimensional Distribution. *Neural Computation*, Vol.13, No.7, 1443-1471

Tax, D.; Duin, R. (1999). Support Vector Domain Description. *Pattern Recognition Letters*, Vol.20, No.14, 1191-1199

Tsang, Ivor W.; Kwok, James T., and Li, Shutao. (2006). Learning the kernel in Mahalanobis one-class support vector machines. In *Proceeding of IJCNN 2006 Conference,* Vancouver, BC, Canada, 1169-1175

Wang, J.; Neskovic, P., Cooper, Leon N. (2005). Pattern Classification via Single Spheres. In: A. Hoffmann, H. Motoda, and T. Scheffer (eds.): *Proceeding of DS 2005 Conference*. LNAI, Vol. 3735, 241–252

Wei, X.-K; Huang, G.-B., Li, Y.-H. (2007A). Mahalanobis Ellipsoidal Learning Machine for One Class Classification. In: *2007 International Conference on Machine Learning and Cybernetics.* Vol. 6, 3528-3533

Wei, X.-K.; Huang, G.-B., Li, Y.-H. (2007B). A New One Class Mahalanobis Hyperplane Learning Machine based on QP and SVD. In LSMS2007

Wei, X.-K.; Li, Y.-H., Feng, Y., Huang, G.-B. (2007C) Solving Mahalanobis Ellipsoidal Learning Machine via Second Order Cone Programming. In: De-Shuang Huang, et al. (eds.): Proceeding of ICIC 2007 Conference. CCIS, Vol.2, 1186-1194

Wei, X.-K.; Li, Y.-H., Li, Y.-F. (2007D). Enclosing Machine Learning: Concepts and Algorithms. *International Journal of Neural Computing and Applications.* Vol. 17, No. 3 , 237-243

Wei, X.-K.; Löfberg J., Feng, Y., Li, Y.-H, Li, Y.-F. (2007E). Enclosing Machine Learning for Class Description. In: D. Liu et al. (eds.): *Proceedings of ISNN2007 Conference*. LNCS, Vol. 4491, 428-437

16

On-line learning of fuzzy rule emulated networks for a class of unknown nonlinear discrete-time controllers with estimated linearization

Chidentree Treesatayapun
Department of Robotic and Manufacturing, CINVESTAV-Saltillo, Ramos Arizpe, 25900 Mexico
Department of Electrical Engineering, North-Chiangmai University, Chiangmai, 50230 Thailand

1. Introduction

Recently, the linearization of a class of unknown discrete-time dynamic systems has achieved considerable topics for the controller design. The unknown functions after system linearization have been estimated by several methods including artificial intelligence techniques such as neural networks, fuzzy logic systems and neurofuzzy networks. In a number of published articles, the issues of system theoretic analysis have been introduced and addressed in the topics of stabilization, tracking performance and the bounded parameters. For all of these cases, the results are validated in the domain around the equilibrium point or state (9; 11). These methods of linearization including local linearization, Taylor series expansion and feedback linearization impose Lipschitz conditions (4; 6; 10; 14; 18). The closed-loop system stability and tracking error have been analyzed in the case of neural network adaptive control (5; 7) but during the learning phase the stability and convergence can not be ensured because of the special conditions. The system stability or bounded signals analysis has been verified (1; 13) and references therein. However, these nonlinear systems under control should be obtained in the format as $y(k+1) = f(k) + g(k)u(k)$ when $y(k)$ and $u(k)$ are the system output and the control input at time index k, respectively and $f(k)$ and $g(k)$ are unknown nonlinear functions. The small learning rate is often defined to solve the stability problem but the convergence is very slow. The discrete-time projection has been introduced for adaptive control systems in (16). The node number of multi-layer neural networks can take more effect of closed-loop stability and tracking performance. In (15), the unknown nonlinear part has been compensated by neural networks and the closed-loop system stability has been also guaranteed for a class on discrete-time systems. Nevertheless, this algorithm needs the renovation when the operating point is changed. In the case of robust system, the dead-zone function has been applied for feedback linearization systems (8) but this control algorithm are only limited for the system with slow trajectory tracking. In this chapter, we discuss about the controller for a class of nonlinear discrete-time systems with estimated unknown nonlinear functions by Muti-input Fuzzy Rules Emulated Networks (MIFRENs). These nonlinear functions are occurred when the control law is constructed and they are completely unknown as *a priori*. All adjustable parameters inside MIFRENs are automatically tuned by the proposed leaning algorithm. By the theoretical analysis, these parameters are all bounded during the system operation with out any request of off-line learning phase. The closed-loop tracking error is also bounded by the universal function approximation of MIFREN.

2. Preliminaries

2.1 Formulation of Nonlinear discrete-time systems

In this work, we devote our interest in to the discrete-time systems which can be described by

$$y(k+1) = f(p(k), u(k)), \tag{1}$$

where $f(\cdot,\cdot)$ is an unknown nonlinear function, k is time index, $y(k) \in R$ denotes the measurable output, $u(k) \in R$ is the control effort and $p(k) = [y(k), y(k-1), \dots, y(k-n+1), u(k-1), u(k-2), \dots, u(k-m+1)]$ when $m \leq n$. For system design in the next section, these following assumptions are still needed

Assumption : System derivative Let define two compact sets Ω_y and Ω_u for the system output y and the control effort u, respectively. The derivative of $f(\cdot,\cdot)$ in (1) with respect to the control effort $u(k)$ is always existed $\forall k = 1, 2, \cdots$ and $0 < |\frac{\partial f(\cdot,u)}{\partial u}| \leq \bar{y_u}$ when $y(\cdot) \in \Omega_y$ and $u(\cdot) \in \Omega_u$ where $\bar{y_u}$ is a finite positive value.

Assumption : Existence of controller For any desired trajectories $r(k)$, let the ideal control effort of the system (1) $u^*(k)$ be existed by

$$u^*(k) = g_u(p(k), r(k+1)), \tag{2}$$

when $g_u(\cdot,\cdot)$ is a smooth function.

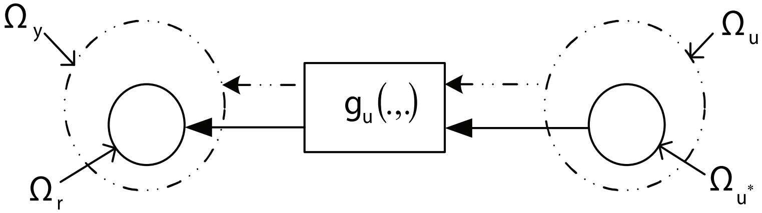


Fig. 1. Illustration of compact sets

With the ideal control effort obtained by (2), the controlled system can provide the output to be the desired trajectory as

$$r(k+1) = f(p(k), u^*(k)). \tag{3}$$

Let $u^*(k) \in \Omega_{u^*}$ and $r(k) \in \Omega_r$, for the output $y(k) \in \Omega_y$ such that $\Omega_r \subset \Omega_y$. The function $g_u(-)$ is a one-to-one mapping function of Ω_r into Ω_{u^*}, that is $\Omega_{u^*} \subset \Omega_u$. With the last assumption, $g_u(-)$ is smooth and Ω_r is a compact set, then Ω_{u^*} must be a compact set also. The clearly illustration is given in Fig. 1.

2.2 Function Approximation with MIFREN

In (2) and (3), the function approximation MIFREN property had been introduced. An unknown nonlinear function $f_u(.)$ can be estimated by MIFREN as

$$f_u(k) = \beta^T F_\mu(y(k), \cdots, y(k-\hat{n}-1), \cdots, u(k-\hat{m}-1)) + \varepsilon(k), \tag{4}$$

where β^T is the target linear parameter of MIFREN, $F_\mu(.)$ is the rule vector at MIFREN's rule-layer $\hat{n}$ and $\hat{m}$ are designed delay-order integers for y and u, respectively and $\varepsilon(k)$ stands for the MIFREN function approximation error. Eventually, the using function approximation result of MIFREN can be given as

$$\hat{f}_u(k) = \hat{\beta}^T(k) F_\mu(y(k), \cdots, y(k-\hat{n}-1), \cdots, u(k-\hat{m}-1)), \tag{5}$$

when $\hat{\beta}(k)$ is the actual linear parameter vector of MIFREN. The vector $\hat{\beta}(k)$ can be automatically tuned via the proposed algorithm as will be discussed in the next section. In this subsection, it will be shown that MIFREN has the property of a universal function approximation using the Stone-Weierstrass theorem (1; 17).

Theorem 2.1 (Universal function approximation of MIFREN)**.** *Let Ω be a compact space of N dimensions and let $\mathcal{F}$ be a set of real functions on a compact set Ω. If*

(1) $\mathcal{F}$ is an algebra,

(2) $\mathcal{F}$ separates points on Ω, and

(3) $\mathcal{F}$ vanishes at no point on Ω,

then $\mathcal{F}$ is dense in $C(\Omega)$, the set of continuous real-valued function on Ω. In other words, for any $\hat{\varepsilon} > 0$ and any function f in $C(\Omega)$, there is a function $\hat{f}$ in $\mathcal{F}$ such that $|f(x) - \hat{f}(x)| < \hat{\varepsilon}$ for all $x \in \Omega$.

Proof: The proof is omitted here moreover for the interested reader can refer to (2) and (3).

□

3. Controller design

In this section, the controller for system given in (1) is constructed with the approximated linearization and MIFRENs approximation.

3.1 Control law based on system linearization

From the system equation described in (1), let use the second-order Taylor expansion with the mean value theorem, we have

$$\begin{aligned} y(k+1) &= f(p(k), u(k-1)) + \frac{\partial f(p(k), u)}{\partial u}\Big|_{u=u(k-1)} \Delta u(k) \\ &\quad + \frac{1}{2}\frac{\partial^2 f(p(k), u)}{\partial u^2}\Big|_{u=\bar{u}_k} \Delta u^2(k), \end{aligned} \tag{6}$$

where $\bar{u}_k = \gamma u(k) + (1-\gamma)u(k-1)$ with $0 \le \gamma \le 1$ and $\Delta u(k) = u(k) - u(k-1)$. To simplify, (6) can be rewritten as

$$y(k+1) = f(p(k), u(k-1)) + f_1(p(k), u(k-1))\Delta u(k) + f_2(p(k), \bar{u}_k)\Delta u^2(k), \tag{7}$$

when $f_1(p(k),u(k-1)) = \frac{\partial f(p(k),u)}{\partial u}\Big|_{u=u(k-1)}$ and $f_2(p(k),\bar{u}_k) = \frac{1}{2}\frac{\partial^2 f(p(k),u)}{\partial u^2}\Big|_{u=\bar{u}_k}$. By using (2) and the second assumption mentioned in the previous section, $f_2(\cdot,\cdot)$ can be given by

$$f_2(p(k),\bar{u}_k)\Delta u^2(k) = f_2(p(k),\gamma u(k) + (1-\gamma)u(k-1))[u(k)-u(k-1)]^2, \tag{8}$$

when $u(k)$ can be obtained by

$$u(k) = g_u(p(k),y(k+1)). \tag{9}$$

Substitute (9) into (8), thus $f_2(\cdot,\cdot)$ can be simplified by

$$\begin{aligned} f_2(p(k),\bar{u}_k)\Delta u^2(k) &= f_2(p(k),\gamma g_u(p(k),y(k+1)) + (1-\gamma)u(k-1)), \\ &\quad \times[g_u(p(k),y(k+1)) - u(k-1)]^2, \\ &= \bar{f}_2(p(k),y(k+1)). \end{aligned} \tag{10}$$

By substituting (10) into (7), we have

$$\begin{aligned} y(k+1) &= f(p(k),u(k-1)) + f_1(p(k),u(k-1))\Delta u(k) + \bar{f}_2(p(k),y(k+1)), \\ &= f_3(p(k),y(k+1)) + f_1(p(k),u(k-1))\Delta u(k), \end{aligned} \tag{11}$$

where $f_3(p(k),y(k+1)) = f(p(k),u(k-1)) + \bar{f}_2(p(k),y(k+1))$. In (11), clearly, we have been forced with the causality problem. Fortunately, with the second assumption, the ideal control effort $u^*(k)$ can provide $r(k+1)$ as described in (3), thus we have

$$r(k+1) = f_3(p(k),r(k+1)) + f_1(p(k),u(k-1))[u^*(k) - u(k-1)]. \tag{12}$$

To continue our design procedure, the ideal control effort $u^*(k)$ can be obtained by

$$\begin{aligned} u^*(k) &= u(k-1) + \frac{r(k+1) - f_3(p(k),r(k+1))}{f_1(p(k),u(k-1))}, \\ &= u(k-1) + \frac{1}{f_1(p(k),u(k-1))}r(k+1) - \frac{f_3(p(k),r(k+1))}{f_1(p(k),u(k-1))}, \end{aligned} \tag{13}$$

or

$$u^*(k) = u(k-1) + f_1^*(p(k))r(k+1) - f_2^*(p(k),r(k+1)), \tag{14}$$

when $f_1^*(p(k)) = \frac{1}{f_1(p(k),u(k-1))}$ and $f_2^*(p(k),r(k+1)) = \frac{f_3(p(k),r(k+1))}{f_1(p(k),u(k-1))}$. From the control law given by (14), the singularity problem of $\frac{1}{f_1(p(k),u(k-1))}$ can be avoided by MIFREN approximation which will be discussed later. Let us consider the ideal control effort in (14), thus these nonlinear functions $f_1^*(\cdot,\cdot)$ and $f_2^*(\cdot,\cdot)$ are unknown. In this work, two MIFRENs are constructed to approximate $f_1^*(\cdot,\cdot)$ and $f_2^*(\cdot,\cdot)$ by MIFREN$_1$ and MIFREN$_2$, respectively. We have

$$u^*(k) = u(k-1) + [\beta_1^{*T}F_1(p(k)) + \varepsilon_1(k)]r(k+1) - \beta_2^{*T}F_2(p(k),r(k+1)) - \varepsilon_2(k), \tag{15}$$

where $F_1(\cdot)$ and $F_2(\cdot)$ are rule-functions of MIFREN$_1$ and MIFREN$_2$, respectively, $\beta_1^* = [\beta_{1,1}^* \quad \beta_{1,2}^* \quad \cdots \quad \beta_{1,n_1}^*]^T$, $\beta_2^* = [\beta_{2,1}^* \quad \beta_{2,2}^* \quad \cdots \quad \beta_{2,n_2}^*]^T$ are ideal weight vectors, n_1 and n_2 denote number of rules for each MIFREN and $\varepsilon_1(\cdot)$ and $\varepsilon_2(\cdot)$ are approximation errors. Let us neglect these errors and use the actual weight vector as $\beta_1(k)$ and $\beta_2(k)$ thus the proposed control law can be given by

$$u(k) = u(k-1) + [\beta_1^T(k)F_1(p(k))]r(k+1) - \beta_2^T(k)F_2(p(k),r(k+1)). \tag{16}$$

With this control equation, the causality problem has been solved by the MIFRENs approximation of unknown nonlinear functions. In the next subsection, the system performance will be analyzed with the designed parameters and main theorem.

3.2 Feedback system error

To guarantee the system performance, we need to design some parameters and theirs operating regions. Let the control error be defined by

$$e(k) = r(k) - y(k), \tag{17}$$

or

$$e(k+1) = r(k+1) - y(k+1), \tag{18}$$

for time index $k+1$. Substitute $y(k+1)$ from (11) into (18), we have

$$e(k+1) = r(k+1) - f_3(p(k),y(k+1)) - f_1(p(k))\Delta u(k). \tag{19}$$

By using Taylor expression and mean value theorem, the control error in (19) can be obtained as

$$\begin{aligned} e(k+1) \quad = \quad & r(k+1) - \Big[f_3(p(k),r(k+1)) + \frac{\partial f_3(p(k),y)}{\partial y}\Big|_{y=\bar{y}_{k+1}}(y(k+1) \\ & -r(k+1))\Big] - f_1(p(k))\Delta u(k), \end{aligned} \tag{20}$$

where $\bar{y}_{k+1}$ is between $r(k+1)$ and $y(k+1)$. Let us consider the system in (11) with the control effort given by (9), we have

$$y(k+1) = f_3(p(k),y(k+1)) + f_1(p(k))[g_u(p(k),y(k+1)) - u(k-1)]. \tag{21}$$

From (21), we can reconsider into two cases as these followings:

Case I In this case, we assume that $y(k+1) = r(k+1)$ and take the derivative with respect to $y(k+1)$ for the both sides of (21) thus we have

$$1 = \frac{\partial f_3(p(k),y(k+1))}{\partial y(k+1)} + f_1(p(k))\frac{\partial g_u(p(k),y(k+1))}{\partial y(k+1)}. \tag{22}$$

Case II For this second case, we reconsider (21) again with $y(k+1) \neq r(k+1)$, take the derivative with respect to $y(k+1)$ for the both sides of (21) and use Tayler expansion with the mean value theorem thus we have

$$\begin{aligned}
y(k+1) &= f_3(p(k),y(k+1)) + f_1(p(k))[g_u(p(k),y(k+1)) - u(k-1)], \\
&= f_3(p(k),r(k+1)) + \frac{\partial f_3(p(k),y)}{\partial y}\Big|_{y=\bar{y}(k+1)}[y(k+1) - r(k+1)] \\
&\quad + f_1(p(k))\Big[g_u(p(k),r(k+1)) + \frac{\partial g_u(p(k),y)}{\partial y}\Big|_{y=\breve{y}(k+1)} \\
&\quad \times [y(k+1) - r(k+1)] - u(k-1)\Big], \\
&= f_3(p(k),r(k+1)) + f_1(p(k))[g_u(p(k),r(k+1)) - u(k-1)] \\
&\quad + \frac{\partial f_3(p(k),y)}{\partial y}\Big|_{y=\bar{y}(k+1)}[y(k+1) - r(k+1)] \\
&\quad + f_1(p(k))\frac{\partial g_u(p(k),y)}{\partial y}\Big|_{y=\breve{y}(k+1)}[y(k+1) - r(k+1)].
\end{aligned} \tag{23}$$

From (12) and $u^*(k) = g_u(p(k),r(k+1))$, we can rearrange (23) to be

$$\begin{aligned}
y(k+1) - r(k+1) &= +\frac{\partial f_3(p(k),y)}{\partial y}\Big|_{y=\bar{y}(k+1)}[y(k+1) - r(k+1)] \\
&\quad + f_1(p(k))\frac{\partial g_u(p(k),y)}{\partial y}\Big|_{y=\breve{y}(k+1)}[y(k+1) \\
&\quad - r(k+1)].
\end{aligned} \tag{24}$$

With the previous assumption, we still have $\bar{y}(k+1) = \breve{y}(k+1)$ and $y(k+1) \neq r(k+1)$ thus (24) can be rewritten as

$$1 = \frac{\partial f_3(p(k),y(k+1))}{\partial y(k+1)} + f_1(p(k))\frac{\partial g_u(p(k),y(k+1))}{\partial y(k+1)}. \tag{25}$$

Taking the results from the both cases, the following relation can be obtained

$$\frac{\partial f_3(p(k),y(k+1))}{\partial y(k+1)} = 1 - f_1(p(k),u(k-1))\frac{\partial g_u(p(k),y(k+1))}{\partial y(k+1)}. \tag{26}$$

Substitute (26) into (20), we have

$$\begin{aligned}
e(k+1) &= r(k+1) - f_3(p(k),r(k+1)) + e(k+1) - f_1(p(k))\Delta u(k) \\
&\quad - f_1(p(k))\frac{\partial g_u(p(k),y)}{\partial y}\Big|_{y=\bar{y}(k+1)} e(k+1),
\end{aligned} \tag{27}$$

or

$$\begin{aligned}
f_1(p(k))\frac{\partial g_u(p(k),y)}{\partial y}\Big|_{y=\bar{y}(k+1)} e(k+1) &= r(k+1) - f_3(p(k),r(k+1)) \\
&\quad - f_1(p(k))\Delta u(k).
\end{aligned} \tag{28}$$

For the controllable system in (11), clearly, $f_1(p(k)) \neq 0$ and $u^*(k) = g_u(p(k), r(k+1))$ or $u(k) = g_u(p(k), y(k+1))$ thus the system sensibility $[\frac{\partial y}{\partial u}]^{-1}$ should be obtained as

$$\begin{aligned} \frac{\partial u(k)}{\partial y}\Big|_{y=y(k+1)} &= \frac{\partial g_u(p(k), y)}{\partial y}\Big|_{y=y(k+1)}, \\ &= \frac{1}{\gamma_y(k)}. \end{aligned} \tag{29}$$

The next time-index error can be rewritten again as

$$e(k+1) = \gamma_y(k)\Big[\frac{r(k+1)}{f_1(p(k))} - \frac{f_3(p(k), r(k+1))}{f_1(p(k))} - \Delta u(k)\Big]. \tag{30}$$

Rearrange (30) with MIFRENs approximation given by (15), we have

$$\begin{aligned} e(k+1) &= \gamma_y(k)\Big[\beta_1^{*T} F_1(p(k)) r(k+1) - \beta_2^{*T} F_2(p(k), r(k+1)) - \Delta u(k)\Big] \\ &\quad + \gamma_y(k)\Big[\varepsilon_1(k) r(k+1) - \varepsilon_2(k)\Big]. \end{aligned} \tag{31}$$

Substitute the proposed control law (16) into (31), we obtain

$$\begin{aligned} e(k+1) &= \gamma_y(k)\Big[\beta_1^{*T} F_1(p(k)) r(k+1) - \beta_2^{*T} F_2(p(k), r(k+1)) \\ &\quad - \beta_1^T(k) F_1(p(k)) r(k+1) + \beta_2^T(k) F_2(p(k), r(k+1))\Big] \\ &\quad + \gamma_y(k)\Big[\varepsilon_1(k) r(k+1) - \varepsilon_2(k)\Big], \\ &= \gamma_y(k)\Big[\widetilde{\beta}_1^T(k) F_1(k) r(k+1) - \widetilde{\beta}_2^T(k) F_2(k)\Big] + \gamma_y(k)\varepsilon_t(k), \end{aligned} \tag{32}$$

when $\widetilde{\beta}_i^T(k) = \beta_i^{*T} - \beta_i^T(k)$ for $i = 1, 2$ and $\varepsilon_t(k) = \varepsilon_1(k) r(k+1) - \varepsilon_2(k)$.

3.3 MIFRENs tuning laws

The parameter vectors $\beta_1(k)$ and $\beta_2(k)$ are required to update during the system operation or on-line learning. To simplify, let us rewrite (32) to be

$$e(k+1) = \gamma_y(k)\Big[\widetilde{\beta}_1^T(k) \quad \widetilde{\beta}_2^T(k)\Big] F(k) + \gamma_y(k)\varepsilon_t(k), \tag{33}$$

where $F(k) = \begin{bmatrix} F_1(k) r(k+1) \\ -F_2(k) \end{bmatrix}$. With (33), we can define the update law as the following:

$$\begin{bmatrix} \beta_1(k+1) \\ \beta_2(k+1) \end{bmatrix} = \begin{bmatrix} \beta_1(k) \\ \beta_2(k) \end{bmatrix} + \frac{\eta}{\bar{y_u} ||F(k)||^2} F(k) \mathfrak{D}(e(k)), \tag{34}$$

where η is the selected learning rate which will be discussed next and $\mathfrak{D}(\cdot)$ is the dead-zone function which can be defined by

$$\mathfrak{D}(e(k)) = \begin{cases} e(k) - \varepsilon_m & \text{if } e(k) > \varepsilon_m \\ 0 & \text{if } |e(k)| \leq \varepsilon_m \\ e(k) + \varepsilon_m & \text{if } e(k) < -\varepsilon_m, \end{cases} \tag{35}$$

when $|\gamma_y(k)\varepsilon_t(k)| \leq \varepsilon_m$ as a small positive number. In the case of $|e(k-1)| > \varepsilon_m$, with the dead-zone function (35) and the next time-index error (33), we have

$$\mathfrak{D}(e(k+1)) = \alpha_D \gamma_y(k) \left[\widetilde{\beta}_1^T(k) \quad \widetilde{\beta}_2^T(k) \right] F(k), \tag{36}$$

where $0 < \alpha_D \leq 1$.

3.4 System analysis

To analyze the system performance and stability, the bounded weight vectors $\widetilde{\beta}_i^T(k)$ and the bounded tracking error $e(k)$ are both given in this work.

Lemma 1: For the nonlinear discrete-time system given in (1) with the control law defined in (16), the error weight vectors $\widetilde{\beta}_i^T(k)$ for $i = 1,2$ are bounded by the tuning law in (34) and the selected learning rate η as the followings:

$$0 < \eta < \frac{2\bar{y_u}}{\alpha_D \gamma_y(k)}, \tag{37}$$

when $0 < \bar{y_u}$, and

$$\frac{2\bar{y_u}}{\alpha_D \gamma_y(k)} < \eta < 0, \tag{38}$$

when $\bar{y_u} < 0$.

Proof: Let us define a Lyapunov candidate function as

$$V_{\widetilde{\beta}}(k) = \widetilde{\beta}_1^T(k)\widetilde{\beta}_1(k) + \widetilde{\beta}_2^T(k)\widetilde{\beta}_2(k). \tag{39}$$

The first difference can be obtained by

$$\begin{aligned} \Delta V_{\widetilde{\beta}}(k) &= V_{\widetilde{\beta}}(k+1) - V_{\widetilde{\beta}}(k), \\ &= \widetilde{\beta}_1^T(k+1)\widetilde{\beta}_1(k+1) + \widetilde{\beta}_2^T(k+1)\widetilde{\beta}_2(k+1) - \widetilde{\beta}_1^T(k)\widetilde{\beta}_1(k) \\ &\quad -\widetilde{\beta}_2^T(k)\widetilde{\beta}_2(k). \end{aligned} \tag{40}$$

Let us define $\widetilde{\beta}_\Sigma(k+1) = \widetilde{\beta}_1^T(k+1)\widetilde{\beta}_1(k+1) + \widetilde{\beta}_2^T(k+1)\widetilde{\beta}_2(k+1)$, from the tuning law given by (34) and $\widetilde{\beta}_i^T(k) = \beta_i^{*T} - \beta_i^T(k)$, we have

$$\begin{aligned} \widetilde{\beta}_\Sigma(k+1) &= \widetilde{\beta}_1^T(k)\widetilde{\beta}_1(k) + \widetilde{\beta}_2^T(k)\widetilde{\beta}_2(k) - \frac{2\eta}{\bar{y_u}||F(k)||^2} \begin{bmatrix} \beta_1(k) \\ \beta_2(k) \end{bmatrix}^T F(k) \\ &\quad \times\mathfrak{D}(e(k+1)) + \frac{\eta^2}{\bar{y_u}^2||F(k)||^4}||F(k)||^2\mathfrak{D}^2(e(k+1)). \end{aligned} \tag{41}$$

Substitute (41) into (40) and use (36), we obtain

$$\begin{aligned} \Delta V_{\widetilde{\beta}}(k) &= -\frac{2\eta}{\bar{y_u}||F(k)||^2} \begin{bmatrix} \beta_1(k) \\ \beta_2(k) \end{bmatrix}^T F(k)\mathfrak{D}(e(k+1)) + \frac{\eta^2}{\bar{y_u}^2||F(k)||^4}||F(k)||^2 \\ &\quad \times\mathfrak{D}^2(e(k+1)) \\ &= -\frac{2\eta}{\alpha_D\gamma_y(k)\bar{y_u}||F(k)||^2}\mathfrak{D}^2(e(k+1)) + \frac{\eta^2}{\bar{y_u}^2||F(k)||^2}\mathfrak{D}^2(e(k+1)) \\ &= \left[\frac{-2}{\alpha_D\gamma_y(k)} + \frac{\eta}{\bar{y_u}}\right]\frac{\eta}{\bar{y_u}||F(k)||^2}\mathfrak{D}^2(e(k+1)). \end{aligned} \tag{42}$$

With the selected learning rate defined by (37) and (38) and $\gamma_y(k)$ given in (29), the first difference of Lyapunov function is negative, thus $\widetilde{\beta}_i^T(k)$ for $i = 1,2$ are bounded.

□

Remark: Normally, with out loss of generality, $\bar{y_u}$ is assumed to be positive thus $\gamma_y(k) < \bar{y_u}$: $\forall k$. The bounded tracking error for the closed-loop system is introduced by the following theorem.

Theorem 3.1 (Bounded tracking error)**.** *For the nonlinear discrete-time system given in (1) with the control law defined in (16), let define a compact set $\Omega_\varepsilon = \{e(k)||e(k) \leq 4\varepsilon_m\}$, thus the ultimate boundary on the tracking error is $\lim_{k\to\infty}|e(k)| \leq \varepsilon_m$ or in a compact set Ω_ε.*

Proof: Let a Lyapunov candidate function be given by

$$V_e(k) = \frac{\eta}{2\bar{y_u}^2 F_o^2} e^2(k) + V_{\widetilde{\beta}}(k), \tag{43}$$

when F_o is defined by $0 < ||F(k)|| \leq F_0, \forall k$. The first difference can be obtained by

$$\begin{aligned} \Delta V_e(k) &= V_e(k+1) - V_e(k), \\ &= \frac{\eta}{2\bar{y_u}^2 F_o^2}[e^2(k+1) - e^2(k)] + \Delta V_{\widetilde{\beta}}(k). \end{aligned} \tag{44}$$

Substitute (42) into (44), we have

$$\begin{aligned} \Delta V_e(k) &= \frac{\eta}{2\bar{y_u}^2 F_o^2}[e^2(k+1) - e^2(k)] - \frac{2\eta\mathfrak{D}^2(e(k+1))}{\alpha_D \gamma_y(k)\bar{y_u}||F(k)||^2} \\ &\quad + \frac{\eta^2\mathfrak{D}^2(e(k+1))}{\bar{y_u}^2||F(k)||^2}. \end{aligned} \tag{45}$$

From the learning rate given by (37- 37), we can rearrange (45) as

$$\begin{aligned} \Delta V_e(k) &< \frac{\eta}{2\bar{y_u}^2 F_o^2} e^2(k+1) - \frac{\eta}{\alpha_D \gamma_y(k)\bar{y_u}||F(k)||^2}\mathfrak{D}^2(e(k+1)), \\ &< \frac{\eta}{2\bar{y_u}^2 F_o^2} e^2(k+1) - \frac{\eta}{\bar{y_u}^2 F_o^2}\mathfrak{D}^2(e(k+1)), \\ &= \frac{\eta}{2\bar{y_u}^2 F_o^2}[e^2(k+1) - 2\mathfrak{D}^2(e(k+1))]. \end{aligned} \tag{46}$$

In this proof, we need to provide only the case when $|e(k+1)| > \varepsilon_m$. With $|e(k+1)| > \varepsilon_m$, the dead-zone function in (35) can be obtained as

$$\mathfrak{D}(e(k+1)) = e(k+1) - \varepsilon_m \mathrm{sign}\{e(k+1)\}. \tag{47}$$

Substitute (47) into (46), we have

$$\begin{aligned} \Delta V_e(k) &< \frac{\eta}{2\bar{y_u}^2 F_o^2}\Big[e^2(k+1) - 2[e(k+1) - \varepsilon_m \mathrm{sign}\{e(k+1)\}]^2\Big], \\ &= \frac{\eta}{2\bar{y_u}^2 F_o^2}\Big[-e^2(k+1) + 4e(k+1)\varepsilon_m \mathrm{sign}\{e(k+1)\} - 2\varepsilon_m^2\Big], \\ &= \frac{\eta}{2\bar{y_u}^2 F_o^2}\Big[-e^2(k+1) - 2\varepsilon_m^2 + 4|e(k+1)|\varepsilon_m\Big], \\ &< \frac{\eta}{2\bar{y_u}^2 F_o^2}\Big[-e^2(k+1) + 4|e(k+1)|\varepsilon_m\Big]. \end{aligned} \tag{48}$$

Consider the result in (48), clearly, $\Delta V_e(k)$ is always negative where $|e(k+1)| > 4\varepsilon_m$, thus $\Delta V_e(k) < 0$ when $|e(k+1)|$ is out side a compact set Ω_ε.

□

4. Computer simulation example

The proposed control algorithm and theorem are verified by the computer simulation. The selected controllable system is described by

$$y(k+1) = \sin(y(k)) + \cos(y(k)u(k))u(k) + 5u(k). \tag{49}$$

The system performance can be demonstrated into two cases as the nominal system and the robust control.

4.1 Nominal system

With the system displayed in (49), the system parameters can be selected as $\varepsilon_m = 0.001$, $\eta = 0.8$ and $\bar{y}_u = 5$. All IF-THEN rules for both MIFRENs are given by the followings:

MIFREN$_1$	Rule 1	If $y(k)$ is N	and $u(k-1)$ is N	Then $f_{1,1}(k) = \beta_{1,1}(k)F_{1,1}(k)$,
	Rule 2	If $y(k)$ is N	and $u(k-1)$ is Z	Then $f_{1,2}(k) = \beta_{1,2}(k)F_{1,2}(k)$,
	Rule 3	If $y(k)$ is N	and $u(k-1)$ is P	Then $f_{1,3}(k) = \beta_{1,3}(k)F_{1,3}(k)$,
	Rule 4	If $y(k)$ is Z	and $u(k-1)$ is N	Then $f_{1,4}(k) = \beta_{1,4}(k)F_{1,4}(k)$,
	Rule 5	If $y(k)$ is Z	and $u(k-1)$ is Z	Then $f_{1,5}(k) = \beta_{1,5}(k)F_{1,5}(k)$,
	Rule 6	If $y(k)$ is Z	and $u(k-1)$ is P	Then $f_{1,6}(k) = \beta_{1,6}(k)F_{1,6}(k)$,
	Rule 7	If $y(k)$ is P	and $u(k-1)$ is N	Then $f_{1,7}(k) = \beta_{1,7}(k)F_{1,7}(k)$,
	Rule 8	If $y(k)$ is P	and $u(k-1)$ is Z	Then $f_{1,8}(k) = \beta_{1,8}(k)F_{1,8}(k)$,
	Rule 9	If $y(k)$ is P	and $u(k-1)$ is P	Then $f_{1,9}(k) = \beta_{1,9}(k)F_{1,9}(k)$,

MIFREN$_2$	Rule 1	If $y(k)$ is N	and $r(k+1)$ is N	Then $f_{2,1}(k) = \beta_{2,1}(k)F_{2,1}(k)$,
	Rule 2	If $y(k)$ is N	and $r(k+1)$ is Z	Then $f_{2,2}(k) = \beta_{2,2}(k)F_{2,2}(k)$,
	Rule 3	If $y(k)$ is N	and $r(k+1)$ is P	Then $f_{2,3}(k) = \beta_{2,3}(k)F_{2,3}(k)$,
	Rule 4	If $y(k)$ is Z	and $r(k+1)$ is N	Then $f_{2,4}(k) = \beta_{2,4}(k)F_{2,4}(k)$,
	Rule 5	If $y(k)$ is Z	and $r(k+1)$ is Z	Then $f_{2,5}(k) = \beta_{2,5}(k)F_{2,5}(k)$,
	Rule 6	If $y(k)$ is Z	and $r(k+1)$ is P	Then $f_{2,6}(k) = \beta_{2,6}(k)F_{2,6}(k)$,
	Rule 7	If $y(k)$ is P	and $r(k+1)$ is N	Then $f_{2,7}(k) = \beta_{2,7}(k)F_{2,7}(k)$,
	Rule 8	If $y(k)$ is P	and $r(k+1)$ is Z	Then $f_{2,8}(k) = \beta_{2,8}(k)F_{2,8}(k)$,
	Rule 9	If $y(k)$ is P	and $r(k+1)$ is P	Then $f_{2,9}(k) = \beta_{2,9}(k)F_{2,9}(k)$,

when N, Z and P denote negative, zero and positive linguistic levels respectively. The membership functions for these rules are illustrated in Fig. (2) and (3). In this work, we use the same membership functions of $y(k)$ and $r(k+1)$ because these variables have equality linguistic levels in the sense of human.

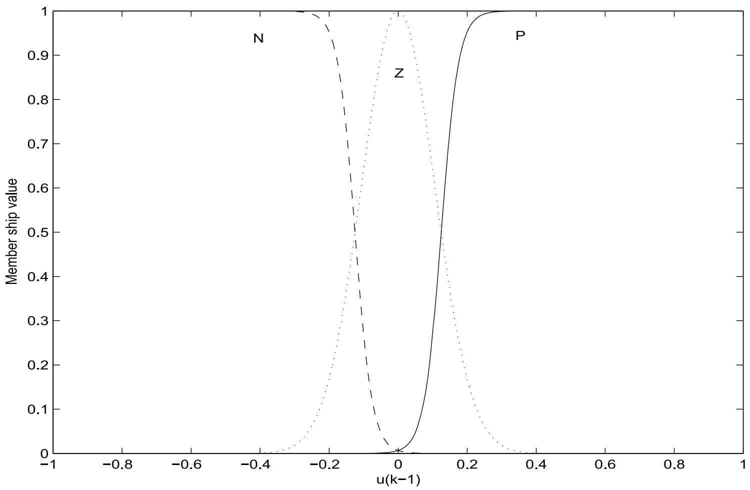


Fig. 2. Membership functions of $u(k-1)$

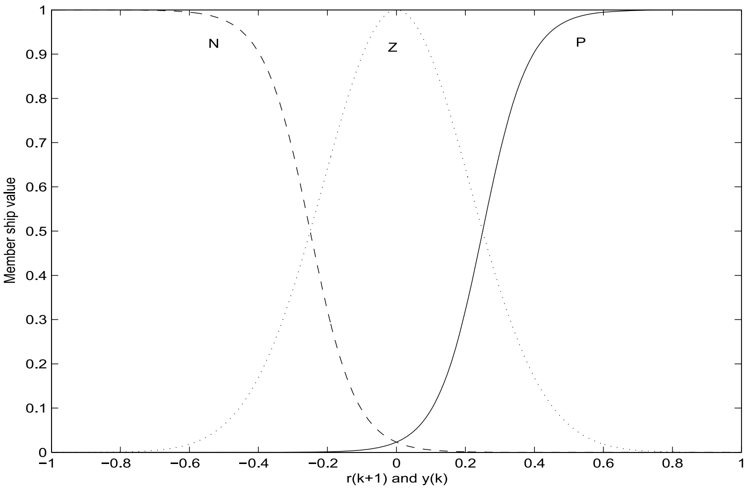


Fig. 3. Membership functions of $r(k+1)$ and $y(k)$

The initial setting $\beta_{i,j}(1)$ for $i = 1,2$ and $j = 1,2,\cdots,9$ can be given as

$\beta_{i,1}(1)$=-1	$\beta_{i,2}(1)$=-0.75	$\beta_{i,3}(1)$=-0.5,
$\beta_{i,4}(1)$=-0.25	$\beta_{i,5}(1)$=0	$\beta_{i,6}(1)$=0.25,
$\beta_{i,7}(1)$=0.5	$\beta_{i,8}(1)$=0.75	$\beta_{i,9}(1)$=1.

In Fig. 4, the tracking performance is quite satisfied with out the off-line learning. The control effort is illustrated in Fig. 5. The convergence of $\beta_i(k)$ is shown by $||\beta_i(k)||$ in Fig. 6 for both MIFRENs.

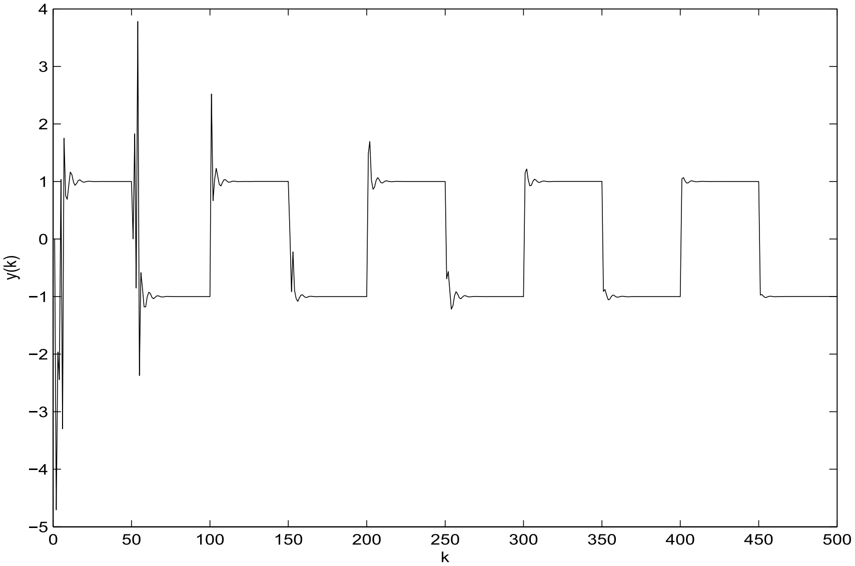


Fig. 4. Tracking performance $y(k)$ for nominal plant.

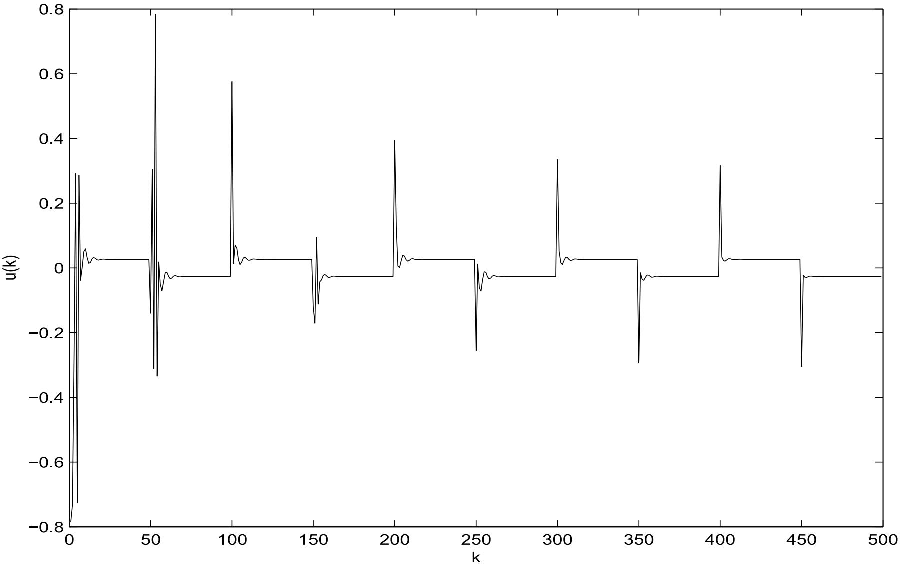


Fig. 5. Control effort $u(k)$ for nominal plant.

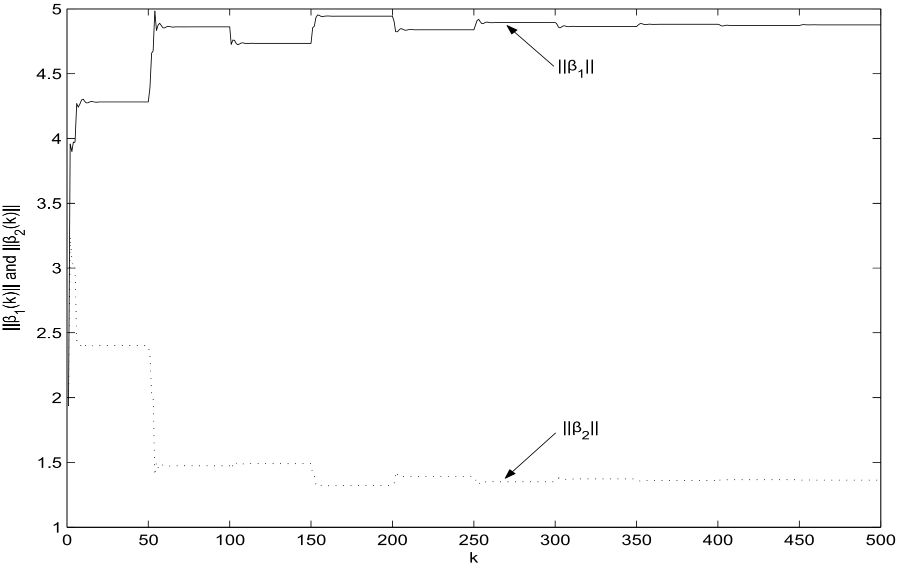


Fig. 6. $||\beta_i(k)||$ for nominal plant.

4.2 Robust control

In the robust system case, the uncertainty terms $\Delta f_1(k)$ and $\Delta f_2(k)$ are included in the system (49) as

$$y(k+1) = \sin(y(k)) + \Delta f_1(k) + \cos(y(k)u(k))u(k) + 5u(k) + \Delta f_2(k)u(k), \tag{50}$$

when

$$\Delta f_1(k) = \begin{cases} 1 & \text{if } 0 < k < 125 \\ 0.75 & \text{if } 125 \le k < 325 \\ -1.25 & \text{if } 325 \le k < 425 \\ 1.25 & \text{if } 425 \le k < 500, \end{cases} \tag{51}$$

and

$$\Delta f_2(k) = \begin{cases} -0.5 & \text{if } 0 < k < 125 \\ 1 & \text{if } 125 \le k < 225 \\ -0.75 & \text{if } 225 \le k < 425 \\ -0.5 & \text{if } 425 \le k < 500. \end{cases} \tag{52}$$

We use the initial setting IF-THEN rules, membership functions, ε_m, η, $\bar{y_u}$ and parameter vectors β_i, as the same as the previous one. With out any off-line learning for MIFRENs, the tracking performance is represented in Fig. 8. The control effort $u(k)$ is shown in Fig. 9. The time variation of $||\beta_i(k)||$ can be illustrated in Fig. 10. These uncertainty terms $\Delta f_1(k)$ and $\Delta f_2(k)$ are varied with time but the tuning vectors are all bounded.

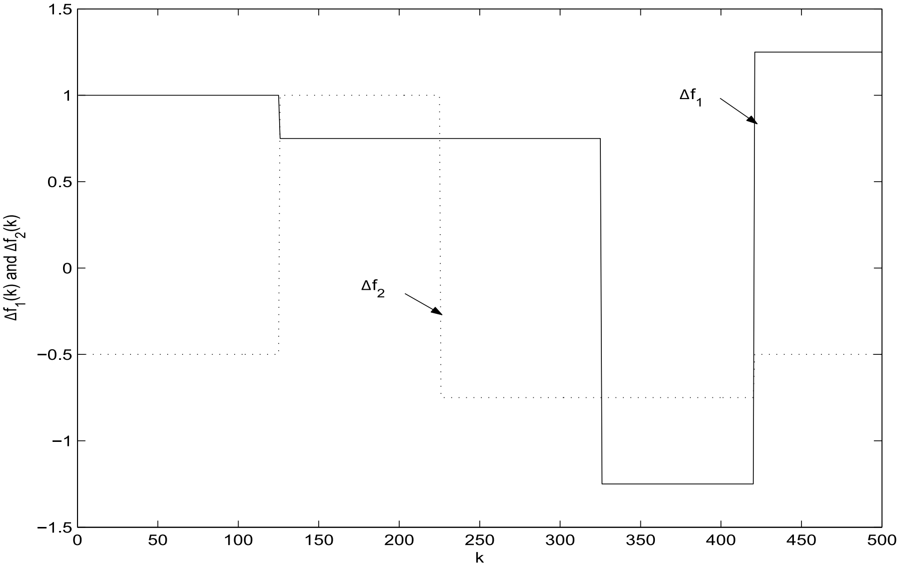


Fig. 7. Illustration of uncertainty $\Delta f_1(k)$ and $\Delta f_2(k)$.

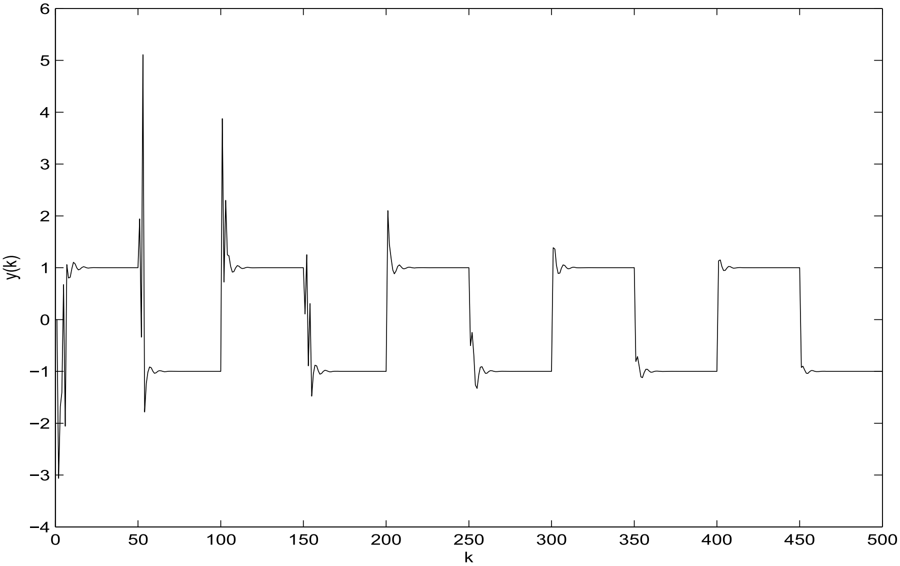


Fig. 8. Tracking performance $y(k)$ for robust system.

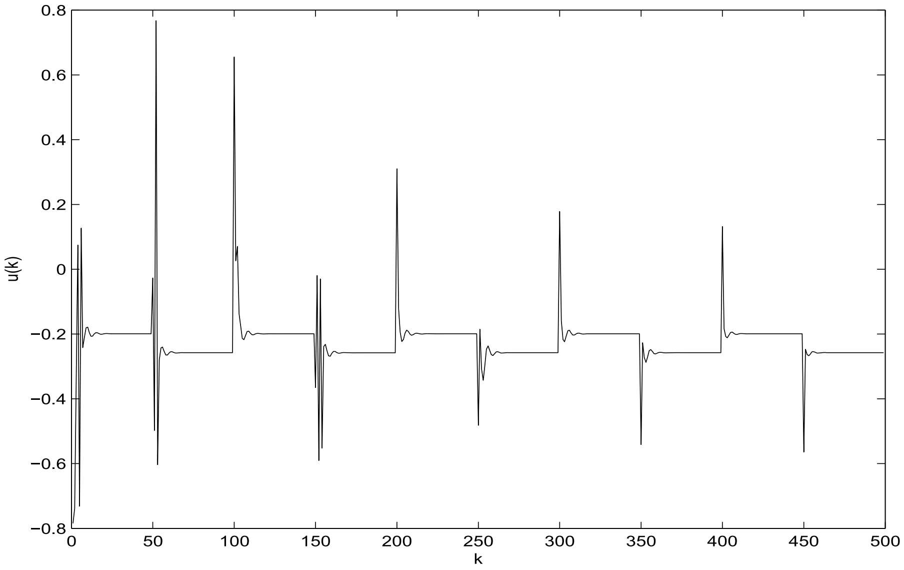


Fig. 9. Control effort $u(k)$ for robust system.

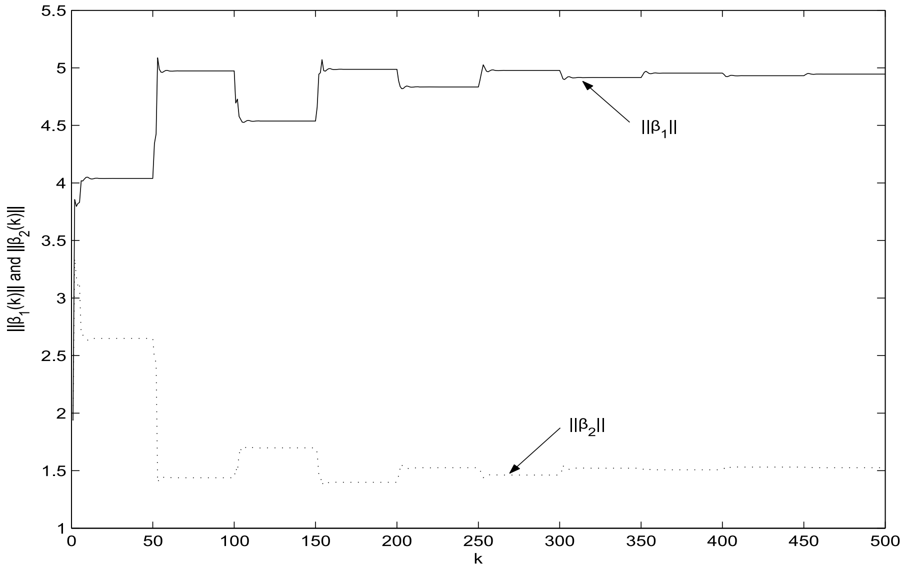


Fig. 10. $||\beta_i(k)||$ for robust system.

5. Experimental setup example

In this section, the performance of our proposed controller is demonstrated by an experimental setup with FESTO mobile robot system called Robotino® . Our task is to design the controller for moving this Robotino® to reach the desired position in (x,y) coordinate as $x_d(i,k)$ and $y_d(i,k)$, respectively. During the movement, the desired angular of Robotino® denoted as $\phi_d(i,k)$ should be maintained as 0° for all i^{th} desired position and time index k. The system configuration can be illustrated in Fig. 12 by the block diagram.

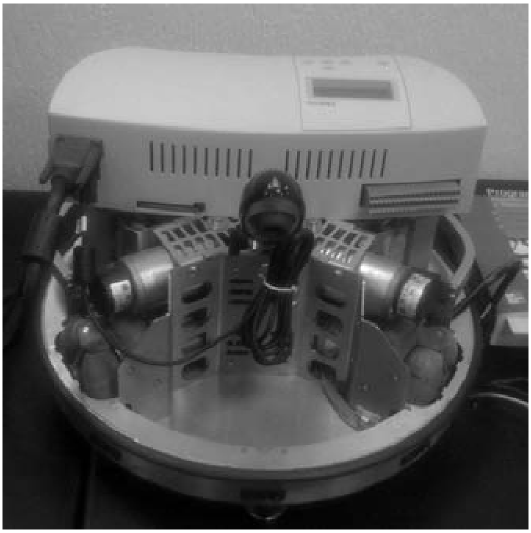


Fig. 11. Robotino.

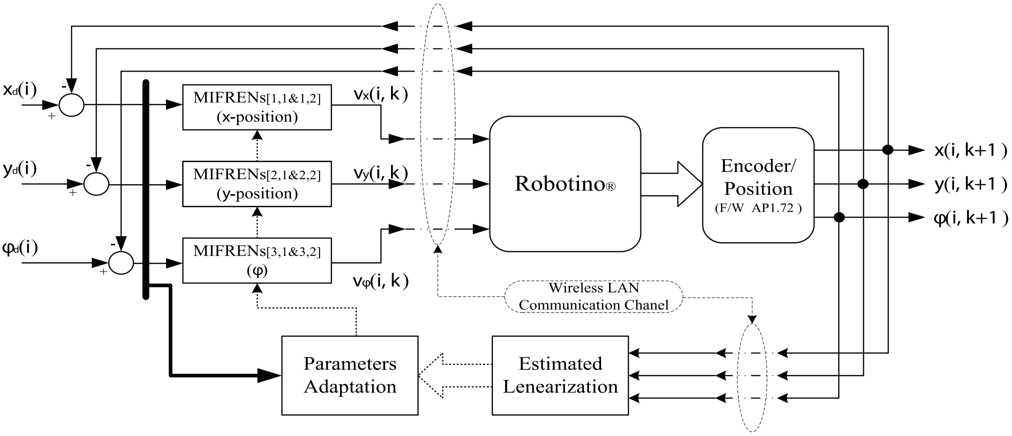


Fig. 12. Block diagram for experimental setup.

The commercial Robotino® needs velocity to control its movement such as velocity in $x-$axis $v_x(i,k)$ for $x-$direction, $v_y(i,k)$ for $y-$direction and $v_{\phi(i,k)}$ for the rotation. In this work, we consider these signals as the control efforts which can be generated by the pair of MIFRENs. The experiment has been demonstrated by 4 desired points and 4 routes as the following: route 1 [(0.0, 0.5)→(0.5, 0)], route 2 [(0.5, 0)→(0.0, 0.0)], route 3 [(0.0, 0.0)→(0.5, 0.5)] and route 4 [(0.5, 0.5)→(0.0, 0.5)]. In Fig. 13, the movement of Robotino® is illustrated in $x-y$ coordinate with errors in x and y axis as e_x and e_y shown in Fig. 14. Because of the fixed angular $\phi_d = 0$, we need to consider only two control efforts v_x and v_y as presented in Fig. 15. At the beginning, on route 1 and 2, the movement of robot is not strange line because MIFRENs need to tune the parameters inside. After that the better results can be obtained in route 3 and 4. In case of losing the wireless signal, we still have the satisfied result as shown in Fig. 16.

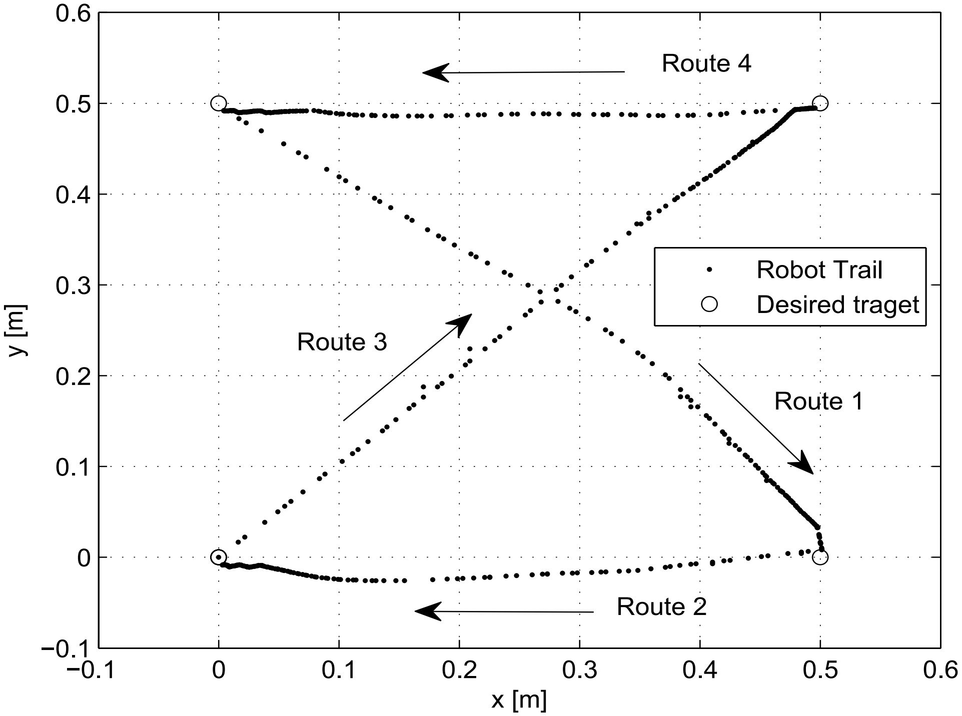


Fig. 13. Experimental result: position $x-y$.

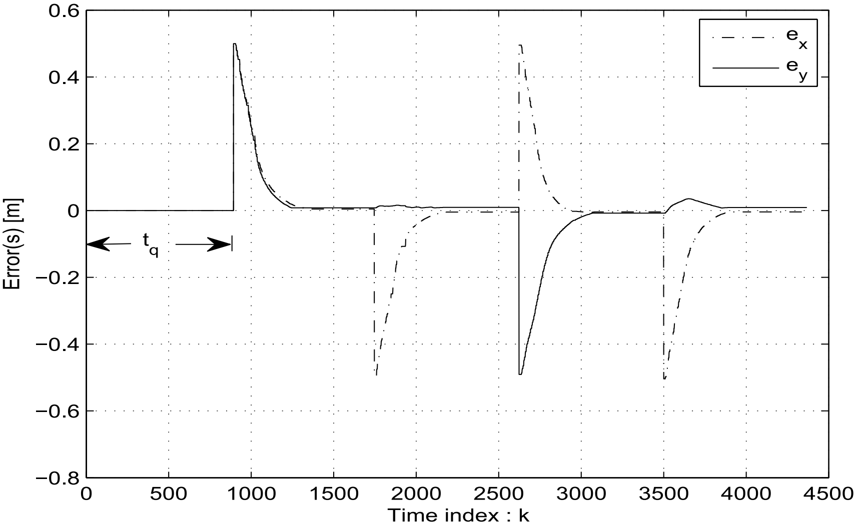


Fig. 14. Experimental result: position errors e_x and e_y.

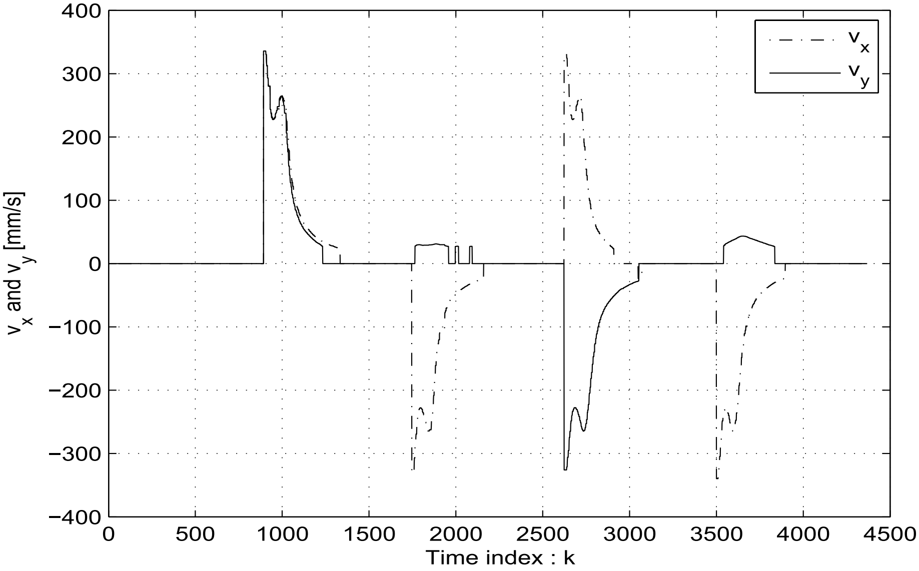


Fig. 15. Experimental result: velocity v_x and v_y.

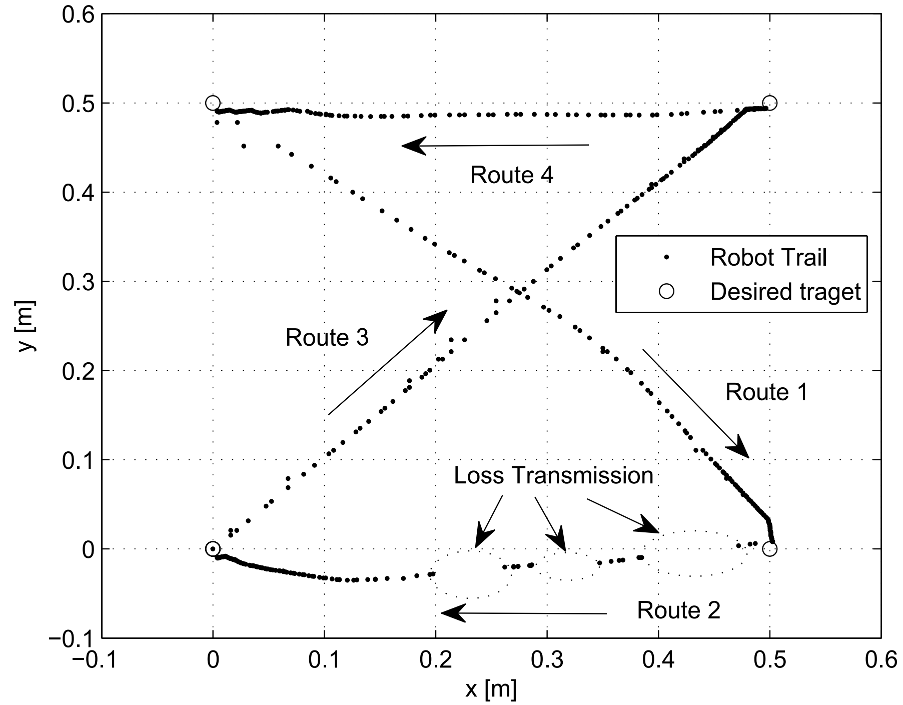


Fig. 16. Experimental result: position $x - y$ (in case of loss signal transmission).

6. Conclusion

In this chapter, an adaptive control for a class of nonlinear discrete-time systems based on multi-input fuzzy rules emulated networks (MIFREN) is introduced by the approximation with Taylor and mean value theorem. With out the need of mathematical system model, the approximation can be existed directly to construct the control law. Two MIFRENs are implemented to estimate these unknown functions obtained by the nonaffine linearization. With the main theorem, the learning algorithm for parameters inside MIFRENs guarantees the convergence of these parameters and the satisfied tracking performance. The computer simulation system demonstrates the accuracy of our mathematic proof. We already consider both operating cases for the nominal plant and the plant with some uncertainties. The bounded parameters $||\beta_1||$ and $||\beta_2||$ and the satisfied tracking performance can be presented for the both cases with the same initial setting. The experimental setup the commercial mobile robot system called Robotino® is given to demonstrate the controller performance.

7. References

[1] C.T. Lin, *Neural fuzzy systems*, Prentice-Hall, 1996.

[2] C. Treesatayapun and S. Uatrongjit, "Adaptive controller with Fuzzy rules emulated structure and its applications'," *Engineering Applications of Artificial Intelligence,* Elsevier, vol. 18, pp. 603-615, 2005

[3] C. Treesatayapun "Nonlinear Systems Identification Using Multi Input Fuzzy Rules Emulated Network," *ITC-CSCC2006 International Technical Conference on Circuits/Systems, Computers and Communications,* Chiangmai, Thailand, 10-13 July 2006

[4] E. Armanda-Bricaire, U. Kotta, and C. H. Moog, "Linearization of discrete-time systems," *SIAM J. Contr. Optim.*, vol. 34, pp. 1999-2023, 1996.

[5] G. L. Plett, "Adaptive Inverse control of linear and nonlinear system using dynamic neural networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 360-376, Mar. 2003.

[6] H. X. Li and H. Deng, "An approximation internal model-based neural control for unknown nonlinear discrete processes," *IEEE Trans. Neural Networks*, vol. 17, no. 3, pp. 659-670, May. 2006.

[7] H. Deng and H. X. Li, "A novel neural network approximate inverse control for unknown nonlinear discrete dynamical systems," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 35, no. 1, pp. 115-123, Feb. 2005.

[8] H. Deng and H. X. Li, "On the new method for the control of discrete nonlinear dynamic systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 526-529, Mar. 2006.

[9] J. B. D. Cabrera and K. S. Narendra, "Issues in the application of neural networks for tracking based on inverse control," *IEEE Trans. Automat. Contr.*, vol. 44, pp. 2007-2027, Nov. 1999.

[10] J. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[11] L. Chen and K. Narendra, "Identification and Control of a Nonlinear Discrete-Time System Based on its Linearization: A Unified Framework," *IEEE Trans. Neural Networks*, vol. 15, no. 3, pp. 663-673, May. 2004.

[12] M. Chen and D.A. Linkens,"A hybrid neuro-fuzzy PID controllers," *Fuzzy Sets and System*, pp. 27-36, 99, 1998.

[13] P. He and S. Jagannathan, "Reinforcement Learning Neural-Network-Based Controller for Nonlinear Discrete-Time Systems With Input Constraints," *IEEE Trans. Syst., Man., Cybern.*, vol. 37, no. 2, pp. 425-436 Apr. 2007.

[14] Q. Gan and C. J. Harris, "Fuzzy local linearization and local basis function expansion in nonlinear system modeling," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 559-565, Aug. 1999.

[15] Q. M. Zhu and L. Guo, "Stable adaptive neurocontrol for nonlinear discrete-time systems," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 653-662, May 2004.

[16] S. S. Ge, J. Zhang, and T. H. Lee, "Adaptive MNN control for a class of non-affine NARMAX systems with disturbances," *Syst. Control Lett.*, vol. 53, pp. 1-12, 2004.

[17] W. Rudin, *Principles of Mathematical Analysis*, New York: McGraw Hill, 1976.

[18] Z. Q.Wu and C. J. Harris, "A neurofuzzy network structure for modeling and state estimation of unknown nonlinear systems," *Int. J. Syst. Sci.*, vol. 28, pp. 335-345, 1997.

17

Knowledge Structures for Visualising Advanced Research and Trends

Maria R. Lee[1] and Tsung Teng Chen[2]
[1] *Shih Chien University*
[2]*National Taipei University*
Taiwan

Abstract

Due to the enormous amount of research publications are available, perceiving the growth of scientific knowledge even in one's own specialise field is a challenge task. It would be helpful if we could provide scientists with knowledge visualisation tools to discover the existence of a scientific paradigm and movements of such paradigms. This chapter introduces the state of the art of visualising knowledge structures. The aim of visualising knowledge structures is to capture intellectual structures of a particular knowledge domain. Approaches to the visualisation of knowledge structures with emphasis on the role of citation-based methods are described. The principal components of factor analysis and Pathfinder network are utilized to reveal new and signification developments of intellectual structure in ubiquitous computing research area. Literature published in the online citation data bases CiteSeer and Web of Science (WoS) are exploited to drive the main research themes and their inter-relationships in ubiquitous computing. The benefit of the results obtained could be for someone new to a specific domain in research study, ubiquitous computing in this case. The outcome uncovers popular topics and important research in the particular domain. Potential developments can be re-used and utilized in other disciplines and share across different research domains.

1. Introduction

Computing technology is a paradigm shift where technology becomes virtually invisible in our lives and is a rapidly advancing and expanding research and development field in this decade. Due to the enormous amount of available scientific research publications, keeping up the growth of scientific knowledge even in one's own specialise field is a challenge task. It would be helpful if we could provide scientists with knowledge visualisation tools to detect the existence of a scientific paradigm and movements of such paradigms. The main scientific research themes are also very difficult to analyze and grasp by using the traditional methodologies. For example, visualising intrinsic structures among documents in scientific literatures could only capture some aspects of scientific knowledge.

This chapter introduces the state of the art of visualising knowledge structures. The aim of visualising knowledge structures is to capture and reveal insightful patterns of intellectual structures shared by scientists in a subject field. This chapter describes approaches to the visualisation of knowledge structure with emphasis on the role of citation-based methods. Instead of depending upon occurrence patterns of content-bearing words, we aim to capture the intellectual structures of a particular knowledge domain.

We focus on the study of ubiquitous computing, also called pervasive computing. Numerous journals and conferences are now dedicated to the study of ubiquitous computing and related topics. Ubiquitous computing was first described by Weiser (1991). Since then, a rich amount of related literatures are published. It would be useful if the content of those publications could be summarised and presented in an easy way to capture structures and facilitate the understanding of the research themes and trend in ubiquitous computing.

The goal of the chapter is to show the scope and main themes of ubiquitous computing research. We begin by examining the survey studies of visualising knowledge structures. Next, the data collection method and intellectual structure techniques, factor analysis and Pathfinder Network, are introduced. The results of the analysis are presented and discussed.

2. Visualising Knowledge Structures

Information visualisation techniques have become a rapid growth research area since the last decade (Card et al 1999; Chen 1999). In information retrieval, the vector-space model (Salton 1991) is an originally and popularly exploited framework for indexing documents based on term frequencies. A focal part of modern statistical probability modelling is Bayesian theorem (Neal 1996), which focuses on the probabilistic relationship between multiple variables and determining the impact of one variable on another. Shannon's information theory (Shannon 1948) describes information could be treated as a quantifiable value in communications. Self-organised feature maps are essentially classification processes through a neural network. (Lin, Soergel and Marchionini 1991) is the first to use self-organised maps to visual information retrieval. WEBSOM organizes textual documents for exploration and search based on self-organised map (Lagus et al 1996).

Visualising knowledge structure is an art of making maps, which shares some intrinsic characteristics with cartography (Chen 2002). Number of useful knowledge visualisation techniques has been applied to detect and extract significant elements from unstructured text. The basis for the visualisation of knowledge structures is formed by the interrelationships between these elements. Citation indexing has been widely applied since 1950s. One of the fundamental objectives of science mapping is to identify the trend associated with a field of study. The map created through citation analysis provides a series of historical data, which cover the literature year by year (Garfield 1975). These maps show intrinsic semantic connections among disciplines of domains. The author co-citation analysis (ACA) was introduced to discover how scientists in a particular subject field are intellectually interrelated as perceived by authors in their scientific publications (White and Griffith 1981). An intellectual structure of prominent authors in the field provides a respectable source for knowledge visualisation.

Knowledge Domain Visualisation (KDV) depicts the structure and evolution of scientific fields (Borner, Chen and Boyak 2002). Some recent works in knowledge discovery and data mining systems compose analysis of engineering domain (Mothe and Dousset 2004; Mothe et al 2006).

2.1 Factor Analysis

Factor analysis is one of the commonly used methods in author co-citation analysis. It has been used to identify the intrinsic dimensionality of given co-citation data in a subject domain. (White and McCain 1995) demonstrates the author co-citation analysis of the information science field that some authors indeed belong to several specialties simultaneously. However, if datasets is big, then the size of the corresponding author co-citation matrix could be large and the analysis becomes computationally complicate and expensive.

White and McCain introduces the raw co-citation should be transformed into Pearson's correlation coefficients using the factor analysis (White and McCain 1995). The correlation coefficients measure the nearness between authors' co-citation profiles. Principal component analysis (PCA) is a suggested alternative to extract factors. The default criterion, Eigen values greater than 1, is normally chosen to decide the number of factors extracted. Missing data should also be replaced by mean co-citation counts for corresponding authors. Pearson's correlation coefficient can be used as a measure of similarity between pairs of authors.

2.2 Pathfinder Network Scaling

Pathfinder network scaling is originally developed by cognitive psychologists for structuring modelling (Schcaneveldt, Durso and Dearholt 1989). Pathfinder network scaling relies on the triangle inequality condition to select the most salient relations from proximity data. The Pathfinder network (PFNET), the results of Pathfinder network scaling, consists of all the vertices from the original graph. The number of edges in a Pathfinder network is driven by the basic structure of semantics. The topology of a PFNET is decided by two parameters *q* and *r*. The corresponding network is denoted as PFNET (*q*, *r*). The *q*-parameter controls the scope that the triangular inequality condition should be set. The *r*-parameter is used to computing the distance of a path. The weight of a path with *k* links is determined by weights *w1, w2, …, wk* of each individual link as follows.

$$W(P) = \left[\sum_{i=1}^{k} w_i^r \right]^{\frac{1}{r}}$$

3. Intellectual Structure of Ubiquitous Computing

Numerous amounts of scientific papers publish every year and the accumulated literatures over the years are voluminous. We utilized the methods that have been developed in visualizing information structure to comprehend the entire body of scientific knowledge. The aim is to discover the development in ubiquitous computing.

3.1 Proposed Process

Figure 1 shows a proposed process to construct a full citation graph from the data drawn from online citation databases, WOS and CiteSeer. The proposed procedure leverages the citation index by using key phrases "ubiquitous computing" to query the index and retrieve all matching documents from the database. The documents retrieved by the query are then used as the initial seed set to retrieve papers that are citing or cited by literatures in the initial seed set (Lee and Chen 2007; Chen and Lee 2006; Pozi 2002). The co-citation matrix is derived from the co-citation relationships between papers. A co-citation relationship existed between two papers when a third paper cites them both, i.e., both papers are listed in the reference portion of the third paper. The full citation graph is built by linking all articles retrieved, which includes more documents than the other schemes reviewed earlier.

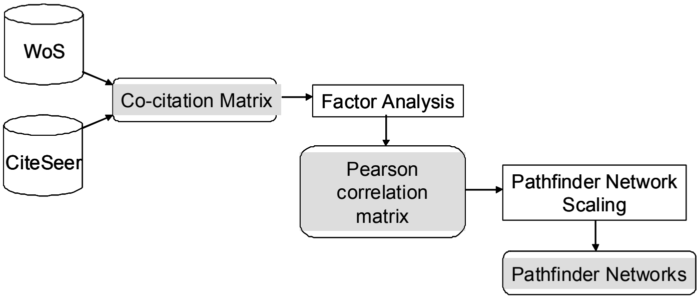


Fig. 1. The Proposed Process

The main usage of the factor analysis is to reduce the number of variables and to detect structure in the relationships between variables. Factor analysis combines correlated variables (papers) into one component (research theme). The co-citation matrix is the input of factor analysis. The co-citation graph is represented by a matrix to compute the correlation matrix of Pearson's correlation coefficients. The Pearson correlation matrix, which is resulted from the factor analysis, is the input of the Pathfinder network scaling. All nodes in a graph are connected by weighted links. The weights are represented by the value of correlation coefficients for each pair of documents.

The citation data are driven from two online citation databases, Citeseer and IS Web of Science (WoS). CiteSeer is an open access free database, which is a scientific literature digital library. Citseer's search engine focuses primarily on the literature in computer and information science. ISI Web of Knowledge database is created by Thomson Reuters in 1997 and integrated access to high quality, multidisciplinary research literature. Web of Science (WoS) is part of ISI Web of Knowledge. WoS covers SCI, SSCI and A&HCI citation databases.

Citation data are collected by querying both databases with the key phrases 〝ubiquitous computing〞 and retrieving the initial key papers' information. The key papers are then used as the initial seed set to retrieve papers that are citing or are cited by literatures in the initial seed set (Chen and Xie 2005). A full citation graph is generated by linking all articles retrieved. The depth of the expanded search is restricted to three layers to maintain the most relevant literatures.

3.2 Main Components derived from Factor Analysis

Factor analysis is applied as a data reduction and structure detection method. The co-citation matrixes generated from CiteSeet and WoS are derived from the citation graphs and fed to factor analysis. The unit of analysis is based on documents rather than author due to a researcher's specialty may evolve over time (Chen and Lee 2006; Lee and Chen 2008).

49 components with Eigenvalue over one were identified from the CiteSeer citation data. These factors collectively explained approximately 84.2% total variances. Papers with a loading over 0.6 to a component are collected and studies to determine the content of the component. A proper descriptive name for each component is decided, which represent the research trends in the ubiquitous computing field. Based on CiteSeer citation data, top 10 components and the variances of the components explained are listed in Table 1. Component 9 does not include any papers with loading larger than 0.6 and therefore, is not listed in the table. The content of these nine factors are described in the context of ubiquitous computing.

Factor	Component Name	Variance Explained	Description
1	Routing protocols for mobile and ad hoc networks	7.206	An ad hoc network is a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration.
2	Location and data management in mobile wireless environment	6.733	Location and data management in a mobile wireless environment is different from the transitional fixed wire environment. Location and data management is required when all the communications over a mobile wireless environment with a sufficient and steady bandwidth.
3	Location and context aware computing	5.702	Location and context aware computing will free the user from the traditional constraints of the desktop. Context refers to the physical and social situation in which computational devices are embedded. Contextual information can be used to provide services that are appropriate to the situational events.
4	Broadcast based data management for asymmetric communication environments	5.023	Broadcasted data has been proposed as a means to efficiently deliver data to clients in asymmetric environments, where the available bandwidth from the server to the clients exceeds the bandwidth in the opposite direction. In the presence of such asymmetry, applications must rely on the broadcast data channel to receive the up-to-date information.
5	Integrating with computer	4.310	Interacting with computer augmented artifacts and environments may greatly

	augmented artifacts and environment		enhance a user's experience. Computer augmented physical objectives or devices may facilitate more effectively computational mediation.
6	Transmission control protocol (TCP) over mobile internetworks	4.120	The study of TCP over mobile internetworks addresses the performance issue of reliable data communication in mobile computing environments. Two changed assumptions need to be addressed in the mobile computing: (1) the end points of the communication link are fixed and (2) the underlying network has high and reliable bandwidth with low latency.
7	Application design for mobile computing	3.524	This factor addresses the disparity of mobile devices in resources, network characteristics, display size, and method of input from application level. Application design strategies may reduce the demands placed on the wireless network.
8	Disconnected operations	3.023	Disconnected operation is a mode of operation that enables a client to continue accessing critical data during temporary failures of a shared data repository. The temporary failures may due to networks or data sources breakdown. The core idea behind this work is utilizing tradition performance improving data, such as caching data, to improve availability.
10	Data access in a scalable distributed environment (e.g. a P2P network)	2.618	Data access in a scalable distributed environment (e.g. a P2P network) is analogous to the data access of an ad hoc mobile network. The lookup mechanism used to locate a desire file is analogous to the ad hoc routing operation that locates nearby mobile nodes to forward data jackets.

Table 1. Top 10 Factors of ubiquitous computing drew from CiteSeer

From the WoS dataset, 30 components with Eigenvalue over one collectively explained approximately 86.4% total variances. These components are selected as the representative major themes of ubiquitous computing. Papers with a loading over 0.5 to a component are collected and studied to determine the content of the component. Based on WOS citation data, top 10 components with descriptions and the variances of the components explained are listed in Table 2.

Factor	Component Description	Variance Explained	Description
1	Foundational studies of ubiquitous computing	17.123	Foundational studies of ubiquitous computing provide a generic platform for location and spatial-aware systems. The platform supports a unified spatial-aware infrastructure based on digital models of the physical world. A universal spatial and context-aware infrastructure is essential to overcome the sheer diversity of exploitable contexts and the myriad of sensing technologies.
2	Power aware routing protocol for wireless sensor network	8.329	The availability of small, lightweight low-cost network is crucial to the success of ubiquitous computing. The lightweight network uses energy sparingly to prolong the operational span of the ubiquitous network. The power saving algorithms and protocols are the focus of much ubiquitous computing related research.
3	Medical informatics, application of ubiquitous computing in health care	6.917	The ubiquitous availability of clinical information is major trend in medical informatics research. The application of new information and communication technologies will offer new opportunities and increase the potential of medical informatics methods and tools. The mobility of hospital environment, such as staff, patients, documents and equipments, makes hospitals' ideal applications for pervasive or ubiquitous computing technology.
4	Context-aware workflow language based on Web services	5.105	Research in this factor seems to explore the common feature of Web services and ubiquitous computing. According to W3C, the web services are defined as "a software system designed to support an interoperable machine to machine interaction over a network". The standardization of ubiquitously available services and interoperability between services (factor 1) becomes the natural bond between web services and ubiquitous computing.
5	Context-aware computing	4.878	Papers in factor 5 try to clarify and define the scope and content of context aware computing. Context awareness is the key to dispersing and enmeshing ubiquitous

			computation in our lives. Contextual information is acquired and utilized by devices to provide services that are appropriate to the situational events.
6	Ambient intelligent systems	4.854	Papers in factor 6 extended the context-aware computing to ambient intelligent systems. How “context” can be effectively utilized by a context-aware system, which in turn exhibit ambient intelligent is the focal issue of research in this factor.
7	Open services gateway initiative (OSGi)	4.766	OSGi is technology standard that can coordinate diverse device technologies and enable compound services across different networking technologies. OSGi can be viewed as an initial effort of commercial realisation of the universal spatial and context-aware infrastructure envisioned by the academy.
8	Ubiquitous applications in education	4.350	Articles in factor 8 explore how learning could be augmented by ubiquitous computing devices in an educational setting. The functionalities of traditional classroom equipment and instruments such as whiteboard, notebook PC, PDAs and other learning aids could be augmented by embedding ubiquitous computing power to enhance the learning environment and enriching the learning experiences.
9	Information technology as the competitive advantages of business	3.573	Papers in factor 9 deal with IT and sustained business competitive advantage. IT has been recognized as one of the key competitive advantage for modern businesses. How an organisation builds up its IT capability, streamline its business process, and reconciles its organisational structure are important issues.
10	Tangible and graspable user interfaces	2.903	Factor 10 includes papers on the study of tangible and graspable user interfaces. A tangible user interface lets users using graspable physical objects to emulate the functions of traditional icon-based computer graphical user interfaces. Instead of clicking or dragging a mouse, the tangible user interface tries to carry out the human computer interaction through manipulation of physical objects.

Table 2. Top 10 Factors of ubiquitous computing drew from Web of Science (WOS)

3.3 Pathfinder Network

The Pearson's correlation coefficients between items (papers) are calculated and used as the basis for PFNET scaling. The value of Pearson correlation coefficient falls between the range -1 and 1. Highly correlated items are placed closely together spatially. The nodes located close to the centre of a PFNET graph represents papers contributed to a fundamental concept, which are frequently referred by other peripheral literature that are positioned in outer branches. The distance between items is inversely propositional to the correlation coefficient, which maps less correlated items apart and highly correlated items spatially adjacent.

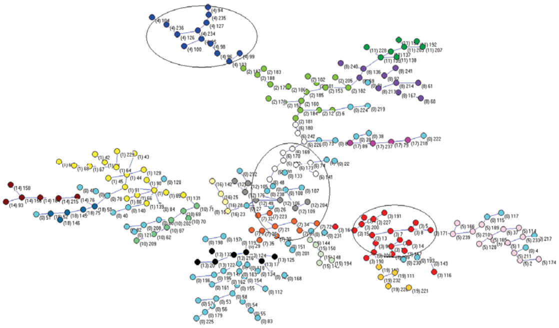


Fig. 2. PFNET Scaling of ubiquitous computing drew from CiteSeer.

Figure 2 represent PFNET scaling of ubiquitous computing from CiteSeer. Articles under the same factor are painted with the same colour. The number in the parenthesis is the factor number which an article belongs to. Cyan nodes with (0) represent articles that are not assigned to any factor. The top ranked components cluster numbered 6, 7, and 12 locate closely to the centre of the PFNET graph (surrounded by a big circle in the centre of the graph), which suggests papers in these components play fundamental roles in ubiquitous computing. A foundational but low-ranked factor may be interpreted as an underdeveloped but important theme in the research field. Component 4 (top left circle, node colour in blue), although ranked high in the amount of variance explained, plays peripheral roles in ubiquitous computing related research. A high-ranked peripheral component is generally an important study, but plays only a supplement role in ubiquitous computing study. Component 3 (lower right circle, node colour in red) plays an interesting role, which indicates that location and context aware computing related researches are important to ubiquitous computing and are important topics in general.

Figure 3 represent PFNET scaling of ubiquitous computing from Web of Science (WoS). Articles under the same factor are painted with the same colour. The number in the parenthesis is the factor number which an article belongs to. Top ranked component cluster number 1, 5, and 6 locate closely to the centre of the PFNET graph (surrounded by a big circle in the centre of the graph), which suggests papers in these components play foundational roles in ubiquitous computing research. Component cluster number 2 (lower-right circle, node colour in green) and 4 (upper right circle, node colour in blue), although ranked high in the amount of variance explained, they play peripheral roles in ubiquitous computing related research. Component cluster number 3 (node colour in red) plays an interesting role, which indicates that medical informatics research is important to ubiquitous computing as well as an important topic in general.

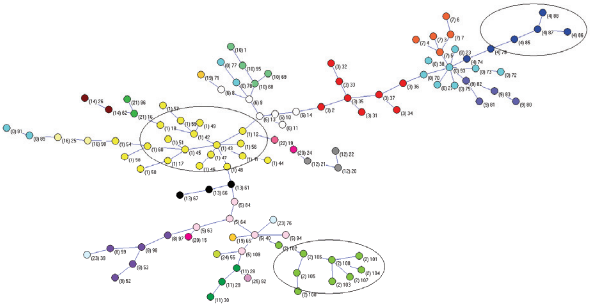


Fig. 3. PFNET Scaling of ubiquitous computing drew from Web of Science (WoS).

4. Discussion

Abowd and Munatt (Abowd and Munatt 2000) investigate the main research focuses in ubiquitous computing publications in 2000, which include natural interface, context-aware applications and automated capture and access. However, they mainly based on the bounded expertise of the author(s) and a rather limited set of references. We propose a visualising knowledge structure method in analyzing large collections of literatures, which reveals the major research themes and their inter-relationships in ubiquitous computing. We utilize the intellectual structure construction and knowledge domain visualisation techniques developed by the information scientists to ease the task of understanding the main research themes in ubiquitous computing.

Based on the citation papers derived from factor analysis and PFNET in ISI Web of Science (WoS) in 2008, foundational ubiquitous computing studies, context-aware computing, and ambient intelligent systems provide fundamental and important knowledge base to ubiquitous computing studies. Power aware routing protocol and context-aware workflow language are relevant and important studies in general, but only play a supporting or supplemental role in ubiquitous computing research.

In contrast, based on the citation data drew from CiteSeer in 2008, the study of application design for mobile computing, TCP over mobile internetworks and network support for real-time applications provide a fundamental and important technical knowledge base to ubiquitous computing studies. Broadcast based data management for asymmetric communication environments and interacting with computer augmented artefacts and environment are relevant and important studies in general, but only play a supporting or supplemental role in ubiquitous computing studies.

The difference between main themes drew from CiteSeer and WoS CiteSeer is due to that CiteSeer citation index is primarily a computer, information science and engineering citation database, whereas WoS is a comprehensive index. The intellectual structure derived from a predominantly science and engineering oriented index is biased toward the technical aspect of ubiquitous computing. In contrast, WoS is a comprehensive citation database. WoS reveals the application and business themes as well as the technical one.

5. Conclusion

Providing scientists with knowledge visualization tools to reveal the scientific paradigm and movements of such a paradigm is a challenge task. We have introduced the method of visualizing knowledge structures with emphasis on the role of citation-based methods. Factor analysis and Pathfinder Network are used to discover new and significant developments of intellectual structure in the ubiquitous computing research field. Literature published in the online citation databases CiteSeer and Web of Science (WoS) in 2008 were explored to drive the research themes.

We tried to provide a broader view of ubiquitous computing study by applying intellectual structure methods developed by information scientists. The main themes can be uncovered with respect to fundamental and important knowledge as well as supporting or supplemental knowledge in ubiquitous computing domain. The results obtained show that the study of application design for mobile computing, TCP over mobile internetworks, network support for real-time applications, foundational ubiquitous computing studies, context-aware computing, and ambient intelligent systems are fundamental topics in ubiquitous computing. Broadcast based data management for asymmetric communication environments, interacting with computer-augmented artefacts and environment, Power aware routing protocol and context-aware workflow language are relevant and important studies in general, but only play a supporting or supplemental role in ubiquitous computing research.

The benefit of the results obtained could be for someone new to a specific domain in research study. The proposed method may be re-used in other disciplines and share across different research domains. One of the future directions is to apply this proposed method to leverage the research theme networks, which is intellectually interrelated the relationships among publications, citations, research projects, and even patents. We also plan to explore further the interdisciplinary researches in future studies.

6. References

Abowd, G. and Munatt , E. (2000) *Charting past, present and future research in ubiquitous computing*, ACM transactions on Computer-Human Interaction, vol 17, pp29-58.

Borner, K., Chen, C. And Boyack, K. (2002) *Visualising Knowledge Domains*, in Annual Review of Information Science and Technology, vol 37, pp179-255.

Card, S., Mackinlay, J. and Shneiderman, B. (1999) *Readings in Information Visualisation : Using Vision to Think*, Morgan Kaufmann.

Chen, G. And Kotz, D. (2000) *A Survey of Context-Aware Mobile Computing Research,* Technical Report TR2000-381, Dept. Of Computer Science, Dartmouth College.

Chen, C. (2002) *Visualisation of Knowledge Structures*, in the Handbook of Software Engineering and Knowledge Engineering, Vol2, pp700-744.

Chen, C. (1999) *Information Visualisation and Virtual Environments*, Springer-Verl ag, Londan.

Chen, T. and Lee, M. Revealing Themes and Trends in the Knowledge Domain's Intellectual Structure, PKSW 2006, LNCS (LNAI), vol 4303, pp99-107.

Chen, T. and Xie, L. *Identifying Critical Focuses in Research Domains*, Proceedings of the Information Visualization, Ninth International Conference on (IV'05), p135-142.

Garfield, E. (1975) *Citation indexes for science : a new dimension in documentation through association of ideas*, Science, 122 (108-111).

Lagus, K. Honkela, T. Kaski, S. and Kohonen, T. (1996), *WEBSOM - A Status Report*, Proceedings of STeP'96. Jarmo Alander, Publications of the Finnish Artificial Intelligence Society, pp. 73-78.

Lee, M. and Chen, T. (2008) *From Knowledge Visualization Techniques to Trends in Ubiquitous Multimedia Computing*, 2008 International Symposium on Ubiquitous Multimedia Computing, 73-78.

Lee, M. and Chen, T. (2007) *Visualizing Trends in Knowledge Management,* Knowledge Science, Engineering and Management, LNAI 4798, 262-271.

Lin, X., Soergel, D. and Marchionini, G. *A self-organising semantic map for information retrieval*, in SIGIR'91, ACM press, 262-269.

Mothe, J., Chrisment, C. Dkaki, T. Dousset, B., and Karouach, S. (2006) *Combining mining and visualisation tools to discover the geographic structure of a domain*, Computers, environment and urban systems, vol 30, pp460-484.

Mothe, J. and Dousset, B. (2004) *Mining document contents in order to analyse a scientific domain*, Sixth International Conference on Social Science Methodology, 2004.

Neal, R. *Bayesian Learning for Neural Networks*, Springer-Verlag, New York, 1996.

Pozi. L (2002) *The Intellectual Structure and Interdisciplinary Breath of Knowledge Management : a Bolometric Study of Its Early Stage of Development*, Scientometrics 55, pp259-272.

Salton, G. (1991) *Developments in automatic text retrieval*, Science, 253, 974-980.

Schcaneveldt, R., Durso, F. and Dearholt, D. (1989) *Network structures in proximity data*, The Psychology of Learning and Motivation, 24, Academic Press, pp249-284.

Shannon, C. (1948) *A mathematical theory of communication.* Bell System Technical Journal, vol. 27, pp. 379–423 and 623–656, July and October, 1948

White, H. and Griffith, B. (1981) *Author co-citation: A Literature measure of intellectual structure*, Journal of the American Society for Information Science, 32, 163-172.

White, H. And McCain, K. (1995) *visualising a discipline: an author co-citation analysis of information science*, Journal of American Society for Information Science, 49(4), 327-356.

18

Dynamic Visual Motion Estimation

Volker Willert

1. Introduction

Visual motion is the projection of scene movements on a visual sensor. It is a rich source of information for the analysis of a visual scene. Especially for dynamic vision systems the estimation of visual motion is important because it allows to deduce the motion of objects as well as the self-motion of the system relative to the environment. Therefore, visual motion serves as a basic information for navigation and exploration tasks, like obstacle avoidance, object tracking or visual scene decomposition into static and moving parts.

Despite many years of progress, visual motion processing continues to puzzle the mind of researchers involved in understanding the principles of visual perception. Basic aspects such as measuring motions of spatially local entities have been widely studied. But what is most striking about motion processing is its temporal dynamics. This is obvious, because the environment perceived by a visual observer like a video camera or the human eye is highly dynamic. Moving objects enter and leave the field of view and also change the way they move, e.g. change the direction or speed. Hence, suitable assumptions about the dynamics of the visual scene and about the correlations between local moving entities are beneficial for the estimation of the scene motion as a whole.

Probabilistic machine learning techniques have become very popular for early vision problems like binocular depth and optical flow computation. The reason for the popularity is because of the possibility to explicitly consider uncertainties inherent in the measurement processes and to incorporate prior knowledge about the state to be estimated. Along this line of argumentation, we present a general approach to visual motion estimation based on a probabilistic generative model that allows to infer visual motion from visual data. We start with a definition of visual motion and point out the basic problems that come along with visual motion estimation. Then, we summarize common ideas that can be found in different state-of-the-art optical flow estimation techniques and stress the need for taking uncertainty into account. Based on the ideas of already existing models we introduce a general Bayesian framework for dynamic optical flow estimation that comprises several different aspects for solving the optical flow estimation problem into one common approach.

So far, the research on optical flow has mainly concentrated on motion estimations using the observation of two frames of an image sequence isolated in time. Our main concern is to stress that visual motion is a dynamic feature of an image input stream and the more visual data has been observed the more precise and detailed we can estimate and predict the motion contained in this visual data. Therefore both motion prediction and observation integration should be explicitly modeled in an inference procedure for optical flow estimation. Recently

- because of the development of efficient loopy belief propagation algorithms - Markov Random Fields regained great popularity to impose smoothness priors on motion measurements. On the contrary side, a way to impose consistent priors that has not been paid much attention lately is to treat the motion estimation as a dynamical system, like in Kalman-Filter approaches, to propagate motion information along time or scale assuming smoothness along the time or scale dimension.

In this chapter, we propose to fuse both ideas - spatial smoothness and smoothness along time and scale - into one common predictive prior model. This allows the formulation of a probabilistic dynamical system to infer visual motion via spatiotemporal belief propagation. The main contribution is the proposal of a certain class of transition probability functions which satisfy a probability mixture model and allow for temporal prediction along with spatial smoothing. For this class of transitions combined with additional factorization assumptions and approximate inference techniques it is possible to get a computationally tractable probabilistic optical flow filter. To show the capability, the benefits, and the drawbacks of the framework, we derive two realizations: one for continuous Gaussian and one for discrete grid-based observation likelihoods as well as Mixture of Gaussians and Mixture of Student's-t-distributions transitions.

2. Problems of optical flow computation

2.1 Visual motion

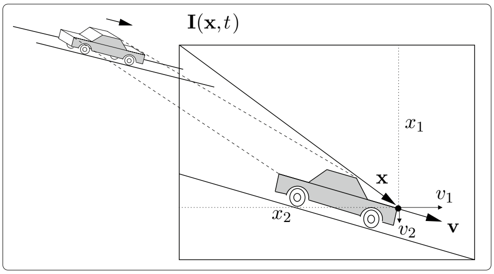


Fig. 1. How a moving object, e.g. a car, is projected onto a surface.

If we move or if objects move through our field of view then the projection of the environment onto the retina of our eyes changes. The projections over time form patterns of spatiotemporal brightness changes that encode the movement of the projections called the visual motion J.J.Gibson (1950). This matter of fact has motivated researchers from diverse areas like psychology of perception, visual neuroscience or computer science to understand the principles and model the mechanisms behind visual motion estimation.

More technically spoken, image sequences captured from a camera stream are mappings from the three-dimensional world onto a two-dimensional light-sensing surface - usually a digital camera-chip - over time. The image brightness $\mathbf{I}(\mathbf{x},t)$ at each spatial position $\mathbf{x} = (x_1\ x_2)^T$ at a particular time t (or over some interval of time) is a measurement how much light fell on the surface. When an object in the world moves relative to this projection surface, the two-dimensional projection of that object moves within the image sequence. The movement of the projected position of each point in the world refers to a velocity vector $\mathbf{v}(\mathbf{x},t) = (v_1\ v_2)^T$ and the set of all these vectors is called the motion field Simoncelli (2003). Figure 1 shows a sketch of the relations just explained.

2.2 Brightness constancy assumption

As mentioned beforehand, the motion field can be estimated using spatiotemporal brightness patterns. In principle, the quality of any estimation problem depends on the unambiguity of the relation between the observed data and the state to be inferred Kay (1993). In our case this is the relation between the observed spatiotemporal brightness and the motion field. The basic approach to visual motion estimation assumes constant brightness during the time brightness is measured Horn & Schunk (1981). This assumption implies that changes in brightness can only be caused by translational movements of the projections onto the image surface with constant velocity. Using the brightness constancy assumption the relation between the brightness $I(\mathbf{x},t)$ of an image pixel at position $\mathbf{x}$ at time t and the corresponding velocity vector $\mathbf{v}(\mathbf{x},t)$ during the time interval Δt reads

$$I(\mathbf{x}+\mathbf{v}(\mathbf{x},t)\Delta t, t+\Delta t) = I(\mathbf{x},t)\,. \tag{1}$$

If the time interval Δt is sufficiently small then the left side of equation (1) can be approximated via a first order Taylor series Simoncelli (1993) as follows

$$\nabla_{\mathbf{x}} I(\mathbf{x},t)\mathbf{v}(\mathbf{x},t) + \frac{\partial I(\mathbf{x},t)}{\partial t} = 0\,. \tag{2}$$

Here, $\nabla_{\mathbf{x}} I(\mathbf{x},t) = (\partial I(\mathbf{x},t)/\partial x_1, \partial I(\mathbf{x},t)/\partial x_2)$ denotes the spatial brightness gradient and $\partial I(\mathbf{x},t)/\partial t$ the temporal derivative of the spatiotemporal brightness pattern. Both equations (1) and (2) are ill-posed problems because they cannot be solved for the velocity unambiguously. Furthermore, the displacement $\mathbf{v}\Delta t$ is only an approximation to the true visual flow, called the optical flow Beauchemin & Barron (1995), because the brightness could also change due to changes in the lighting conditions. Nevertheless, if there is (i) constant illumination during the time interval Δt (ii) perpendicular projection of the reflected light onto the image surface and (iii) purely translational motion of the object parallel to the image surface, then optical flow is equivalent to the true motion field Beauchemin & Barron (1995). There are further approaches to optical flow computation like the phase-based approach which describes flow in fourier domain. A detailed description of all the different methods as well as exhaustive comparisons can be found in Barron et al. (1994).

2.3 Motion ambiguity

Besides incomplete models (1, 2) there are a series of fundamental problems concerning motion estimation. The movement of an isolated pixel cannot be estimated without considering its neighboring pixels. Therefore, only using the brightness constancy assumption (1) or the continuity equation (2) is insufficient. The structure of the local brightness pattern composed by the neighboring pixels directly affects the uncertainty of the motion estimate of the center pixel. The velocity $\mathbf{v}(\mathbf{x},t)$ is only unique if the brightness within the brightness pattern varies along the two spatial dimensions. That means, the brightness gradient $\nabla I(\mathbf{x},t)$ must not be zero for any dimension $\mathbf{x} = (x_1\ x_2)^T$. If this condition is not satisfied it is not possible to find an unambiguous correspondence between pixels (brightness patterns) from temporal consecutive image frames, an issue which is also called the aperture or correspondence problem Jähne (1997).

In figure 2 four types of motion uncertainties caused by an ambiguity in the correspondence of brightness patterns are shown. Here, a car drives down a road and the motion of four different parts A-D of the car should be estimated. Although all parts move with the same velocity the uncertainties of the estimates differ. Part A exhibits no structure in brightness

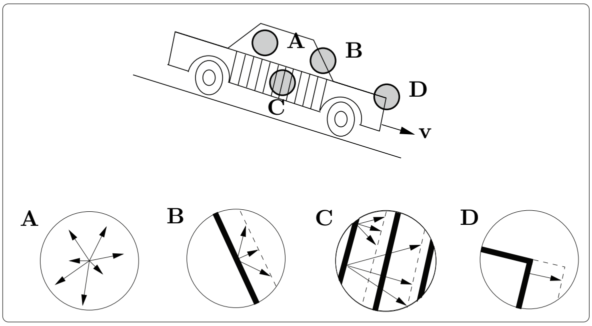


Fig. 2. Different types of motion uncertainties caused by ambiguous correspondence of brightness patterns.

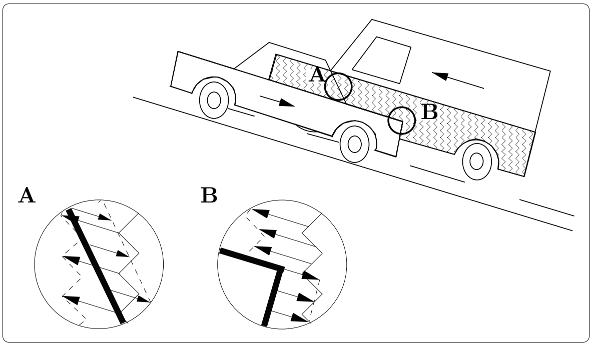


Fig. 3. Motion ambiguities caused by incoherently moving image parts and occlusion at motion discontinuities.

at all. Therefore, the brightness is constant over both spatial position and time. In this case, the brightness places no constraint on the velocity Simoncelli (2003). The local brightness of parts B and C varies only in one direction. In these cases, only the velocity component that is perpendicular to the edge direction is constrained. Along the edges the brightness does not change and thus the velocity component parallel to the edge direction cannot be estimated. In part D, the local brightness varies along two dimensions, in which case the optical flow vector is uniquely constrained and the velocity estimate gets unambiguous.

Another problem that forces ambiguities are pixel occlusions at motion boundaries. This is depicted in figure 3. Part A and B show two situations with brightness patterns overlapping while moving. This leads to occlusion of neighboring pixels that have been seen beforehand and newly appearing pixels that have been occluded in the past. The brightness pattern within a neighborhood of pixels along such motion boundaries are not stable over time and therefore temporal correspondences cannot be found unambiguously.

3. Motion disambiguation

To be able to capture all these ambiguities some authors Simoncelli et al. (1991); Zetzsche & Krieger (2001) propose to introduce uncertainty to the model that describes the relation between image intensities and pixel velocities. For this purpose, the velocity of an image location and the images of a sequence are understood as statistical signals

$$I(\mathbf{x}+(\mathbf{v}(\mathbf{x},t)+\eta_{\mathbf{v}})\Delta t,t+\Delta t)=I(\mathbf{x},t)+\eta_I\,, \tag{3}$$

with $\eta_{\mathbf{v}}$ defining additive noise on the velocity and η_I additive noise on the image intensity. More general, a function F_I

$$F_I\Big(I\big(\mathbf{x}+(\mathbf{v}(\mathbf{x},t)+\eta_{\mathbf{v}})\Delta t,t+\Delta t\big),I\big(\mathbf{x},t\big)\Big)=\eta_I\,, \tag{4}$$

defines the relation between temporal consecutive images and the flow field. This implies probabilities for the existence of image intensities and velocities namely conditional probability density functions (pdf)

$$P\big(I(\mathbf{x},t+\Delta t)|I(\mathbf{x},t),\mathbf{v}(\mathbf{x},t)\big) \tag{5}$$

that represent the uncertain relation between image observations $I(\mathbf{x},t+\Delta t)$ conditioned on image observations $I(\mathbf{x},t)$ and on the hidden states $\mathbf{v}(\mathbf{x},t)$. Using Bayes' rule and some prior knowledge about the distribution of the velocity $P(\mathbf{v}(\mathbf{x},t))$ and the intensities $P(I(\mathbf{x},t))$ the conditional pdf for the velocity conditioned on the image data can be inferred

$$\begin{aligned}
P\big(I(\mathbf{x},t+\Delta t),I(\mathbf{x},t),\mathbf{v}(\mathbf{x},t)\big) &= P\big(I(\mathbf{x},t+\Delta t)|I(\mathbf{x},t),\mathbf{v}(\mathbf{x},t)\big)\times P\big(I(\mathbf{x},t)\big)\times P\big(\mathbf{v}(\mathbf{x},t)\big)\\
&= P\big(\mathbf{v}(\mathbf{x},t)|I(\mathbf{x},t+\Delta t),I(\mathbf{x},t)\big)\times P\big(I(\mathbf{x},t+\Delta t)\big)\times P\big(I(\mathbf{x},t)\big)\,,\\
P\big(\mathbf{v}(\mathbf{x},t)|I(\mathbf{x},t+\Delta t),I(\mathbf{x},t)\big) &= \frac{P\big(I(\mathbf{x},t+\Delta t)|I(\mathbf{x},t),\mathbf{v}(\mathbf{x},t)\big)\times P\big(\mathbf{v}(\mathbf{x},t)\big)}{P\big(I(\mathbf{x},t+\Delta t)\big)}\,.
\end{aligned} \tag{6}$$

So far, the velocities $\mathbf{v}(\mathbf{x},t)$ and the image intensities $I(\mathbf{x},t)$ have been assumed to be independent for different positions $\mathbf{x}$ and times t. This is already a strong approximation. In the next section this approximation is relaxed towards more spatiotemporal dependence. The expectation is that pdfs are able to tackle the addressed ambiguity problems related to motion processing. As can be seen in Simoncelli (1993); Weiss (1993); Zelek (2002) specific information about the mentioned problems can, in principle, be extracted from the shape of the pdfs.

During the last ten years velocity distributions have been suggested and discussed by several authors Simoncelli et al. (1991); Singh (1990); Weiss & Fleet (2002); Wu (1995). Additionally, a lot of different mainly deterministic approaches have been developed for visual motion estimation Baker et al. (2007); Beauchemin & Barron (1995). All these approaches are based on common constraints on the flow field that are suitable for disambiguation and improvement of the estimates. The constraints F define correlations between motion estimations at different points in image location $\mathbf{x}$, different points in time t or different image scales k. Assuming independence between the dimensions $\mathbf{x}$, t, and k the constraints can be formulated in a probabilistic way as

$$F_x\big(\mathbf{v}(\mathbf{x},t,k),\{\mathbf{v}(\mathbf{x}',t,k)\}_{\mathbf{x}'}\big)=\eta_x\,, \tag{7}$$

$$F_t\big(\mathbf{v}(\mathbf{x},t,k),\{\mathbf{v}(\mathbf{x},t',k)\}_{t'}\big)=\eta_t\,, \tag{8}$$

$$F_k\big(\mathbf{v}(\mathbf{x},t,k),\{\mathbf{v}(\mathbf{x},t,k')\}_{k'}\big)=\eta_k\,. \tag{9}$$

Here, $\{\mathbf{v}(\mathbf{x}',t,k)\}_{\mathbf{x}'}$ denotes the set of spatial neighbors to pixel velocity $\mathbf{v}(\mathbf{x},t,k)$. The constraint is defined by the function F_x and η_x considers additive noise on the constraining model.

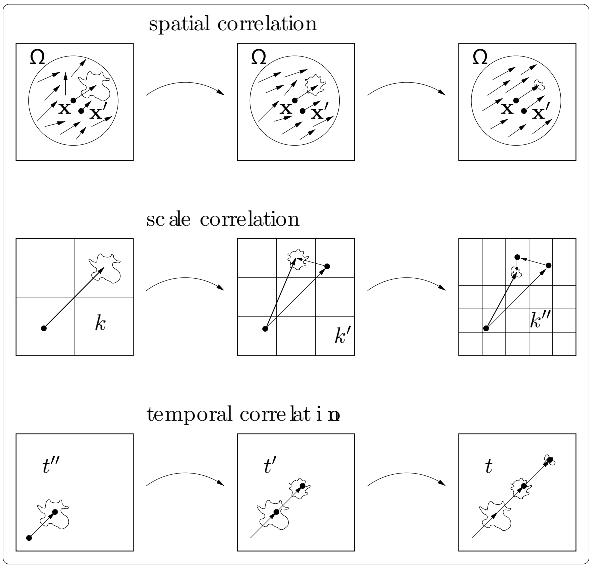


Fig. 4. Three basic principles to disambiguate visual motion using constraints along the space, time, or scale dimension.

For the set of temporal t' and scale k' neighbors the notations are analogous. The most general case would be to assume total dependence between all dimensions

$$F\big(\mathbf{v}(\mathbf{x},t,k), \{\mathbf{v}(\mathbf{x}',t,k)\}_{\mathbf{x}'}, \{\mathbf{v}(\mathbf{x},t',k)\}_{t'}, \{\mathbf{v}(\mathbf{x},t,k')\}_{k'}\big) = \eta\,. \tag{10}$$

Unfortunately, considering full dependence together with a probabilistic treatment leads to combinatorial explosion and is in most cases computationally intractable.

A sketch of the three different types of independent disambiguation along the three dimensions is given in figure 4. The most established method to reduce ambiguities is the integration of motion information over space (see Fig. 4 first row). That means, interactions between neighboring velocities or even higher order derivations are considered Anandan (1989); Beauchemin & Barron (1995); Horn & Schunk (1981); Lukas & Kanade (1981). This is often accounted for by smoothness constraints for neighboring velocities assuming that all pixels within the neighborhood Ω move similarly.

Further improvements are made using multiscale approaches (see Fig. 4 second row). This is desirable, e.g., for being able to represent both large and small velocities at coarse and fine resolutions with a reasonable effort. It is usually done in such a way that the larger velocities at coarser scale are calculated first, then a warped version of the image is calculated using these large, coarsely sampled velocities, and afterwards the residual velocities at the next finer scale are calculated, since they have been calculated in a frame that is moving along with the velocities extracted from coarser scale Anandan (1989); Bergen et al. (1992); Brox et al. (2004); Memin & Perez (1998); Weber & Malik (1995).

Another important aspect of motion estimation is the fact that motion is a dynamic feature of an image sequence. Thus, the longer we observe a movement the more precisely we can estimate and predict its characteristics. This has motivated several approaches Black (1994);

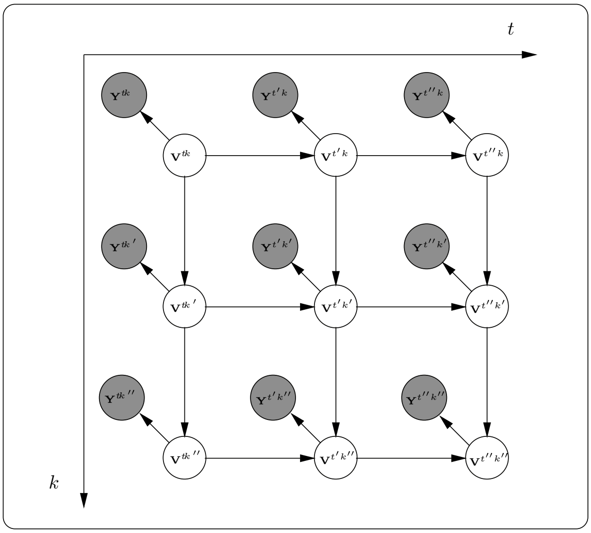


Fig. 5. A probabilistic directed graphical model for visual motion estimation. Here, $t' = t + 1$ and $t'' = t + 2$ denote future timesteps and $k' = k + 1$ and $k'' = k + 2$ denote finer scales. Observable nodes are shaded gray, hidden nodes are white.

Elad & Feuer (1998); Singh (1991) to recursively estimate the optical flow over time including a prediction model that defines some temporal relation between pixel movements (see Fig. 4 third row). For prediction, a model for the underlying dynamics is needed to predict image motion.

Some authors work in a probabilistic framework assuming that velocity distributions are Gaussian parameterized by a mean and covariance. Kalman filtering can then be used to properly combine the information from scale to scale or time to time taking into account uncertainties of the measurements Simoncelli (1999); Singh (1991). The presumption of Gaussian distributed velocity measurements is sometimes incomplete because velocity distributions are often multimodal or ambiguous Simoncelli et al. (1991); Weiss & Fleet (2002), especially at motion boundaries. To circumvent this problem, particle filtering methods for non-Gaussian velocity distributions have recently been used to improve motion estimation for tracking single or multiple objects in a scene Isard & Blake (1998); Rosenberg & Werman (1997).

4. A dynamical system for motion estimation

Having all the different possibilities of motion disambiguation in mind, the question is how to unify them in one general framework for motion estimation. In this section we derive a quite general probabilistic solution which is still computationally tractable.

4.1 Dynamic Bayesian Network

A Dynamic Bayesian Network (DBN) is a directed graphical model of a dynamic stochastic process. Here, we propose such a network as depicted in figure 5 to model the dynamics of visual motion. The structure of the graphical model in figure 5 is similar to a Markov random field. The difference is that the edges are directed. As can be seen, it tightly couples several

Markov chains along time that are defined for each scale k via *Markov chains along scale* defined at each time step t. Note that the DBN forces an independency structure. The probability that a node is in one of its states depends directly only on the states of its parents Yedidia et al. (2003).

We assume a generative model for the observables $\mathbf{Y}^{tk}$ of an image sequence $\mathbf{I}^{1:T,1:K}$ with T images at equidistant points in time $t \in \mathcal{T}$ at K spatial resolution scales $k \in \mathcal{K}$ with $t' = t + 1$ and $k' = k + 1$ being the next time step and the next finer scale, respectively. Without loss of generalization we define the time intervals $\Delta t = 1$ and scale intervals $\Delta k = 1$ to be unity. Here, the observable $\mathbf{Y}^{tk}$ comprises image data of several frames within a time interval around t at the same scale k. For example, $\mathbf{Y}^{tk} = (\mathbf{I}^{tk}, \mathbf{I}^{t'k})$ has to be at least a pair of images with both images being defined over the same image range X^k at the same scale k but at consecutive points in time t and t'. Each image $\mathbf{I}^{tk}$ consists of image intensities $I^{tk}_{\mathbf{x}}$ at each image position $\mathbf{x} \in \mathcal{X}^k$. Similarly, the hidden state $\mathbf{V}^{tk}$ is a flow field at time slice t and scale k defined over the image range X^k with velocity vectors $\mathbf{v}^{tk}_{\mathbf{x}}$ at each image position $\mathbf{x}$.

4.2 Generative model

The probabilistic generative model is precisely defined by the following probabilities and factorization assumptions:

First, an initial prior for the flow field at time $t = 1$ and scale $k = 1$

$$P(\mathbf{V}^{11}) = \prod_{\mathbf{x}} P(\mathbf{v}^{11}_{\mathbf{x}}), \tag{11}$$

defining some preference for the speed and direction of the velocities in the flow field. Often this is chosen to be a product of zero mean Gaussian distributions to prefer slow and smooth velocities Weiss & Fleet (2002). Second, the specification of the *observation likelihood* for the images $\mathbf{Y}^{tk}$ given the flow $\mathbf{V}^{tk}$ for all times $t \in \mathcal{T}$ and scales $k \in \mathcal{K}$

$$P(\mathbf{Y}^{tk}|\mathbf{V}^{tk}) = \prod_{\mathbf{x}} \ell(\mathbf{Y}^{tk}, \mathbf{v}^{tk}_{\mathbf{x}}). \tag{12}$$

This factorisation assumption is somewhat unusual because we do not assume the image observation to factorize in pixel observations but assume the observation likelihood to factorize in the velocities only. And third, the specification of the *transition probabilities* for the flow fields $\mathbf{V}^{t'k'}$ at the new timestep t' at finer scale k' given the flow field $\mathbf{V}^{t'k}$ at the same time t' but coarser scale k and the flow field $\mathbf{V}^{tk'}$ from last time t but at the same scale k'. For the first time slice $t = 1$ and the coarsest scale $k = 1$ the transitions are conditioned only on one flow field $\mathbf{V}^{1k}$ or $\mathbf{V}^{t1}$.

$$\begin{aligned}
P(\mathbf{V}^{1k'}|\mathbf{V}^{1k}) &= \prod_{\mathbf{x}} \phi_k(\mathbf{v}^{1k'}_{\mathbf{x}}, \mathbf{V}^{1k}), \\
P(\mathbf{V}^{t'1}|\mathbf{V}^{t1}) &= \prod_{\mathbf{x}} \phi_t(\mathbf{v}^{t'1}_{\mathbf{x}}, \mathbf{V}^{t1}), \\
P(\mathbf{V}^{t'k'}|\mathbf{V}^{t'k}, \mathbf{V}^{tk'}) &= \prod_{\mathbf{x}} \phi_k(\mathbf{v}^{t'k'}_{\mathbf{x}}, \mathbf{V}^{t'k}) \phi_t(\mathbf{v}^{t'k'}_{\mathbf{x}}, \mathbf{V}^{tk'}).
\end{aligned} \tag{13}$$

These equations explicitly express that the probability distribution for each flow field factorises into independent distributions for each velocity vector. Nevertheless, although each velocity vector is not dependent on velocity vectors from the flow field at the same time and scale it heavily depends on all the velocity vectors from the flow fields at coarser scale and

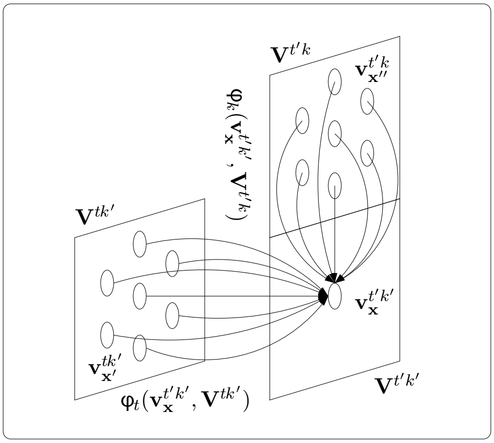

$\mathbf{V}^{t'k}$
$\mathbf{v}_{\mathbf{x}''}^{t'k}$
$\phi_k(\mathbf{v}_{\mathbf{x}}^{t'k'}, \mathbf{V}^{t'k})$
$\mathbf{V}^{tk'}$
$\mathbf{v}_{\mathbf{x}}^{t'k'}$
$\mathbf{v}_{\mathbf{x}'}^{tk'}$
$\phi_t(\mathbf{v}_{\mathbf{x}}^{t'k'}, \mathbf{V}^{tk'})$
$\mathbf{V}^{t'k'}$

Fig. 6. Pairwise potentials of the scale-time transition probability.

past time. Further on, the conditional dependence $P(\mathbf{v}_{\mathbf{x}}^{t'k'}|\mathbf{V}^{t'k},\mathbf{V}^{tk'})$ can be split in two pairwise potentials ϕ_k, ϕ_t. This will allow us to maintain only factored beliefs during inference, which makes the approach computationally practicable.

4.3 A general class of flow field transitions

To further specify the generative model we have to define the formulas for the prior (11), the observation likelihood (12), and the transitions (13). Some concrete examples are given in the next section 5. For the flow field transitions in equation (13) we propose a certain class of transition probability functions which satisfy a probability mixture model. Equation (13) consists of two pairwise potentials. The first potential $\phi_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{V}^{tk'})$ assumes that the flow field at every spatial scale k transforms from $t \to t'$ according to itself. The second potential $\phi_k(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{V}^{t'k})$ realizes a refinement from coarser to finer scale $k \to k'$ at every time t'. A sketch of the information flow is shown in figure 6.

To motivate the temporal transition factor $\phi_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{V}^{tk'})$ we assume that the direction and speed of a flow vector $\mathbf{v}_{\mathbf{x}}^{t'k'}$ at position $\mathbf{x}$ at time t' is functionally related to a previous flow vector $\mathbf{v}_{\mathbf{x}'}^{tk'}$ at some corresponding position $\mathbf{x}'$ at time t,

$$\mathbf{v}_{\mathbf{x}}^{t'k'} \sim f_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{v}_{\mathbf{x}'}^{tk'};\theta_t) \; , \tag{14}$$

including some free parameters θ_t that allow for adaptation of the temporal relation. Now, asking what the corresponding position $\mathbf{x}'$ in the previous image was, we assume that we can infer it from the flow field itself as follows

$$\mathbf{x}' \sim f_{xt}(\mathbf{x}',\mathbf{x} - \mathbf{v}_{\mathbf{x}}^{t'k'};\theta_{xt}) \; . \tag{15}$$

In principle f_{xt} can be any arbitrary function that defines the relation between neighboring positions. Again the free parameters θ_{xt} allow for adaptation of the spatial relation. Note that here we use $\mathbf{v}_{\mathbf{x}}^{t'k'}$ to retrieve the previous corresponding point $\mathbf{x}'$. This is a suitable approximation as long as the similarity $\mathbf{v}_{\mathbf{x}}^{t'k'} \approx \mathbf{v}_{\mathbf{x}'}^{tk'}$ is not heavily violated. Combining both factors (14) and (15) and integrating $\mathbf{x}'$ leads to the first pairwise potential

$$\phi_t(\mathbf{v}_{\mathbf{x}}^{t'k'}, \mathbf{V}^{tk'}) = \sum_{\mathbf{x}'} f_{xt}(\mathbf{x}', \mathbf{x} - \mathbf{v}_{\mathbf{x}}^{t'k'}; \theta_{xt}) f_t(\mathbf{v}_{\mathbf{x}}^{t'k'}, \mathbf{v}_{\mathbf{x}'}^{tk'}; \theta_t)\,. \tag{16}$$

Equivalent to (14) for the scale transition factor $\phi_k(\mathbf{v}_{\mathbf{x}}^{1k'}, \mathbf{V}^{1k})$ we assume that the origin of a flow vector $\mathbf{v}_{\mathbf{x}}^{t'k'}$ at position $\mathbf{x}$ at finer scale k' corresponds to a flow vector $\mathbf{v}_{\mathbf{x}''}^{t'k}$ from coarser scale k at some corresponding position $\mathbf{x}''$,

$$\mathbf{v}_{\mathbf{x}}^{t'k'} \sim f_k(\mathbf{v}_{\mathbf{x}}^{t'k'}, \mathbf{v}_{\mathbf{x}''}^{t'k}; \theta_k)\,. \tag{17}$$

Since it is uncertain how strong a position $\mathbf{x}''$ at coarser scale k influences the velocity estimate at position $\mathbf{x}$ at finer scale k', we assume that we can infer it from the neighborhood similar to (15)

$$\mathbf{x}'' \sim f_{xk}(\mathbf{x}'', \mathbf{x}; \theta_{xk})\,. \tag{18}$$

The considerations for the scale transition are analogous to the ones for the temporal transition. Again, combining both factors (17) and (18) and integrating $\mathbf{x}''$ we get the second pairwise potential

$$\phi_k(\mathbf{v}_{\mathbf{x}}^{t'k'}, \mathbf{V}^{t'k}) = \sum_{\mathbf{x}''} f_{xk}(\mathbf{x}'', \mathbf{x}; \theta_{xk}) f_k(\mathbf{v}_{\mathbf{x}}^{t'k'}, \mathbf{v}_{\mathbf{x}''}^{t'k}; \theta_k)\,, \tag{19}$$

that imposes a spatial smoothness constraint on the flow field via spatial weighting of motion estimations from coarser scale. The combination of both potentials (16) and (19) results in the complete conditional flow field transition probability as given in (13). The transition factors (16) and (19) allow us to *unroll* two different kinds of spatial constraints along the temporal and the scale axes while adapting the free parameters for scale and time transition differently. This is done by splitting not only the transition in two pairwise potentials, one for the temporal- and one for the scale-transition, but also every potential in itself in two factors, one for the transition noise and the other one for an additional spatial constraint. In this way, the coupling of the potentials (16) and (19) realizes a combination of (A) scale-time prediction and (B) an integration of motion information neighboring in time, in space, and in scale.

4.4 Inference

The overall data likelihood $P(\mathbf{Y}^{1:T,1:K}, \mathbf{V}^{1:T,1:K})$ is assumed to factorize as defined by the directed graph in figure 5

$$\begin{aligned} P(\mathbf{Y}^{1:T,1:K}, \mathbf{V}^{1:T,1:K}) \quad = \quad & \prod_{t=1}^{T} \prod_{k=1}^{K} P(\mathbf{Y}^{tk} | \mathbf{V}^{tk}) \times P(\mathbf{V}^{11}) \times \\ & \prod_{t=1}^{T-1} P(\mathbf{V}^{t'1} | \mathbf{V}^{t1}) \prod_{k=1}^{K-1} P(\mathbf{V}^{1k'} | \mathbf{V}^{1k}) P(\mathbf{V}^{t'k'} | \mathbf{V}^{t'k}, \mathbf{V}^{tk'})\,. \end{aligned} \tag{20}$$

What we are usually interested in is the probability for some flow field given all the data aquired so far. For the offline case where all the data of a sequence is accessible this would be the probability $P(\mathbf{V}^{tk} | \mathbf{Y}^{1:T,1:K})$. For the online case where only the past data is accessible the probability $P(\mathbf{V}^{tk} | \mathbf{Y}^{1:t,1:K})$ would be interesting. To infer these probabilities, Bayes' rule and marginalization has to be applied. For the offline case this reads

$$P(\mathbf{V}^{tk} | \mathbf{Y}^{1:T,1:K}) = \sum_{\mathbf{V}^{1:T,1:K} \setminus \mathbf{V}^{tk}} \frac{P(\mathbf{Y}^{1:T,1:K}, \mathbf{V}^{1:T,1:K})}{P(\mathbf{Y}^{1:T,1:K})}\,. \tag{21}$$

The online case neglects future observations and simplifies to

$$P(\mathbf{V}^{tk}|\mathbf{Y}^{1:t,1:K}) = \sum_{\mathbf{V}^{1:t,1:K}\setminus\mathbf{V}^{tk}} \frac{P(\mathbf{Y}^{1:t,1:K}, \mathbf{V}^{1:t,1:K})}{P(\mathbf{Y}^{1:t,1:K})}. \tag{22}$$

Either for the online or offline case, the direct computation of the marginals using equation (21) or (22) would take exponential time Yedidia et al. (2003). The most prominent solution to this problem is *Belief Propagation* (BP) which is a very efficient approximate inference algorithm especially applicable if the graph has a lot of loops and many hidden nodes like it is the case for our graphical model for dynamic motion estimation (see figure 5).

4.5 Approximate inference

Here, we propose an approximate inference algorithm based on Belief Propagation and restrict ourselves to the online case (22) since its extension to the offline case is straightforward Bishop (2006) (see also section 5). The marginal probabilities that are now computed only approximately are called *beliefs* and here we use α's as the notation for forward filtered beliefs

$$\alpha(\mathbf{v}_{\mathbf{x}}^{tk}) \approx P(\mathbf{v}_{\mathbf{x}}^{tk}|\mathbf{Y}^{1:t,1:K}). \tag{23}$$

Let us start with the inference of the flow field at first time slice $t = 1$ and coarsest scale $k = 1$ just having access to the observable $\mathbf{Y}^{11}$. Applying Bayes' rule we get

$$\alpha(\mathbf{v}_{\mathbf{x}}^{11}) = P(\mathbf{v}_{\mathbf{x}}^{11}|\mathbf{Y}^{11}) = \frac{\ell(\mathbf{Y}^{11}, \mathbf{v}_{\mathbf{x}}^{11})P(\mathbf{v}_{\mathbf{x}}^{11})}{P(\mathbf{Y}^{11})}. \tag{24}$$

This is the initial belief that has to be propagated along time and scale. To derive an approximate forward filter suitable for online applications we propose the following message passing scheme that realizes a recurrent update of the beliefs. Let us assume, we isolate one time slice at time t and neglect all past and future beliefs, then we would have to propagate the messages $m_{k\rightarrow k'}$ from coarse to fine and the messages $m_{k'\rightarrow k}$ from fine to coarse to compute a belief over the scale Markov chain. Similarly, if we isolate one scale k for all time slices and neglect all coarser and finer beliefs, then we would have to propagate the messages $m_{t\rightarrow t'}$ from the past to the future and the messages $m_{t'\rightarrow t}$ from the future to the past to compute a belief over the temporal Markov chain. For the realization of a forward scale-time filter, we combine the forward passing of temporal messages $m_{t\rightarrow t'}$ and the computation of the likelihood messages $m_{Y\rightarrow v} = \ell(\mathbf{Y}^{t'k'}, \mathbf{v}_x^{t'k'})$ at all scales k. As a simplification we restrict ourselves to propagating messages only in one direction $k \rightarrow k'$ and neglect passing back the message $m_{k'\rightarrow k}$. The consequence of this is that not all the $\mathbf{V}$-nodes at time t have seen all the data $\mathbf{Y}^{1:t,1:K}$ but only all past data up to the current scale $\mathbf{Y}^{1:t,1:k}$. This reduces computational costs but the flow field on the finest scale $\mathbf{V}^{t,K}$ is now the only node that sees all the data $\mathbf{Y}^{1:t,1:K}$. Nevertheless, we also tested passing back the messages $m_{k'\rightarrow k}$ which only slightly improved the accuracy but increased computational costs.

The factored observation likelihood and the transition probability we introduced in (12) and (13) ensure that the forward propagated joint belief

$$P(\mathbf{V}^{t,1:K}|\mathbf{Y}^{1:t,1:K}) = \prod_{\mathbf{x}} P(\mathbf{v}_{\mathbf{x}}^{t,1:K}|\mathbf{Y}^{1:t,1:K})$$

will remain factored. Similar to BP in a Markov random field, we assume independency for all neighboring nodes in the Markov blanket. This means the belief over $\mathbf{V}^{tk}$ and $\mathbf{V}^{tk'}$ at time t is assumed to be factored which implies that also the belief over $\mathbf{V}^{t'k}$ and $\mathbf{V}^{tk'}$ factorizes.

$$P(\mathbf{V}^{t'k},\mathbf{V}^{tk'}|\mathbf{Y}^{1:t',1:k'} \setminus \mathbf{Y}^{t'k'}) = P(\mathbf{V}^{t'k}|\mathbf{Y}^{1:t',1:k})P(\mathbf{V}^{tk'}|\mathbf{Y}^{1:t,1:k'}) = \prod_{\mathbf{x}} \alpha(\mathbf{v}_{\mathbf{x}}^{t'k})\alpha(\mathbf{v}_{\mathbf{x}}^{tk'})\,, \tag{25}$$

where we used $\setminus$ as the notation for excluding $\mathbf{Y}^{t'k'}$ from the set of measurements $\mathbf{Y}^{1:t',1:k'}$. The two-dimensional forward filter propagates the belief over $\mathbf{V}^{t'k}$ and $\mathbf{V}^{tk'}$ from (25) via multiplying with the scale-time transition (13) and marginalizing over $\mathbf{V}^{t'k}$ and $\mathbf{V}^{tk'}$. The result is multiplied with the new observation likelihood (12) and normalized by $P(\mathbf{Y}^{t'k'})$ to get the updated belief

$$\begin{aligned}
P(\mathbf{v}_{\mathbf{x}}^{t'k'}|\mathbf{Y}^{1:t',1:k'}) &= \tfrac{1}{P(\mathbf{Y}^{t'k'})} \quad \ell(\mathbf{Y}^{t'k'},\mathbf{v}_{\mathbf{x}}^{t'k'}) \sum_{\mathbf{V}^{t'k}} \sum_{\mathbf{V}^{tk'}} P(\mathbf{v}_{\mathbf{x}}^{t'k'}|\mathbf{V}^{t'k},\mathbf{V}^{tk'}) P(\mathbf{V}^{t'k},\mathbf{V}^{tk'}|\mathbf{Y}^{1:t',1:k'} \setminus \mathbf{Y}^{t'k'}),\\
\alpha(\mathbf{v}_{\mathbf{x}}^{t'k'}) &\propto \quad m_{Y\to v}(\mathbf{v}_{\mathbf{x}}^{t'k'}) \sum_{\mathbf{V}^{t'k}} \sum_{\mathbf{V}^{tk'}} \overbrace{\phi_k(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{V}^{t'k})\phi_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{V}^{tk'})}^{} \; \overbrace{\prod_{\mathbf{x}} \alpha(\mathbf{v}_{\mathbf{x}}^{t'k})\alpha(\mathbf{v}_{\mathbf{x}}^{tk'})}^{},\\
&\propto \quad m_{Y\to v}(\mathbf{v}_{\mathbf{x}}^{t'k'}) \underbrace{\sum_{\mathbf{V}^{t'k}} \phi_k(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{V}^{t'k}) \prod_{\mathbf{x}} \alpha(\mathbf{v}_{\mathbf{x}}^{t'k})}_{} \underbrace{\sum_{\mathbf{V}^{tk'}} \phi_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{V}^{tk'}) \prod_{\mathbf{x}} \alpha(\mathbf{v}_{\mathbf{x}}^{tk'})}_{},\\
&\propto \quad m_{Y\to v}(\mathbf{v}_{\mathbf{x}}^{t'k'}) \quad \times \quad m_{k\to k'}(\mathbf{v}_{\mathbf{x}}^{t'k'}) \quad \times \quad m_{t\to t'}(\mathbf{v}_{\mathbf{x}}^{t'k'})\,.
\end{aligned} \tag{26}$$

As can be seen, the complete scale-time forward filter can now be defined by the computation of updated beliefs α as the product of incoming messages,

$$\alpha(\mathbf{v}_{x}^{tk}) \propto m_{Y\to v}(\mathbf{v}_{\mathbf{x}}^{tk})\; m_{k\to k'}(\mathbf{v}_{\mathbf{x}}^{tk})\; m_{t\to t'}(\mathbf{v}_{\mathbf{x}}^{tk})\,. \tag{27}$$

Inserting the proposed class of temporal transitions (16) into (26) leads to the following temporal message

$$\begin{aligned}
m_{t\to t'}(\mathbf{v}_{\mathbf{x}}^{t'k'}) &= \sum_{\mathbf{V}^{tk'}} \phi_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{V}^{tk'}) \prod_{\mathbf{x}} \alpha(\mathbf{v}_{\mathbf{x}}^{tk'})\,,\\
&= \sum_{\mathbf{V}^{tk'}} \sum_{\mathbf{x}'} f_{xt}(\mathbf{x}',\mathbf{x}-\mathbf{v}_{\mathbf{x}}^{t'k'};\theta_{xt}) f_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{v}_{\mathbf{x}'}^{tk'};\theta_t) \prod_{\mathbf{x}} \alpha(\mathbf{v}_{\mathbf{x}}^{tk'})\,,\\
&= \sum_{\mathbf{v}_{\mathbf{x}}^{tk'}} \sum_{\mathbf{x}'} f_{xt}(\mathbf{x}',\mathbf{x}-\mathbf{v}_{\mathbf{x}}^{t'k'};\theta_{xt}) f_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{v}_{\mathbf{x}'}^{tk'};\theta_t) \alpha(\mathbf{v}_{\mathbf{x}}^{tk'}) \underbrace{\sum_{\substack{\mathbf{V}^{tk'}\\ \setminus \mathbf{v}_{\mathbf{x}'}^{tk'}}} \prod_{\mathbf{z}\neq\mathbf{x}'} \alpha(\mathbf{v}_{\mathbf{z}}^{tk'})}_{1}\,,\\
&= \sum_{\mathbf{x}'} f_{xt}(\mathbf{x}',\mathbf{x}-\mathbf{v}_{\mathbf{x}}^{t'k'};\theta_{xt}) \sum_{\mathbf{v}_{\mathbf{x}}^{tk'}} f_t(\mathbf{v}_{\mathbf{x}}^{t'k'},\mathbf{v}_{\mathbf{x}'}^{tk'};\theta_t) \alpha(\mathbf{v}_{\mathbf{x}'}^{tk'})\,.
\end{aligned} \tag{28}$$

Note that the summation $\sum_{\mathbf{V}^{tk'}}$ is summing over all possible flow fields, i.e. $\sum_{\mathbf{V}^{tk'}}$ represents X^k summations $\sum_{\mathbf{v}_{1,1}^{tk'}} \sum_{\mathbf{v}_{1,2}^{tk'}} \sum_{\mathbf{v}_{2,1}^{tk'}} \cdots$ over each local flow field vector. We separated these into a summation $\sum_{\mathbf{v}_{\mathbf{x}'}^{tk'}}$ over the flow field vector at $\mathbf{x}'$ and a summation $\sum_{\mathbf{V}^{tk'} \setminus \mathbf{v}_{\mathbf{x}'}^{tk'}}$ over all other flow field vectors at $\mathbf{x} \neq \mathbf{x}'$. Then, we use the equivalence $\sum_{\mathbf{V}^{tk'} \setminus \mathbf{v}_{\mathbf{x}'}^{tk'}} \prod_{\mathbf{z}\neq\mathbf{x}'} \alpha(\mathbf{v}_{\mathbf{z}}^{tk'}) =$

Algorithm 1 Scale-Time Filter

```
Initialize the priors $\alpha(\mathbf{v}_\mathbf{x}^{0,1:K})$

for $t' = 1$ to $T$ do
    for $k' = 1$ to $K$ do
        for $\mathbf{x} = 1$ to $X^{k'}$ do

            Compute the messages

            $m_{Y\to v}(\mathbf{v}_\mathbf{x}^{t'k'})$

            $m_{t\to t'}(\mathbf{v}_\mathbf{x}^{t'k'}) = \sum_{\mathbf{x}'} f_{xt}(\mathbf{x}', \mathbf{x} - \mathbf{v}_\mathbf{x}^{t'k'}; \theta_{xt}) \sum_{\mathbf{v}_\mathbf{x}^{tk'}} f_t(\mathbf{v}_\mathbf{x}^{t'k'}, \mathbf{v}_{\mathbf{x}'}^{tk'}; \theta_t) \alpha(\mathbf{v}_{\mathbf{x}'}^{tk'})$

            $m_{k\to k'}(\mathbf{v}_\mathbf{x}^{t'k'}) = \sum_{\mathbf{x}'} f_{xk}(\mathbf{x}', \mathbf{x}; \theta_{xk}) \sum_{\mathbf{v}_\mathbf{x}^{t'k}} f_k(\mathbf{v}_\mathbf{x}^{t'k'}, \mathbf{v}_{\mathbf{x}'}^{t'k}; \theta_k) \alpha(\mathbf{v}_{\mathbf{x}'}^{t'k})$

            Update the beliefs

            $\alpha(\mathbf{v}_x^{t'k'}) \propto m_{Y\to v}(\mathbf{v}_\mathbf{x}^{t'k'})\ m_{t\to t'}(\mathbf{v}_\mathbf{x}^{t'k'})\ m_{k\to k'}(\mathbf{v}_\mathbf{x}^{t'k'})$

        end for
    end for
end for
```

$\prod_{\mathbf{z}\neq\mathbf{x}'} \sum_{\mathbf{v}_\mathbf{z}^{tk'}} \alpha(\mathbf{v}_\mathbf{z}^{tk'}) = 1$. Similarly, we arrive at the scale message if we insert the scale transition (19) into (26)

$$
\begin{aligned}
m_{k\to k'}(\mathbf{v}_\mathbf{x}^{t'k'}) &= \sum_{\mathbf{V}^{t'k}} \phi_k(\mathbf{v}_\mathbf{x}^{t'k'}, \mathbf{V}^{t'k}) \prod_{\mathbf{x}} \alpha(\mathbf{v}_\mathbf{x}^{t'k}), \\
&= \sum_{\mathbf{V}^{t'k}} \sum_{\mathbf{x}'} f_{xk}(\mathbf{x}', \mathbf{x}; \theta_{xk}) f_k(\mathbf{v}_\mathbf{x}^{t'k'}, \mathbf{v}_{\mathbf{x}'}^{t'k}; \theta_k) \prod_{\mathbf{x}} \alpha(\mathbf{v}_\mathbf{x}^{t'k}), \\
&= \sum_{\mathbf{v}_\mathbf{x}^{t'k}} \sum_{\mathbf{x}'} f_{xk}(\mathbf{x}', \mathbf{x}; \theta_{xk}) f_k(\mathbf{v}_\mathbf{x}^{t'k'}, \mathbf{v}_{\mathbf{x}'}^{t'k}; \theta_k) \alpha(\mathbf{v}_{\mathbf{x}'}^{t'k}) \underbrace{\sum_{\substack{\mathbf{V}^{t'k} \\ \setminus \mathbf{v}_\mathbf{x}^{t'k}}} \prod_{\mathbf{z}\neq\mathbf{x}'} \alpha(\mathbf{v}_\mathbf{z}^{t'k})}_{1}, \\
&= \sum_{\mathbf{x}'} f_{xk}(\mathbf{x}', \mathbf{x}; \theta_{xk}) \sum_{\mathbf{v}_\mathbf{x}^{t'k}} f_k(\mathbf{v}_\mathbf{x}^{t'k'}, \mathbf{v}_{\mathbf{x}'}^{t'k}; \theta_k) \alpha(\mathbf{v}_{\mathbf{x}'}^{t'k}). \qquad (29)
\end{aligned}
$$

Finally, the three equations (27), (28), and (29) define a very efficient tightly coupled scale-time forward filter for visual motion estimation. It realizes a complete probabilistic recurrent estimation of a set of flow fields $\mathbf{V}^{t,1:K}$ with different resolutions k swept along the time dimension t. It follows the principle that the longer you observe a scene and the finer the resolution of the data is the more accurate the flow can be estimated.

5. Filter realisations

The pseudo-code, Algorithm 1, shows the very compact form of the derived scale-time filter suitable for an algorithmic implementation. What remains to be done, is the specification of the observation likelihood (12) and the potentials of the transition probability (16) and (19). Without loss of generalization the derivation for the filter assumes discrete state variables which is reflected in using summations $\sum$ for marginalization. If continuous state variables are given the summations $\sum$ simply have to be replaced by integrals $\int$. Everything else keeps being the same. To show the applicability of the framework, we derive two realizations: One for continuous Gaussian and one for discrete grid-based observation likelihoods as well as Mixture of Gaussians and Mixture of Student's-t-distributions transitions. Both realizations have already been published at the International Conference on Machine Learning and Applications Willert et al. (2007; 2008). Here, we summarize the essentials of the modelling in relation to the general filter framework. For optical flow estimation results, discussions on the parameters, and benchmark tests we refer to the published material.

5.1 The Gaussian realisation

We define the observation likelihood and the transitions in such a way that we are left with a purely Gaussian belief representation. This results in a filter similar to an extended Kalman Filter only propagating means and covariances along scale and time.

5.1.1 Gaussian observation likelihood

We follow a similar argumentation as Simoncelli et al. (1991) to obtain the $\ell(\mathbf{v}_{\mathbf{x}}^{tk})$-factors (12) of the observation likelihood. However, our likelihood results from a generative model assuming that a scalar field patch of temporal derivatives $\mathbf{I}_{t,\mathbf{x}}^{tk} \in \mathbb{R}^{X^k \times 1}$ centered around $\mathbf{x}$ is generated by the velocity $\mathbf{v}_{\mathbf{x}}^{tk} \in \mathbb{R}^{2\times 1}$ at position $\mathbf{x}$ and the gradient field patch $(\nabla \mathbf{I}_{\mathbf{x}}^{tk})^T \in \mathbb{R}^{X^k \times 2}$ centered around the same position $\mathbf{x}$.

While introducing this model based on *patches* around position $\mathbf{x}$ instead of only the *pixel* at position $\mathbf{x}$ itself we imply that the optical flow is locally constant in a sense similar to the Lucas-Kanade constraint Lukas & Kanade (1981). Additionally, we assume i.i.d. additive Gaussian noise s_t, $\mathbf{S}_v$ on the temporal derivatives and the flow field, respectively.

$$\ell(\mathbf{v}_{\mathbf{x}}^{tk}) = \mathcal{N}(-\mathbf{I}_{t,\mathbf{x}}^{tk} | (\nabla \mathbf{I}_{\mathbf{x}}^{tk})^T \mathbf{v}_{\mathbf{x}}^{tk}, \mathbf{\Sigma}_{\ell,\mathbf{x}}^{tk}) \,, \tag{30}$$

$$\mathbf{\Sigma}_{\ell,\mathbf{x}}^{tk} = \begin{pmatrix} \ddots & \dots & \mathbf{0} \\ \vdots & \sigma_{\ell,\mathbf{x}\mathbf{x}'}^{tk} & \vdots \\ \mathbf{0} & \dots & \ddots \end{pmatrix} , \tag{31}$$

$$\sigma_{\ell,\mathbf{x}\mathbf{x}'}^{tk} = \frac{(\nabla I_{\mathbf{x}'}^{tk})^T \mathbf{S}_v \nabla I_{\mathbf{x}'}^{tk} + s_t}{f_\ell(\mathbf{x}', \mathbf{x}, t, k)} \,. \tag{32}$$

In notation (30), the patches can be regarded as vectors and the covariance matrix $\mathbf{\Sigma}_{\ell,\mathbf{x}}^{tk}$ is a diagonal with entries $\sigma_{\ell,\mathbf{x}\mathbf{x}'}^{tk}$ that depend on the position $\mathbf{x}'$ relative to the center $\mathbf{x}$, the time t, the scale k, the flow field covariance $\mathbf{S}_v$ and the variance on the temporal derivatives s_t. Here, f_ℓ takes into account the spatial uncertainty of the velocity measurement and can implement any kind of spatial weighting, such as a binomial blurring filter proposed in Simoncelli (1999) or an

anisotropic and inhomogenous Gaussian weighting $f_\ell = \mathcal{N}(\mathbf{x}'|\mathbf{x},\mathbf{\Sigma}^{tk}_{l,\mathbf{x}})$ which is investigated in Willert et al. (2008).
In contrast to Simoncelli (1999), we introduced time t as an additional dimension and derived a more compact notation by putting the spatial weighted averaging directly into the likelihood formulation defining multivariate Gaussian distributions for vectors that describe image patches centered around image locations. Allowing for uncertainties $\mathbf{\Sigma}^{tk}_{\ell,\mathbf{x}}$ that are adaptive in location $\mathbf{x}$, scale k and time t we are able to tune the local motion measurements dynamically e.g. dependent on the underlying structure of the intensity patterns.

5.1.2 Mixture of Gaussians transition

For the temporal contraint (14) we now chose a Gaussian

$$\mathbf{v}^{t'k'}_{\mathbf{x}} \sim \mathcal{N}(\mathbf{v}^{t'k'}_{\mathbf{x}}|\mathbf{v}^{tk'}_{\mathbf{x}'},\sigma_t)\ , \tag{33}$$

which says that the change in time of the flow field is white with undirectional transition noise between $\mathbf{V}^{tk'}$ and $\mathbf{V}^{t'k'}$. For the spatial interaction (15) an inhomogeneous anisotropic Gaussian is assumed

$$\mathbf{x}' \sim \mathcal{N}(\mathbf{x}'|\mathbf{x} - \mathbf{v}^{t'k'}_{\mathbf{x}},\mathbf{\Sigma}^{tk}_{t,\mathbf{x}})\ . \tag{34}$$

to be able to steer the orientation and to adapt the strength of the uncertainty in spatial identification $\mathbf{\Sigma}^{tk}_{t,\mathbf{x}}$ between corresponding positions in time. Combining both factors (33) and (34) and integrating $\mathbf{x}'$ we get a *Mixture of Gaussians* (MoG) as the first pairwise potential (16)

$$\phi_t(\mathbf{v}^{t'k'}_{\mathbf{x}},\mathbf{V}^{tk'}) = \sum_{\mathbf{x}'}\mathcal{N}(\mathbf{x}'|\mathbf{x} - \mathbf{v}^{t'k'}_{\mathbf{x}},\mathbf{\Sigma}^{tk}_{t,\mathbf{x}})\mathcal{N}(\mathbf{v}^{t'k'}_{\mathbf{x}}|\mathbf{v}^{tk'}_{\mathbf{x}'},\sigma_t)\ , \tag{35}$$

with the Gaussian spatial coherence constraint being the mixing coefficients. Equivalent to (33) for the scale transition factor (19) we chose a Gaussian

$$\mathbf{v}^{t'k'}_{\mathbf{x}} \sim \mathcal{N}(\mathbf{v}^{t'k'}_{\mathbf{x}}|\mathbf{v}^{t'k}_{\mathbf{x}''},\sigma_k)\ , \tag{36}$$

assuming white transition noise σ_k. The influence of neighboring velocity states from coarser scale is also modelled as an adaptive Gaussian kernel similar to (34)

$$\mathbf{x}'' \sim \mathcal{N}(\mathbf{x}''|\mathbf{x},\mathbf{\Sigma}^{tk}_{k,\mathbf{x}})\,. \tag{37}$$

Again, combining both factors (36) and (37) and integrating $\mathbf{x}''$ we get a MoG as the second pairwise potential

$$\phi_k(\mathbf{v}^{t'k'}_{\mathbf{x}},\mathbf{V}^{t'k}) = \sum_{\mathbf{x}''}\mathcal{N}(\mathbf{x}''|\mathbf{x},\mathbf{\Sigma}^{tk}_{k,\mathbf{x}})\mathcal{N}(\mathbf{v}^{t'k'}_{\mathbf{x}}|\mathbf{v}^{t'k}_{\mathbf{x}''},\sigma_k)\,, \tag{38}$$

that imposes a spatial smoothness constraint on the flow field via adaptive spatial weighting of motion estimations from coarser scale. The combination of both potentials (16) and (19) results in the complete conditional flow field transition probability as given in (13).

5.1.3 Approximate inference

To arrive at a Gaussian belief we introduce a last approximative restriction. We want every factor of the posterior probability (27) to be Gaussian distributed

$$\alpha(\mathbf{v}_x^{tk}) \propto m_{Y\to v}(\mathbf{v}_{\mathbf{x}}^{tk})\; m_{t\to t'}(\mathbf{v}_{\mathbf{x}}^{tk})\; m_{k\to k'}(\mathbf{v}_{\mathbf{x}}^{tk}) :\approx \mathcal{N}(\mathbf{v}_{\mathbf{x}}^{tk}|\boldsymbol{\mu}_{\mathbf{x}}^{tk},\boldsymbol{\Sigma}_{\mathbf{x}}^{tk})\;. \tag{39}$$

We fulfill this constraint by making all single messages Gaussian distributed. This already holds for the observation likelihood $m_{Y\to v}(\mathbf{v}_{\mathbf{x}}^{tk})$. A more accurate technique (following assumed density filtering) would be to first compute the new belief α exactly as a MoG and then collapse it to a single Gaussian. However, this would mean extra costs. Here, we do not investigate the tradeoff between computational cost and accuracy for different collapsing methods. Inserting Gaussian distributed beliefs α into the propagation equations (28, 29) leads to two different MoGs for the resulting messages

$$m_{t\to t'}(\mathbf{v}_{\mathbf{x}}^{t'k'}) = \sum_{\mathbf{x}'} \hat{p}_{\mathbf{x}'}^{t'k'} \mathcal{N}(\mathbf{v}_{\mathbf{x}}^{t'k'}|\hat{\boldsymbol{\mu}}_{\mathbf{x}'}^{t'k'},\hat{\boldsymbol{\Sigma}}_{\mathbf{x}'}^{t'k'}) \approx \mathcal{N}(\mathbf{v}_{\mathbf{x}}^{t'k'}|\boldsymbol{\omega}_{\mathbf{x}}^{t'k'},\boldsymbol{\Omega}_{\mathbf{x}}^{t'k'})\;, \tag{40}$$

with

$$\begin{aligned}
\hat{p}_{\mathbf{x}'}^{t'k'} &= \mathcal{N}(\mathbf{x}-\mathbf{x}'|\boldsymbol{\mu}_{\mathbf{x}'}^{tk'},\check{\boldsymbol{\Sigma}}_{\mathbf{x}'}^{tk'})\;, && (41)\\
\hat{\boldsymbol{\mu}}_{\mathbf{x}'}^{t'k'} &= (\sigma_t+\boldsymbol{\Sigma}_{\mathbf{x}'}^{tk'})\check{\Lambda}_{\mathbf{x}'}^{tk'}(\mathbf{x}-\mathbf{x}') + \boldsymbol{\Sigma}_{t,\mathbf{x}}^{tk}\check{\Lambda}_{\mathbf{x}'}^{tk'}\boldsymbol{\mu}_{\mathbf{x}'}^{tk'}\;, && (42)\\
\hat{\boldsymbol{\Sigma}}_{\mathbf{x}'}^{t'k'} &= \boldsymbol{\Sigma}_{t,\mathbf{x}}^{tk}\check{\Lambda}_{\mathbf{x}'}^{tk'}(\sigma_t+\boldsymbol{\Sigma}_{\mathbf{x}'}^{tk'})\;, && (43)\\
\check{\boldsymbol{\Sigma}}_{\mathbf{x}'}^{tk'} &= \left[\check{\Lambda}_{\mathbf{x}'}^{tk'}\right]^{-1} = \sigma_t + \boldsymbol{\Sigma}_{t,\mathbf{x}}^{tk} + \boldsymbol{\Sigma}_{\mathbf{x}'}^{tk'}\;,
\end{aligned}$$

and

$$m_{k\to k'}(\mathbf{v}_{\mathbf{x}}^{t'k'}) = \sum_{\mathbf{x}''} \overline{p}_{\mathbf{x}''}^{t'k'} \mathcal{N}(\mathbf{v}_{\mathbf{x}}^{t'k'}|\boldsymbol{\mu}_{\mathbf{x}''}^{t'k},\overline{\boldsymbol{\Sigma}}_{\mathbf{x}''}^{t'k'}) \approx \mathcal{N}(\mathbf{v}_{\mathbf{x}}^{t'k'}|\boldsymbol{\pi}_{\mathbf{x}}^{t'k'},\boldsymbol{\Pi}_{\mathbf{x}}^{t'k'})\;, \tag{44}$$

with

$$\overline{p}_{\mathbf{x}''}^{t'k'} = \mathcal{N}(\mathbf{x}''|\mathbf{x},\boldsymbol{\Sigma}_{k,\mathbf{x}}^{tk})\;,\; \overline{\boldsymbol{\Sigma}}_{\mathbf{x}''}^{t'k'} = \sigma_k + \boldsymbol{\Sigma}_{\mathbf{x}''}^{t'k}\;. \tag{45}$$

In order to satisfy the Gaussian constraint formulated in (39) the MoG's are collapsed into single Gaussians (40, 44) again. This is derived by minimizing the Kullback-Leibler Divergence between the given MoG's and the assumed Gaussians for the means $\boldsymbol{\omega}_{\mathbf{x}}^{tk},\boldsymbol{\pi}_{\mathbf{x}}^{tk}$ and the covariances $\boldsymbol{\Omega}_{\mathbf{x}}^{tk},\boldsymbol{\Pi}_{\mathbf{x}}^{tk}$ which results in closed-form solutions for these parameters. The final *predictive belief* $\alpha(\mathbf{v}_x^{tk})$ follows from the product of these Gaussians

$$\begin{aligned}
\alpha(\mathbf{v}_x^{tk}) &= \ell(\mathbf{v}_x^{tk})\,\mathcal{N}(\mathbf{v}_{\mathbf{x}}^{tk}|\tilde{\boldsymbol{\mu}}_{\mathbf{x}}^{tk},\tilde{\boldsymbol{\Sigma}}_{\mathbf{x}}^{tk})\;, && (46)\\
\tilde{\boldsymbol{\Sigma}}_{\mathbf{x}}^{tk} &= \boldsymbol{\Pi}_{\mathbf{x}}^{tk}\left[\boldsymbol{\Pi}_{\mathbf{x}}^{tk}+\boldsymbol{\Omega}_{\mathbf{x}}^{tk}\right]^{-1}\boldsymbol{\Omega}_{\mathbf{x}}^{tk}\;, && (47)\\
\tilde{\boldsymbol{\mu}}_{\mathbf{x}}^{tk} &= \boldsymbol{\Omega}_{\mathbf{x}}^{tk}\left[\boldsymbol{\Pi}_{\mathbf{x}}^{tk}+\boldsymbol{\Omega}_{\mathbf{x}}^{tk}\right]^{-1}\boldsymbol{\pi}_{\mathbf{x}}^{tk} + \boldsymbol{\Pi}_{\mathbf{x}}^{tk}\left[\boldsymbol{\Pi}_{\mathbf{x}}^{tk}+\boldsymbol{\Omega}_{\mathbf{x}}^{tk}\right]^{-1}\boldsymbol{\omega}_{\mathbf{x}}^{tk}\;. && (48)
\end{aligned}$$

By applying the approximation steps (39, 40) and (44) we guarantee the posterior (27) to be Gaussian which allows for Kalman-filter like update equations since the observation is defined to factorize into Gaussian factors (30). The final recurrent motion estimation is given by

$$\begin{aligned}
\alpha(\mathbf{v}^{tk}_{\mathbf{x}}) &= \mathcal{N}(\mathbf{v}^{tk}_{\mathbf{x}}|\boldsymbol{\mu}^{tk}_{\mathbf{x}},\boldsymbol{\Sigma}^{tk}_{\mathbf{x}}) && (49)\\
&= \mathcal{N}(-\mathbf{I}^{tk}_{t,\mathbf{x}}|(\nabla\mathbf{I}^{tk}_{\mathbf{x}})^T\mathbf{v}^{tk}_{\mathbf{x}},\boldsymbol{\Sigma}^{tk}_{\ell,\mathbf{x}})\mathcal{N}(\mathbf{v}^{tk}_{\mathbf{x}}|\tilde{\boldsymbol{\mu}}^{tk}_{\mathbf{x}},\tilde{\boldsymbol{\Sigma}}^{tk}_{\mathbf{x}})\,, && (50)\\
\boldsymbol{\Sigma}^{tk}_{\mathbf{x}} &= \left[\tilde{\Lambda}^{tk}_{\mathbf{x}} + \nabla\mathbf{I}^{tk}_{\mathbf{x}}\Lambda^{tk}_{\ell,\mathbf{x}}(\nabla\mathbf{I}^{tk}_{\mathbf{x}})^T\right]^{-1}\,, && (51)\\
\boldsymbol{\mu}^{tk}_{\mathbf{x}} &= \tilde{\boldsymbol{\mu}}^{tk}_{\mathbf{x}} - \boldsymbol{\Sigma}^{tk}_{\mathbf{x}}\nabla\mathbf{I}^{tk}_{\mathbf{x}}\Lambda^{tk}_{\ell,\mathbf{x}}\tilde{\mathbf{I}}^{tk}_{t,\mathbf{x}}\,. && (52)
\end{aligned}$$

For reasons explained in Simoncelli (1999) the innovations process is approximated as the following

$$\tilde{\mathbf{I}}^{tk}_{t,\mathbf{x}} \approx \partial/\partial t\mathcal{T}\left(\mathbf{I}^{tk}_{\mathbf{x}},\tilde{\boldsymbol{\mu}}^{tk}_{\mathbf{x}}\right)\,, \tag{53}$$

with $\mathcal{T}$ applying a backward warp plus bilinear interpolation on the image $\mathbf{I}^{tk}_{\mathbf{x}}$ using the predicted velocities $\tilde{\boldsymbol{\mu}}^{tk}_{\mathbf{x}}$ from (48). We end up with a Gaussian scale-time filter which is, in comparison to existent filtering approaches Elad & Feuer (1998), Simoncelli (1999), Singh (1991), not a Kalman Filter realization but related to an extended Kalman Filter since the result of the nonlinear transitions is linearized after each message pass with the collpase of each MoG to a single Gaussian.

5.2 The grid-based realisation

A grid-based filter allows only a discrete set of state variables but is otherwise not restricted to any particular form of distribution. Here, we neglect the scale dimension but show how past and future observables can be processed offline via a Two-Filter.

5.2.1 Observation likelihood

Now, we define the observation likelihood $P(\mathbf{Y}^t|\mathbf{V}^t)$ by assuming that the likelihood factor $\ell(\mathbf{Y}^t|\mathbf{v}^t_{\mathbf{x}})$ of a local velocity $\mathbf{v}^t_{\mathbf{x}}$ should be related to finding the same or similar image patch centered around $\mathbf{x}$ at time t' that was present at time t but centered around $\mathbf{x} - \mathbf{v}^t_{\mathbf{x}}$. More rigorously, let $\mathcal{S}(\mathbf{x},\boldsymbol{\mu},\boldsymbol{\Sigma},\nu)$ be the Student's t-distribution and $\mathcal{N}(\mathbf{x},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \lim_{\nu\to\infty}\mathcal{S}(\mathbf{x},\boldsymbol{\mu},\boldsymbol{\Sigma},\nu)$ be the normal distribution of a variable $\mathbf{x}$ with mean $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$ and the degrees of freedom ν. In the following the covariance is chosen to be isotropic $\boldsymbol{\Sigma} = \sigma^2\mathbf{E}$ (with identity matrix $\mathbf{E}$). We define

$$\ell(\mathbf{Y}^{t+1}|\mathbf{v}^{t+1}_{\mathbf{x}}) = \sum_{\mathbf{x}'}\mathcal{N}(\mathbf{x}'|\mathbf{x},\varrho_I)\mathcal{S}(I^{t'}_{\mathbf{x}'}|I^{t+1}_{\mathbf{x}'-\mathbf{v}^{t+1}_{\mathbf{x}}},\sigma_I,\nu_I) = \sum_{\mathbf{x}'}\mathcal{N}(\mathbf{x}'|\mathbf{x}-\mathbf{v}^t_{\mathbf{x}},\varrho_I)\mathcal{S}(I^{t'}_{\mathbf{x}'+\mathbf{v}^{t+1}_{\mathbf{x}}}|I^{t+1}_{\mathbf{x}'},\sigma_I,\nu_I)\,. \tag{54}$$

Here, $\mathcal{N}(\mathbf{x}'|\mathbf{x},\varrho_I)$ implements a Gaussian weighting of locality centered around $\mathbf{x}$ for I^{t+1} and around $\mathbf{x} - \mathbf{v}^t_{\mathbf{x}}$ for I^t. The parameter ϱ_I defines the spatial range of this image patch and σ_I the grey value variance. The univariate Student's t-distribution $\mathcal{S}(I^{t+1}_{\mathbf{x}'},I^t_{\mathbf{x}'-\mathbf{v}^t_{\mathbf{x}}},\sigma_I,\nu_I)$ realizes a robust behaviour against large gray-value differences within image patches, which means these gray-values are treated as outliers and are much less significant for the distribution.

5.2.2 Mixture of Student's t transition

Similarly to equation (54), we define the transition probability $P(\mathbf{v}^{t+1}_{\mathbf{x}}|\mathbf{V}^t)$ by assuming that the flow field transforms according to itself like defined in 16 and further specified as

$$P(\mathbf{v}^{t+1}_{\mathbf{x}}|\mathbf{V}^t) \propto \mathcal{N}(\mathbf{x}'|\mathbf{x}-\mathbf{v}^{t+1}_{\mathbf{x}},\varrho_V)\mathcal{S}(\mathbf{v}^{t+1}_{\mathbf{x}}|\mathbf{v}^t_{\mathbf{x}'},\sigma_V,\nu_V)\,. \tag{55}$$

Using a heavy tailed Student's t distribution, we assume robust spatiotemporal coherence because evaluations on first derivative optical flow statistics Roth & Black (2005) and on prior distributions that allow to imitate human speed discrimination tasks Stocker & Simoncelli (2006) provide strong indication that they resemble such heavy tailed Student's t-distributions. The parameter ϱ_V defines the spatial range of a flow-field patch, so we compare velocity vectors within flow-field patches at different times t and $t+1$. We introduced new parameters ϱ_V and σ_V for the uncertainty in spatial identification between two images and the transition noise between $\mathbf{V}^t$ and $\mathbf{v}_\mathbf{x}^{t+1}$, respectively. The robustness against outliers is controlled by ν_V, with smaller/larger ν_V decreasing/increasing the influence of incoherently moving pixels within the observed spatial range ϱ_V. With $\nu_V \to \infty$ the uncertainty for the velocity gets Gaussian distributed and (55) equals the transition probability formulated in Burgi et al. (2000) which expresses the belief that pixels *are, on average, moving along a straight line with constant velocity*. Therefore, the proposed spatiotemporal transition model 55 can be seen as a generalization of the transition model proposed by Burgi et al. (2000).

5.2.3 Two-Filter inference

Like beforehand, for inference we need to propagate beliefs over the flow field $\mathbf{V}^t$. Storing a distribution over a whole flow field $\mathbf{V}^t$ is infeasible if one does not make factorization assumptions. The factored observation likelihoods and transition probabilities we introduced ensure that the forward propagated beliefs will remain factored. However, the standard backward messages do not exactly factor under this model. Hence we follow a two-filter approach Kitagawa (1994) where the "backward filter" is strictly symmetric to the forward filter.

Following the derivation for the temporal belief propagation 28 and specifying the transition probability as in equation 55 the forward filter reads

$$\begin{aligned} \alpha(\mathbf{v}_\mathbf{x}^{t+1}) &\propto \ell(\mathbf{Y}^{t+1}|\mathbf{v}_\mathbf{x}^{t+1})\,\alpha^*(\mathbf{v}_\mathbf{x}^{t+1}) && (56)\\ \alpha^*(\mathbf{v}_\mathbf{x}^{t+1}) &\propto \sum_{\mathbf{x}'}\mathcal{N}(\mathbf{x}'|\mathbf{x}-\mathbf{v}_\mathbf{x}^{t+1},\varrho_V)\sum_{\mathbf{v}_{\mathbf{x}'}^t}\mathcal{S}(\mathbf{v}_\mathbf{x}^{t+1}|\mathbf{v}_{\mathbf{x}'}^t,\sigma_V,\nu_V)\,\alpha(\mathbf{v}_{\mathbf{x}'}^t)\,. && (57)\end{aligned}$$

If we have access to a batch of data (or a recent window of data) we can compute smoothed posteriors as a basis for an EM-algorithm and train the free parameters. In our two-filter approach we derive the backward filter as a mirrored version of the forward filter, but using

$$P(\mathbf{v}_\mathbf{x}^t|\mathbf{V}^{t+1}) \propto \sum_{\mathbf{x}'}\mathcal{N}(\mathbf{x}'|\mathbf{x}+\mathbf{v}_\mathbf{x}^t,\varrho_V)\;\mathcal{S}(\mathbf{v}_\mathbf{x}^t,\mathbf{v}_{\mathbf{x}'}^{t+1}|\sigma_V,\nu_V) \qquad (58)$$

instead of (55). This equation is motivated in exactly the same way as we motivated (55): we assume that $\mathbf{v}_\mathbf{x}^t \sim \mathcal{S}(\mathbf{v}_{\mathbf{x}'}^{t+1},\sigma_V,\nu_V)$ for a corresponding position $\mathbf{x}'$ in the subsequent image, and that $\mathbf{x}' \sim \mathcal{N}(\mathbf{x}-\mathbf{v}_\mathbf{x}^t,\varrho_V)$ is itself defined by $\mathbf{v}_\mathbf{x}^t$. However, note that using this symmetry of argumentation is actually an approximation to our model because applying Bayes rule on (55) would lead to a different, non-factored $P(\mathbf{V}^t|\mathbf{V}^{t+1})$. What we gain by the approximation $P(\mathbf{V}^t|\mathbf{V}^{t+1}) \approx \prod_x P(\mathbf{v}_\mathbf{x}^t|\mathbf{V}^{t+1})$ are factored β's which are feasible to maintain computationally. The backward filter equations read

$$\begin{aligned} \beta^*(\mathbf{v}_\mathbf{x}^t) &\propto \ell(\mathbf{Y}^t|\mathbf{v}_\mathbf{x}^t)\,\beta(\mathbf{v}_\mathbf{x}^t)\,, && (59)\\ \beta(\mathbf{v}_\mathbf{x}^t) &\propto \sum_{\mathbf{x}'}\mathcal{N}(\mathbf{x}'|\mathbf{x}+\mathbf{v}_\mathbf{x}^t,\varrho_v)\sum_{\mathbf{v}_{\mathbf{x}'}^{t+1}}\mathcal{S}(\mathbf{v}_\mathbf{x}^t|\mathbf{v}_{\mathbf{x}'}^{t+1},\sigma_V,\nu_V)\beta^*(\mathbf{v}_{\mathbf{x}'}^{t+1})\,. && (60)\end{aligned}$$

To derive the smoothed posterior we need to combine the forward and backward filters. In the two-filter approach this reads

$$\begin{aligned}\gamma(\mathbf{v}_\mathbf{x}^t) &= P(\mathbf{v}_\mathbf{x}^t|\mathbf{Y}^{1:T}) = \frac{P(\mathbf{Y}^{t+1:T}|\mathbf{v}_\mathbf{x}^t)\,P(\mathbf{v}_\mathbf{x}^t|\mathbf{Y}^{1:t})}{P(\mathbf{Y}^{1:T})} \qquad (61)\\ &= \frac{P(\mathbf{v}_\mathbf{x}^t|\mathbf{Y}^{t+1:T})P(\mathbf{Y}^{t+1:T})P(\mathbf{v}_\mathbf{x}^t|\mathbf{Y}^{1:t})}{P(\mathbf{v}_\mathbf{x}^t)P(\mathbf{Y}^{1:T})}\\ &\propto \alpha(\mathbf{v}_\mathbf{x}^t)\,\beta(\mathbf{v}_\mathbf{x}^t)\,\frac{1}{P(\mathbf{v}_\mathbf{x}^t)}\,, \qquad (62)\end{aligned}$$

with $P(\mathbf{Y}^{t+1:T})$ and $P(\mathbf{Y}^{1:T})$ being constant. If both the forward and backward filters are initialized with $\alpha(\mathbf{v}_\mathbf{x}^0) = \beta(\mathbf{v}_\mathbf{x}^T) = P(\mathbf{v}_\mathbf{x})$ we can identify the unconditioned distribution $P(\mathbf{v}_\mathbf{x}^t)$ with the prior $P(\mathbf{v}_\mathbf{x})$. For details on the standard forward-backward-algorithm we refer to Bishop (2006).

6. Summary

A reliable and robust motion estimate is an important low-level processing unit that has the potential to bootstrap a number of visual perception tasks to be solved by a cognitive vision system. Since the estimation of motion information has to rely on highly uncertain visual information a probabilistic treatment of the problem is proposed. Based on three basic approaches to solve motion ambiguities, the derivation of a probabilistic filter is given that combines all these three approaches into one recurrent framework. The derivation comprises an efficient approximate inference algorithm based on belief propagation applied on a directed graphical model with a graph topology suitable for intertwining belief propagation along two dimensions, scale and time, simultaneously. Introducing some factorisation assumptions and a special class of transition probabilities results in a very compact and computationally efficient algorithm. For this algorithm two implementations are presented. The first one realizes a purely factored Gaussian belief propagation and the second one the propagation of a factored non-parametric discrete distribution. The presented framework provides a flexible basis for the realization of user specific motion estimation algorithms with the focus on online applications. It also serves as an exploration platform to investigate in adaptation mechanisms and online learning strategies for example to improve the optical flow estimation accuracy or increase the robustness for highly dynamic scenes.

7. References

Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion, *IJCV: International Journal of Computer Vision* **2**(3): 283–310.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M. & Szeliski, R. (2007). A database and evaluation methodology for optical flow, *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV)*.

Barron, J., Fleet, D. & Beauchemin, S. (1994). Performance of optical flow techniques, *IJCV: International Journal of Computer Vision* **12**(1): 43–77.

Beauchemin, S. & Barron, J. (1995). The computation of optical flow, *ACM Computing Surveys* **27**(3): 433–467.

Bergen, J. R., Anandan, P., Hanna, K. J. & Hingorani, R. (1992). Hierarchical model-based motion estimation, *Proceedings of the Second European Conference on Computer Vision*

(ECCV), Vol. 588 of *Lecture Notes In Computer Science*, Springer-Verlag, London, UK, pp. 237–252.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, Springer Science+Business Media.

Black, M. (1994). Recursive non-linear estimation of discontinuous flow fields, *ECCV*, pp. 138–145.

Brox, T., Bruhn, A., Papenberg, N. & Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping, *in* T. Pajdla & J. Matas (eds), *Proceedings of the 8th European Conference on Computer Vision (ECCV)*, Vol. 3024 of *Lecture Notes in Computer Science*, Springer-Verlag, Prague, Czech Republic, pp. 25–36.

Burgi, P., A.L.Yuille & Grzywacz, N. (2000). Probabilistic motion estimation based on temporal coherence, *Neural Computation* **12**: 1839–1867.

Elad, M. & Feuer, A. (1998). Recursive optical flow estimation-adaptive filtering approach, *Journal of Visual Communication and image representation* **9**: 119–138.

Horn, B. K. P. & Schunk, B. G. (1981). Determining optic flow, *Artificial Intelligence* **17**: 185–204.

Isard, M. & Blake, A. (1998). Condensation - conditional density propagation for visual tracking, *IJCV: International Journal of Computer Vision* **29**: 5–28.

Jähne, B. (1997). *Digitale Bildverarbeitung*, Springer-Verlag Berlin Heidelberg.

J.J.Gibson (1950). The perception of the visual world, *Technical report*, Houghton Mifflin Company, Boston, MA.

Kay, S. (1993). *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, Englewood Cliffs, NJ.

Kitagawa, G. (1994). The two-filter formula for smoothing and an implementation of the gaussian-sum smoother, *Annals Institute of Statistical Mathematics* **46**(4): 605–623.

Lukas, B. D. & Kanade, T. (1981). An iterative image-registration technique with an application to stereo vision, *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, Vancouver, Canada, pp. 674–679.

Memin, E. & Perez, P. (1998). A multigrid approach for hierarchical motion estimation, *Proceedings of the Sixth International Conference on Computer Vision (ICCV)*, Bombay, India, pp. 933–938.

Rosenberg, Y. & Werman, M. (1997). A general filter for measurements with any probability distribution, *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, San Juan, Puerto Rico, pp. 106–111.

Roth, S. & Black, M. (2005). On the spatial statistics of optical flow, *ICCV*, pp. 42–49.

Simoncelli, E. (1993). *Distributed Representation and Analysis of Visual Motion*, PhD thesis, MIT Department of Electrical Engineering and Computer Science.

Simoncelli, E. (1999). *Handbook of Computer Vision and Applications*, Academic Press, chapter Bayesian Multi-Scale Differential Optical Flow, pp. 397–421.

Simoncelli, E. (2003). Local analysis of visual motion, *The Visual Neuroscience*, MIT Press.

Simoncelli, E., Adelson, E. & Heeger, D. (1991). Probability distributions of optical flow, *CVPR*, pp. 310–315.

Singh, A. (1990). An estimation-theoretic framework for image-flow computation, *Proceedings of the Third International Conference on Computer Vision*, Osaka, Japan, pp. 168–177.

Singh, A. (1991). Incremental estimation of image flow using a kalman filter, *IEEE Workshop on Visual Motion*, pp. 36–43.

Stocker, A. A. & Simoncelli, E. P. (2006). Noise characteristics and prior expectations in human visual speed perception, *Nature Neuroscience* **9**(4): 578–585.

Weber, J. & Malik, J. (1995). Robust computation of optical flow in a multi-scale differential framework, *IJCV: International Journal of Computer Vision* **14(1)**: 5–19.

Weiss, Y. (1993). *Bayesian Motion Estimation and Segmentation*, PhD thesis, MIT Department of Brain and Cognitive Science.

Weiss, Y. & Fleet, D. (2002). Velocity likelihoods in biological and machine vision, *Probabilistic Models of the Brain: Perception and Neural Function*, MIT Press, pp. 77–96.

Willert, V., Toussaint, M., Eggert, J. & Körner, E. (2007). Uncertainty optimization for robust dynamic optical flow estimation, *ICMLA*, pp. 450–457.

Willert, V., Toussaint, M., Eggert, J. & Körner, E. (2008). Probabilistic exploitation of the lucas and kanade smoothness constraint, *ICMLA*, pp. 259–266.

Wu, Q. X. (1995). A correlation-relaxation-labeling framework for computing optical flow - template matching from a new perspective, *IEEE Trans. PAMI* **17(3)**: 843–853.

Yedidia, J., Freeman, W. & Weiss, Y. (2003). *Exploring Artificial Intelligence in the New Millennium*, Morgan Kaufmann, chapter Understanding Belief Propagation and Its Generalizations, pp. 239–236.

Zelek, J. (2002). Bayesian real-time optical flow, *Proceedings of the 15th International Conference on Vision Interface (VI)*, Calgary, Canada, pp. 310–315.

Zetzsche, C. & Krieger, G. (2001). Nonlinear mechanisms and higher-order statistics in biological vision and electronic image processing: Review and perspectives, *Journal of Electronic Imaging* **10(1)**: 56–99.

Concept Mining and Inner Relationship Discovery from Text

Jiayu Zhou, Shi Wang
School of Computing, Informatics and Decision Systems Engineering, Arizona State University
USA
Institute of Computing Technology, Chinese Academy of Sciences
China

1. Introduction

From the cognitive point of view, knowing concepts is a fundamental ability when human being understands the world. Most concepts can be lexicalized via words in a natural language and are called Lexical Concepts. Currently, there is much interest in knowledge acquisition from text automatically and in which concept extraction, verification, and relationship discovery are the crucial parts (Cao et al., 2002). There are a large range of other applications which can also be benefit from concept acquisition including information retrieval, text classification, and Web searching, etc. (Ramirez & Mattmann, 2004; Zhang et al., 2004; Acquemin & Bourigault, 2000)

Most related efforts in concept mining are centralized in term recognition. The common used approaches are mainly based on linguistic rules (Chen et al., 2003), statistics (Zheng & Lu, 2005; Agirre et al., 2004) or a combination of both (Du et al., 2005; Velardi et al., 2001). In our research, we realize that concepts are not just terms. Terms are domain-specific while concepts are general-purpose. Furthermore, terms are just restricted to several kinds of concepts such as named entities. So even we can benefit a lot from term recognition we cannot use it to learn concepts directly.

Other relevant works in concept mining are focused on concepts extraction from documents. Gelfand has developed a method based on the Semantic Relation Graph to extract concepts from a whole document (Gelfand et al., 1998). Nakata has described a method to index important concepts described in a collection of documents belonging to a group for sharing them (Nakata et al., 1998). A major difference between their works and ours is that we want to learn huge amount of concepts from a large-scale raw corpus efficiently rather than from one or several documents. So the analysis of documents will lead to a very higher time complexity and does not work for our purpose.

There are many types relationships between lexical concepts such as antonymy, meronomy and hyponymy, among which the study of hyponymy relationship has attracted many effort of research because of its wide use. There are three mainstream approaches—the *Symbolic* approach, the *Statistical* approach and the *Hierarchical* approach—to discovery general hyponymy relations automatically or semi automatically (Du & Li, 2006). The Symbolic approach, depending on lexicon-syntactic patterns, is currently the most popular technique (Hearst, 1992; Liu et al., 2005; Liu et al., 2006; Ando et al., 2003). Hearst (Hearst, 1992) was one of the early researchers to extract hyponymy relations from Grolier's Encyclopedia by matching 4 given lexicon-syntactic patterns, and more importantly, she discussed about extracting lexicon-syntactic patterns by existing hyponymy relations. Liu (Liu et al., 2005; Liu et al., 2006) used the "isa" pattern to extract Chinese hyponymy relations from unstructured Web corpus, and have been proven to have a promising performance. Zhang (Zhang et al., 2007) proposed a method to automatically extract hyponymy from Chinese domain-specific free text by three symbolic learning methods. The statistical approach usually adopts clustering and associative rules. Zelenko et al. (Zelenko et al., 2003) introduced an application of kernel methods to extract two certain kinds of hyponymy relations with promising results, combining Support Vector Machine and Voted Perception learning algorithms. The hierarchical approach is trying to build a hierarchical structure of hyponymy relations. Caraballo (Caraballo, 1999) built a hypernymy hierarchy of nouns via a bottom-up hierarchical clustering technique, which was akin to manually constructed hierarchy in WordNet.

In this paper, we use both linguistic rules and statistical features to learn lexical concepts from raw texts. Firstly, we extract a mass of concept candidates from text using lexico-patterns, and confirm a part of them to be concepts according to their matched patterns. For the other candidates we induce an *Inner-Constructive Model* (CICM) of words which reveal the rules when several words construct concepts through four aspects: (1) parts of speech, (2) syllables, (3) senses, and (4) attributes. Once the large scale concept set is built based on the CICM model, we developed a framework to discover inner relationships within concepts. A lexical hyponym acquisition is proposed based on this framework.

2. Extracting Concepts from Text using Lexico-Patterns

In this research, our goal is to extract huge amount of domain-independent concept candidates. A possible solution is to process the text by Chinese NLU systems firstly and then identity some certain components of a sentence to be concepts. But this method is limited dut to the poor performance of the existing Chinese NLU systems, which still against many challenge at present for Chinese (Zhang & Hao, 2005). So we choose another solution based on lexico-patterns.

2.1 The Lexico-Patterns of Lexical Concepts

Enlightened Hearst's work (Hearst, 1992), we adopt lexico-patterns to learning lexical concepts from texts. But first design a lot of lexico-patterns patterns manually, some of which are shown in Table

<table>
<tr><th>ID</th><th>Lexical Patterns</th></tr>
<tr><td>1</td><td>&lt;?C1&gt;&lt;是&gt;&lt;一 | &gt;&lt;个 | 种 | &gt;&lt;?C2&gt;</td></tr>
<tr><td>2</td><td>&lt;?C1&gt;&lt;、><?C2>&lt;或者 | 或是 | 以及 | 或 | 等 | 及 | 和 | 与&gt;&lt;其他 | 其它 | 其余&gt;</td></tr>
<tr><td>3</td><td>&lt;?C1&gt;&lt;、><?C2>&lt;等等 | 等&gt;&lt;?C3&gt;</td></tr>
<tr><td>4</td><td>&lt;?C1&gt;&lt;如 | 象 | 像&gt;&lt;?C2&gt;&lt;或者 | 或是 | 或 | 及 | 和 | 与 | 、&gt;&lt;?C3&gt;</td></tr>
<tr><td>5</td><td>&lt;?C1&gt;&lt;、><?C2&gt;&lt;是 | 为&gt;&lt;?C3&gt;</td></tr>
<tr><td>6</td><td>&lt;?C1&gt;&lt;、><?C2&gt;&lt;各 | 每 | 之 | 这&gt;&lt;种 | 类 | 些 | 样 | 流&gt;&lt;?C3&gt;</td></tr>
<tr><td>7</td><td>&lt;?C1&gt;&lt;或者 | 或是 | 或 | 等 | 及 | 和 | 与&gt;&lt;其他 | 其它 | 其余&gt;&lt;?C2&gt;</td></tr>
<tr><td>8</td><td>&lt;?C1&gt;&lt;或者 | 或是 | 或 | 及 | 和 | 与&gt;&lt;?C2&gt;&lt;等等 | 等&gt;&lt;?C3&gt;</td></tr>
<tr><td>9</td><td>&lt;?C1&gt;&lt;中 | 里 | 内 | &gt;&lt;含 | 含有 | 包含 | 包括&gt;&lt;?C2&gt;</td></tr>
<tr><td>10</td><td>&lt;?C1&gt;由<?C2&gt;<组成 | 构成&gt;</td></tr>
</table>

Table 1. The Lexico-Patterns for Extracting Concepts from Text

Here is an example to show how to extract concepts from text using lexico-patterns:

Example 1. Lexico-Pattern_No_1{
Pattern: < ?C1> <是><一 | > < 个 | 种> < ?C2>
Restrict Rules:
not_contain(<?C2>,<! 标点>)^lengh_greater_than(<?C1>,1)^lengh_greater_than(<?C2>,1)^
lengh_less_than(<?C1>,80)^lengh_less_than(<?C2>,70)^not_end_with(<?C1>,<这 | 那>)^
not_end_with(<?C2>,<的|而已|例子|罢了>)^
not_begin_with(<?C2>,<这 | 的 | 它 | 他 | 我 | 那 | 你 | 但>)^
not_contain(<?C2>,<这些 | 那些 | 他们 | 她们 | 你们 | 我们 | 他 | 它 | 她 | 你 | 谁>)}

Sample sentences and the concepts extracted:

(1) 地球是一个行星，地球会爆炸吗？(The earth is a planet, will it blast?) →<?C1>=地球(The earch); <?C2>=行星(a planet)
(2) 很久很久以前地球是一个充满生机的星球.(Long long ago the Earth is a planet full of vitality.) →<?C1>=很久很久以前地球(Long long ago the Earth); <?C2>=充满生机的星球(a planet full of vitaligy)

How to devise good patterns to get as much concepts as possible? We summarized the following criteria through experiments:
(1) *High accuracy criterion*. Concepts distributing in sentences meet linguistics rules, so each pattern should reflect at least one of these rules properly. We believe that we should know linguistics well firstly if we want create to good patterns.
(2) *High coverage criterion*. We want to get as much concepts as possible. Classifying all

concepts into three groups by their characteristics, (i.e. concepts which describe physical objects, concepts and the concepts which describe time) is a good methodology for designing good patterns to get more concepts.

2.2 Confirming Concepts using Lexico-Patterns

Obviously, not all the chunks we got in section 2.1 are concepts, such as <?C1>=很久很久以前地球(Long long ago the Earth) in Example 1 above. In order to identify concepts from the candidates, we introduce a hypothesis, called Hypothesis 1.

Hypothesis 1. A chunk ck extracted using lexico-patterns in section 2.1 is a concept if (1) {ck} has been matched by sufficient lexico-patterns, or (2) {ck} has been matched sufficient times.

To testify our hypothesis, we randomly draw 10,000 concept candidates from all the chunks and verify them manually. The association between the possibility of a chunk to be a concept and its matched patterns is shown as Fig. 1:

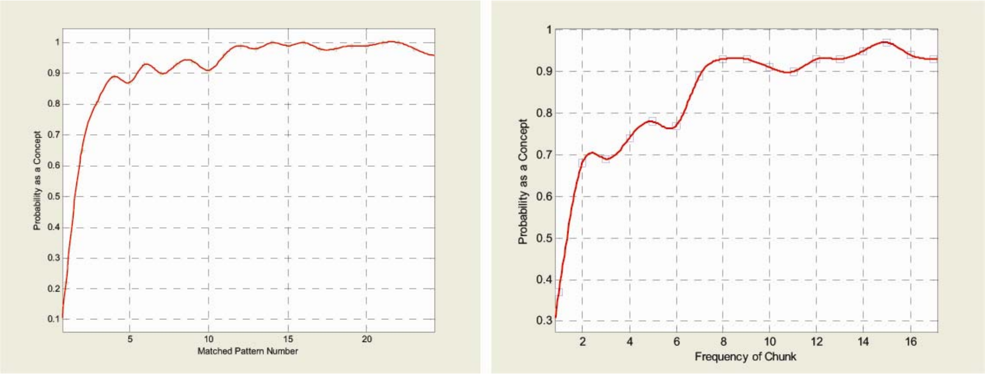


Fig. 1. Association between the lexico-patterns number / the times matched by all the patterns of chunks and their possibility of being concepts}

The left chart indicates our hypothesis that the chunks which matched more patterns are more likely to be concepts and the right chart shows that the frequency of the chunk does work well to tell concepts from candidate chunks too. In our experiments, we take the number of patterns matchings to be 5 and threshold of matching frequency as 14, and single out about 1.22% concepts from all the candidate chunks with a precision rate of 98.5%. While we are satisfied with the accuracy, the recall rate is rather low. So in the next step, we develop CICMs to recognize more concepts from chunks.

3. Learning Concepts using CICM

The CICM is founded on an instinctive hypothesis:

Hypothesis 2. Most lexical concepts obey certain inner constructive rules.

That means, when some words form a concept, each word must play a certain role and has certain features. We develop the hypothesis enlightened mainly from the knowledge of linguistics (Lu et al., 1992) and the cognitive process of human beings creating lexical concepts (Laurance & Margolis, 1999). Some examples will be given to illuminate the Hypothesis 2 after present the definition of CICM.

3.1 Definition of CICM

According to Hypothesis 2, we can tell whether an unknown chunk is a concept or not by checking whether each word in it whether obeys the CICM. The problems are how to materialize these rules and how to get them. The POS models can reveal these rules using the parts of speech of words but is not precise enough and has many defections (Yu, 2006). To get better performance we probe into the structure of concepts more deeply and find that besides POS, we must ensure each word's more definite role through at least other three aspects.

Definition 1. The word model W=< PS, SY, SE, AT> of a word w is a 4-tuple where (1) PS is all the parts of speech of w; (2) SY is the number of w's syllable; (3) SE is the senses of w in HowNet; and (4) AT is the attributes of w.

The *word models* are integrated information entities to model words. The reason of choosing these four elements listed above will be clarified when we construct CICMs.

Definition 2. Given a concept cpt=$w_1 \dots w_{i-1}\ w_i\ w_{i+1} \dots w_n$ with n words, the C-Vector of the word w_i towards cpt is a n-tuple:

$$\text{C-Vector}(w_i) = < i, W_1, \dots, W_{i-1}, W_{i+1}, \dots, W_n > \quad (1)$$

The *C-Vector* of a word stands for one constructive rule when it forms concepts by linking other words and *i* is its position in the concept. A word can have same *C-Vectors* towards many different concepts. The *C-Vector* is the basis of CICM.

Definition 3. The Concept Inner-Constructive Models (CICMs) of a word w is a bag of C-Vectors, in which each C-Vector is produced by a set of concepts contain w.

Essentially, CICMs of words represent the constructive rules when they construct concepts. In the four elements of word models, PS and SY embody the syntactical information which have significant roles when conforming concepts in Chinese (Lu et al., 1992) and are universal for all types of words. SE and AT reveal the semantic information of words and are also indispensably. HowNet is an elaborate semantic lexicon attracted many attentions in many related works (Dong & Dong, 2006). But there are still some words which are missing in it so we need to introduce attributes as a supplement. Attributes can tell the semantic differences at the quantative level or qualitative level between concepts. Tian has developed a practicable approach to acquire attributes from large-scale corpora (Tian, 2007).

ID	C-Vectors	Sample Concepts
1	< 1, W(管理) >	生产 管理
2	< 1, W(许可证) >	生产 许可证
3	< 1,W(实习),W(报告)>	生产 实习 报告
4	< 2,W(食品)>	食品 生产
5	< 2,W(分布式)>	分布式 生产
6	< 2,W(国民),W(总值)>	国民 生产 总值
7	< 2,W(新疆),W(建设),W(兵团)>	新疆 生产 建设 兵团
8	< 3,W(广东省),W(春耕)>	广东省 春耕 生产
9	< 3,W(国家),W(安全),W(监督),W(管理局)>	国家 安全 生产 监督 管理局
…	…	…

Table 2. CICM of “生产”

Note that we omit the details of each word vector for simplicity. Taking ``国民 生产 总值'' for example, the full *C-Vector* is:

<   2,

<{n},2,{属性值,归属,国,人,国家},{有组成,有数量}>,

<{n},2,{数量,多少,实体},{有值域,是抽象概念}>>

3.2 Learning CICMs

Using CICMs as the inner constructive rules of concepts, our next problem is how to get these models. We use the confirmed concepts obtained in section 2.2 as a training set and learn CICMs hidden in them automatically. It is an instance learning process and the following procedure is implemented for this task:

Algorithm

CICMs Instance Learning Algorithm:

(1) Initializing the resources including (1.1) A words dictionary in which each one has fully parts of speech; (1.2) The HowNet dictionary; and (1.3) An attributes base of words (Tian, 2007).

(2) Constructing a model set MSet to accommodate all the words' models which is empty initially.

(3) For each concept cpt in the training set, segment it and create each word's C-Vector(w_i). Subsequently, if C-Vector(w_i) ∈ MSet(w_i), then just accumulate the frequency; otherwise add C-Vector(w_i) to MSet(w_i).

(4) Removing the C-Vectors which have low frequency for each word's MSet.

Based on experiments, we choose 10% as the threshold of the number of the concepts containing the word in the training set. We exclude the vectors which have low frequency, that is, if a C-Vector for a word is supported by just a few concepts, we look at it as an exception.

4. Clustering Words for More Efficient Analogy

Essentially, CICMs are models of instance analogy. We want to learn new concepts by ``recalling'' the old ones just as human beings. For example, we can build CICMs for the word ``*生产(produce, production)*'' like Table 2 and then identify that ``*药品 生产 (pharmaceutical production)*'' is also a concept, because the latter has the same constructive rule as ``*食品 生产(food production)*''.

But unluckily, even we know ``*药品 生产*'' is a concept, our system still cannot tell whether `` *药品 制造(pharmaceutical manufacture)*'' is also a concept for there are no CICMs for the word ``*制造*''. The reason for this is that the system still cannot make use of word similarity. Therefore, we need to cluster words based on the similarity of CICMs and then learn more new concepts.

4.1 Similarity Measurement of Words

The similarity measurement of CICMs is the basis of clustering words in our task. Our measurement is founded on the intuitive distribute hypothesis that:

Hypothesis 3 In concepts, similar words have similar CICMs.

According to Hypothesis 3, the similarity of two words w_1, w_2 is defined as:

$$\mathrm{sim}(w_1, w_2)=\mathrm{sim}(\mathrm{CICM}(w_1), \mathrm{CICM}(w_2)) \tag{2}$$

The commonly used similarity measure for two sets includes *minimum distance*, *maximum distance*, and *average distance*. Considering that there are still some noises in our training set which would result in some wrong C-Vectors in CICMs, we choose the average distance for it is more stable for noisy data, that is:

$$\begin{aligned}\mathrm{sim}(w1, w2) &= \frac{1}{|CICM(w1)|}\sum_{vec_i \in CICM(w1)} sim(vec_i, CICM(w2)) \\ &= \frac{1}{|CICM(w1)||CICM(w2)|}\sum_{vec_i \in CICM(w1)}\sum_{vec_i \in CICM(w1)} sim(vec_i, vec_j)\end{aligned} \tag{3}$$

Now the problem is how to calculate the similarity of two C-Vectors of two words now. For two C-Vectors:

$$\text{C-Vector}_i=<i, W_1,\ldots,W_n>, \text{C-Vector}_j=<j,W_1,\ldots,W_m> \tag{4}$$

We standardize them to an \emph{N-Vector} that is:

$$\text{C-Vector}_i=<W^i_{-N},\ldots,W^i_N>, \text{C-Vector}_j=<W^j_{-N},\ldots,W^j_N> \tag{5}$$

and $W_k=\varnothing$ if there is no word model in position *k* for both of them. We adopt the cosine similarity when compare two vectors, that is:

$$sim(vec_i, vec_j) = \cos(\overrightarrow{vec_i}, \overrightarrow{vec_j}) = \frac{\overrightarrow{vec_i}\cdot\overrightarrow{vec_j}}{|\overrightarrow{vec_i}|\times|\overrightarrow{vec_j}|} \tag{6}$$

4.2 Clustering Words based on the Density

Among all the clustering methods using density functions has prominent advantages--anti-noisiness and the capability of finding groups with different Inspired by DENCLUE (Hinneburg & Keim, 1998), we define a influence function of a word *w_0* over another word *w*:

$$f_B^w = f_B(w_0, w) \tag{7}$$

which is a function proportionately to the similarity of *w_0* and *w*, and reveals the influence degree *w_0* over *w*. Commonly used influence functions include *Square Wave Function* and *Gauss Function*. The former is suitable for the data which dissimilar distinctly while the later is more suitable for reflect the smooth influence of *w_0*. Because a word is related with many other words in different degrees but no simply 1 or 0 in corpus, it is more reasonable to choose Gauss Influence Function:

$$f_{Gauss}^{w}(w_0) = e^{\frac{-(1-\sin(w0,w1))^2}{2\sigma^2}} \tag{8}$$

We call Equation (8) the *Gauss Mutual Influence* of *w*, w_0 for $f^w_{Gauss}(w_0)=f^{w0}_{Gauss}(w)$. It makes each word linked with many other words to some extent. According to it, we can cluster words into groups. Before giving the definition of a word group, we develop some definitions first for further discussing:

Definition 4. Given a parameter ξ, ξ_region(w_0)={w | fGuess(w,w_0)> ξ } is called ξ_region of w_0. Given a parameter *MinPts*, w_0 is called a *CoreWord* if | ξ_region(w_0) | >MinPts. The minimal ξ which makes w_0 to be a CoreWord is called the CoreDistance of w_0 and be marked as ξ*.

Definition 5. We call w_0 is direct reachable to w' if w_0 is a CoreDistance and w'$\in\xi$_region(w_0)$ and marked as $d_{reachable(w0,w')}$. For a set of words w_0, w_1, …,w_n=w', if $d_{reachable(wi,wi+1)}$ for all w_i, $1 \le i<n$$, then w_0 is reachable to w', that is, reachable(w_0,w').

Based on the definitions above, a word group can be seen as the maximal words set based on the reachable property. The corresponding clustering algorithm is given below:

```
(1)Taking $\xi = \xi$* and for all the words w perform the following operation:
 $\xi_{cur}= \xi$*;$cw_{cur}$={w};
 while($\xi_{cur} < 1$){
   $cw_{pre}=cw_{cur}$;
```

```
  if(| $\xi_{cur}$_region($w_0$) | > MinPts){
    Build a word group $cv_{cur}$ which contains all the words in $\xi_{cur}$_region(w)
    and takes $w$ as the CoreWord of it.
    if($\frac{|cv_{cur} - cv_{pre}|}{|cv_{cur}|} < \alpha$ ){break;}
  }else{
    break;
  }\\if
  $\xi_{pre}$= $\xi_{cur}$;  $\xi_{cur}$+= $\Delta$ ;
 }//while
 }cw=$cw_{pre}$;
(2)For each pair of CoreWords $w_i$, $w_j$
if(d_reachable(w_i,w_j)
  Merge $cw_i$,$cw_j$ into a new group $cw_{i+j}$ which has two CoreWords $cw_i$ and $cw_j$
(3)Repeate (2) until no new groups are generated.
```

Many groups with different density will be generated in (2) for we set value for ξ not a single number but a large range of field. The groups with high density will be created firstly and be covered by the dilute groups. We escape choosing the parameter of ξ by doing this.

4.3 Identifying Concepts using CICMs

Having the learned CICMs and word cluster, identifying method of new concepts is straightforward. Given a chunk, we just create its local L-Vector and judge whether it satisfies one of its or its similar words' C-Vector we have learned.

Definition 6. For a chunk c_k=w_0… w_n, the Local C-Vector for a word w_i in it :
L_Vector(w_i, c_k)=<i,W_0,…,W_{i-1}, W_{i+1},…, W_n>.
Theorem 1. For a chunk c_k=w_0 … w_n , for each word w_i in it, there is L_Vector(w_i, c_k) $\in$ CICM(gw_i), then c_k is a concept, where gw_i is the similar word group of w_i.

5. Inner Relationship Discovery

The concept extractor introduced in last chapters makes a large-scale concept set available to be used for discovering inner relationships of the concept. In this study, we have found a special kind of Chinese hyponymy relationship, called lexical hyponymy, which is of great importance in ontology learning. To the best of our knowledge, no existing method can extract these hyponym relations. In this chapter, we will show a semi-automatic lexical hyponymy acquisition approach within a large-scale concept set, integrating symbolic, statistical and hierarchal techniques.

In a large-scale concept set C, if a subset S={<cpt1>,<cpt2>,…,<cptn>} exists, where

<cpt1>=<pref1><suf>,
<cpt2>=<pref2><suf>,
…
<cptn>=<prefn><suf>.

The <suf> here denotes a common suffix of all concepts. We may think the <suf> *could* be a hypernymy concept and following relations *may* exist:
HISA(<cpt1>, <suf>), HISA(<cpt2>,<suf>),…, HISA(<cptn>,<suf>)
For instance, given S={炭疽活菌苗，冻干鼠疫活菌苗，结核活菌苗，自身菌苗，外毒素菌苗}, we can segment the concepts as follows,

<自身菌苗>=<自身><菌苗>,
<外毒素菌苗>=<外毒素><菌苗>,
<结核活菌苗>=<结核活><菌苗>,
<炭疽活菌苗>=<炭疽活><菌苗>,
<冻干鼠疫活菌苗>=<冻干鼠疫活><菌苗>,

where the corresponding hypernymy concept suffix <suf> is <菌苗> and all HISA relations come into existence. However, if we consider the suffix chunk <苗> to be <suf> instead of <菌苗> (i.e. we segment the concept <外毒素菌苗>:=<外毒素菌><苗>), all HISA relations do not exist. Moreover, the suffix <苗> can not even be considered as a concept. We notice that a subset S′={结核活菌苗，炭疽活菌苗，冻干鼠疫活菌苗} of S contains a longer common hyponymy <活菌苗>, lexical hyponymy relations HISA(结核活菌苗，活菌苗), HISA(炭疽活菌苗，活菌苗) and HISA(冻干鼠疫活菌苗，活菌苗).

We will investigate into such common suffix in a concept set and mine lexical hyponymy taking advantage of the common suffix features. There is a limitation in this approach: the size of the concept set should be *very large* in order to find such common chunks. In an extreme case, we can extract nothing if there is only one concept in the concept set, even if the only concept in the set contains rich lexical hyponymy relations. However, there is no definition how large can be thought to be very large and we will analysis this factor in the experiment section.

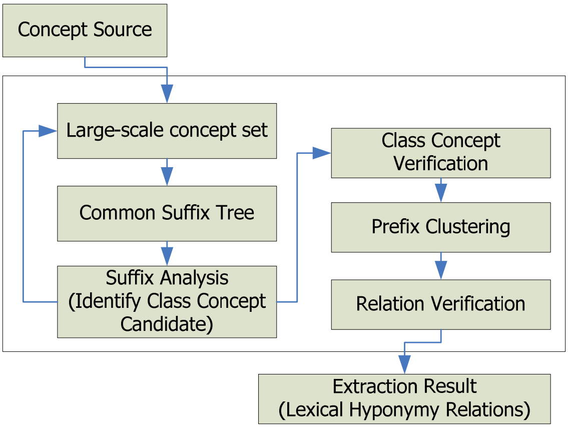

Concept Source
Large-scale concept set
Common Suffix Tree
Suffix Analysis (Identify Class Concept Candidate)
Class Concept Verification
Prefix Clustering
Relation Verification
Extraction Result (Lexical Hyponymy Relations)

Fig. 2. Lexical hyponymy acquisition framework

Figure 2 describes our framework of lexical hyponymy acquisition. We use a Google-based statistical acquisition model [16] to extract concepts from web corpus, which results in a large-scale concept set and then clustered them into a common suffix tree according to suffixes of concepts. The suffix analysis module uses a set of statistical-based rules to analyze suffix nodes. Class concept candidates, which are concepts, are identified by our Google-base verification module and used to enlarge the original concept set. A class concept verification process was taken to verify class concept candidates. Human judgment-based relation verification is taken after a prefix clustering process dedicating to reduce the verification cost is done. Finally we got extracted hyponymy relations from the common suffix tree with a hierarchical structure.

6. Common Suffix Tree Clustering

To find and analyze the common suffix, we propose a data structure called *common suffix tree (CST)*, inspired by suffix tree clustering (Cusfield, 1997).

Definition 7. A common suffix tree containing *m* concepts is a tree with exactly m leaves. Each inner node, other than leaf, has more than two children, and contains a single Chinese gram. Each leaf indicates a concept with a *longest shared suffix* that equals the string leading from the leaf to root. Along with the path, the string from each inner node to root is a shared suffix of the concept indicated by leaves it can reach.

With CST, not only are we able to find what is the longest shared suffix, we can also find which concepts share a certain common suffix. Following CST clustering algorithm will help us construct a CST in liner time complexity:

```
CST Clustering Algorithm:
Use the suffix-based clustering, and compute big 1-gram concept clusters.
Until(convergence) {
    From each n-gram cluster, iterate the algorithm to get finer, hierarchy n+1
    gram clusters.
}
```

The convergence condition of algorithm above is when the all clusters leave one leaf. For instance, in a given concept-set S={北京第六中学，南京第十六中学，天津第二十六中学，经济学，生物学，好学，同学，木鱼，黄花鱼，烧黄花鱼，鲤鱼}, The CST algorithm can be described as following steps:

1) Using the suffix-based clustering, we get big 1-gram clusters ({*} represents the least common suffix):

[北京第六中,南京第十六中,天津第二十六中,经济,生物,好,同]{学}

[木, 黄花, 烧黄花, 鲤]{鱼},

2) From each 1-gram cluster, we iterate the algorithm to get finer, hierarchical clusters until convergence:

[[[北京第, [南京第,天津第二]{十}]{六}]{中},经济,生物,好,同]{学}

[木, [#, 烧]{黄花}, 鲤]{鱼},

where # represents an empty entry.
Figure 3 visualized the CST structure of {学}-cluster. The rest parts of our framework are built on the computing and analysis on suffixes of CST.

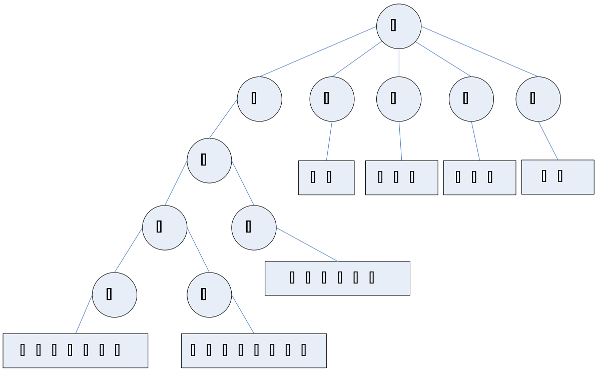


Fig. 3. Common Suffix Tree of {学}-cluster

7. Suffix Analysis

Given the "学" cluster in the example above, the suffix collection *S*={*第十六中学, 十六中学, 六中学, 中学*} may all hypernymy concepts we interested in, without any other information supporting (1-gram suffix causes great ambiguous, therefore we leave it alone in our system).
Some suffix concepts may be extracted by some Chinese word segment systems [20], however, there is no word segment system adopted in our system, because the segment system performs poor in a large scale general-purposed concept set, where many suffixes cannot be correctly segmented and thus lowered the performance of the entire system.
However, some useful statistic features can be obtained in a concept-set to identify class concepts. For a suffix chunk <*ck*> in concept-set, we may have patterns such as *CNT[*<*ck*><*>*]*, *CNT[*<*><*ck*>*]*, *CNT[*<*><*ck*><*>*]* and etc., where *CNT[*<*pattern*>*]* means the frequency of <*pattern*> in concept set. A list of examples of such patterns was listed in Table 3.

Pattern	Example
(1) *ISCpt[*<*ck*>*]*	<大学>∈S
(2) *CNT[*<*ck*><*>*]*	<大学>学生服务部, <大学>校区, …
(3) *CNT[*<*><*ck*>*]*	理工<大学>, 科技<大学>, …
(4) *CNT[*<*><*ck*><*>*]*	北京<大学>学生会, 中国<大学>评估组, …

Table 3. Statistical patterns and examples

Pattern (1) is not a real statistic. The pattern, once appears in the given concept set, prove that indicated suffix <*ck*> is a class concept candidate. If the concept set is *large enough* (i.e. for any <*cpt*>, always exists <*cpt*> ∈*S*), this single rule can be used to identify all class concept

candidates. Actually, our concept set can never achieve that large.
The emergence of pattern (2) and (3) is a strong indication of class concept, which usually can be some components of other words. Class concept <*大学(university)*> can be used as *limitation* of other concept, such as <*大学校区(university campus)*>, which indicates a special kind of <*校区(campus)*>. Experiment in following content shows that once the pattern (2) or (3) appears, the empirical probability of <*ck*> to be a concept is very high.

The information embedded in the pattern (4) is richer. We rewrite the *CNT[<*><ck>]* as f_{suf}(<*ck*>), called *suffix frequency*. The *i*th suffix of concept <*cpt*>= <x_m>…<x_1> (i.e. <x_i>…<x_1>), where <x_i> a is single gram, is denoted as *Suf(<cpt>, i)* and *m* is the length of <*cpt*>. The *suffix probability* S_{suf}*(<cpt>,n)* is defined as:

$$\begin{aligned} S_{suf}(<cpt>,n) &= p_{suf}(<x_m>\ldots<x_1>,\ <x_{n+1}><x_n>\ldots<x_1>) \\ &= f_{suf}(Suf(<cpt>,n+1)) \,/\, f_{suf}(Suf(<cpt>,n)) \end{aligned} \tag{9}$$

where $n \leqq$ *Length(<cpt>)-1*. $p_{suf}(<ck_1>, <ck_2>)$ is the joint probability of chunk <ck_1> and <ck_2>. We define that single-gram concepts have no suffix probability.

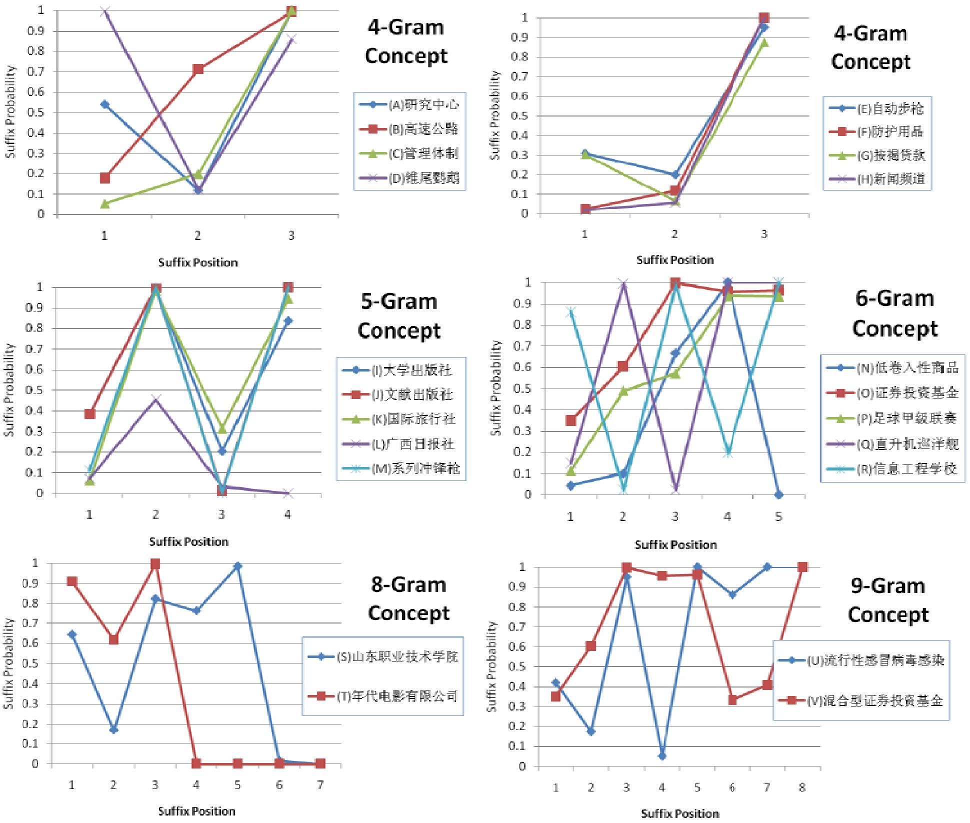


Fig. 4. Case study: suffix probability of 4, 5, 6, 8, 9-gram concepts respectively.

Figure 4 shows some cases of suffix probability whose numbers of grams composed are ranging from 4 to 9. Such cases illustrate how suffix probability changes with varying number of grams.

Figure 4 (V) shows the change of suffix frequency of concept <*cpt*>:= *"混合型证券投资基金"* in a concept set with a size of 800,000. Figure 4 (U) shows the situation when <*cpt*>*:="流行性感冒病毒感染"*. For instance,

$$S(\text{"混合型证券投资基金"}, 2)=P(\text{"基金"}, \text{"资基金"})$$
$$=F(\text{"资基金"})/F(\text{"基金"})= 300/498 = 0.60241$$

From the case (S) we observe that *S(<cpt>, 3)=0.99667* is the maximum among *S(<cpt>, 2)*, *S(<cpt>, 3)* and *S(<cpt>, 4)*. At the same time *Suf(<cpt>, 4)* (i.e. 投资基金) is a class concept. Same situation could be found in maximum point *S(<cpt>, 5)* and *S(<cpt>, 8)*, while *Suf(<cpt>, 6)* and *Suf(<cpt>, 8)* are both class concept. In another case when <cpt>="流行性感冒病毒感染", we find the same phenomenon that class concept happened to appear in inflexions, which makes us believe it to be a useful rule. The rule is proved to be very effective in later experiment and is defined as follows:

Definition 8. *(Suffix Probability Inflexion Rule)* In a large-scale concept set, whenever the suffix probability *S(<cpt>,n)* encounters an inflexion, the suffix *Suf(<cpt>,n+1)* $=<w_{n+1}><w_n>\dots<w_1>$ is considered to be a class concept candidate, which is called *Inflexion Rule*.

The suffix probability inflexion rule is exported from empirical study, and the hidden theoretical support of this rule is based on *mutual information*. The higher the *S(<cpt>,n)*, then the suffix $<w_n>\dots<w_1>$ and $<w_{n+1}><w_n>\dots<w_1>$ has higher mutual information, which may lead to a close correlation, the sudden reduce of mutual information means differentiation in linguistic usage.

Based on the discussions above, we summarize three *Suffix Concept Identification (SCI)* Rules:

1. *Pattern ISCpt[<wx>] appears, then <wx> must be a concept.*
2. *Pattern CNT[<wx><*>] or CNT[<*><wx><*>] appears, then <wx> can be a concept.*
3. *Suffix Probability Inflexion Rule.*

The experimental baseline comparisons among three rules are listed in Table 4. We use SCI rules in an 800,000 concept set and 300 test cases and manually extract all the class concept candidates in test cases, denoted by *cm*. Then we use SCI rules to extract class concepts, denoted by *ca*. We adopt following evaluation measurements in baseline experiment:

$$Precision = | ca \cap cm | / | ca |$$
$$Recall = | ca \cap cm | / | cm |$$

The average value and standard deviation of precisions and recalls are computed in 5 baseline scheme. Rules based on (1), (2) or the combinations of which have a low recall although with a high precision, as a result of the data sparsity. However, rule (3) holds a high precision and at the same time has a promising recall once combined with the other two rules.

	Precision		Recall	
	Average	Std. Dev	Average	Std. Dev
Rule(1)	100%	0	-	n/a
Rule(2)	95.753%	0.4603	-	n/a
Rule(3)	98.641%	0.1960	65.125%	2.393
Rule(1,2)	96.561%	0.5133	-	n/a
Rule(1,2,3)	98.145%	0.5029	66.469%	2.792

Table 4. SCI Rules Baseline Comparison (- mean the value is lower than 5%).

8. Class Concept Verification

In previous section we mentioned that not every concept could be a class concept. In this section, we proposed a lexicon-syntactic approach to verify class concept by scoring concepts via Googling web corpus.

Through our investigation, class concepts primarily appear in three kinds of lexicon-syntactic patterns which have different semantic meanings: Class I patterns appear when people are trying to give examples. Class II patterns are used when people construct question sentences. Class III patterns are, on the other hand, commonly used when we give definitions. The generic type of Class II is <Which><*>, where <Which> is some of the interrogatives. The generic type of Class II is <是> <Unit><*>, and here <Unit> is some of the unit quantifiers. Therefore, the pattern II and III includes a number of patterns. All three types of pattern with examples are summarized as shown in Table 5.

Pattern Type	Pattern Examples	Examples
Class I <*Such as*><*>	<ClassCpt>例如	一些*水果例如*香蕉，它如何繁衍后代
	等<ClassCpt>	76%预期深圳*等城市*的房价将下跌
Class II <*Which*><*>	什么<ClassCpt>	福威镖局在福州府的*什么大街*
	哪些<ClassCpt>	中国*哪些城市*适宜工作?
	那种<ClassCpt>	青苹果和红苹果*哪种苹果*有营养
Class III <*是*><*Unit*><*>	是一个<ClassCpt>	法国夏特瑞城*是一个小镇*
	是一种<ClassCpt>	宪政*是一种文化*
	是一类<ClassCpt>	他和你*是一类人*

Table 5. Patterns and examples in three classes

Definition 9. Google provides statistical information in web corpus, probability framework based on which has been built by (Zhou et al., 2007; Cilibrasi & Vitanyi, 2007). Given a lexical chunk <ck>, the frequency of this term is defined as number of pages containing such term, denoted by f(<ck>).

Definition 10. For a concept <cpt>, the pattern frequency is defined as f(Pattern(<cpt>)), where Pattern(<cpt>) is applying the concept to a certain pattern. Pattern association is defined as the pattern frequency of the concept dividing its frequency, denoted by p(Pattern(<cpt>)).

$$p(Pattern(< cpt >)) = f(Pattern(< cpt >)) / f(< cpt >) \tag{10}$$

To verify class concepts, pattern associations can be used as attributes to train a classifier by machine learning algorithms. However, according to the linguistic property of the three classes, the pattern associations of a certain concept are likely to associate well with only one pattern in each class. Therefore we only use the patterns that can have the maximum pattern association in each class. We use the liner combination to sum pattern associations of all three classes into a scoring function, which is proved to be more effective than adopting three separate attributes.

Three classes of patterns are assigned with different class weights w_I, w_{II}, w_{III}, which can be used to adjust score according to liner analysis methods. Besides, we take the frequency of concept as a coefficient of the score, which indicates that a concept with a higher frequency is more likely to be a class concept. To sum all effects above, the expression of scoring a concept <cpt> is:

$$Score(< cpt >) = Log(f(< cpt >)) \times \sum_{i \in (I, II, III)} (w_i \times \underset{j \in Class_i}{Max}(p(Pattern_j(< cpt >)))) \tag{11}$$

To obtain a score threshold identifying class concept, we firstly annotate a training set of 3000 concepts, including 1500 class concept and 1500 non-class concept. We then use Google to retrieve pattern associations of training set. So the pattern associations are calculated into a score. And we use a linear analysis method to adjust the class weighs that can maximize the scoring function, and finally we get a score threshold. Concepts that exceed the given threshold are classified as class concept and vice versa. In our experiment, the class concept classifier we built is proved to achieve a remarkable high accuracy at 95.52%.

9. Prefix Clustering

Due to the property of lexical hyponymy relations, they hardly appear in other sources such as text corpus and web corpus, which makes human judgment a compulsory step in the relation verification process. In a large-scale concept set, the number of lexical hyponymy relations is huge, and thus it becomes a misery if we need to manually verify each relation.

In a concept sub-set S={<京津塘高速公路>, <长株潭高速公路>, <京石高速公路>, <京承高速公路>, <信息高速公路>} with the suffix Suf={<*>, 4} and Suf={<*>, 2}, where <*> denotes the wildcard of concepts, but the hyponymy relation within term <信息高速公路(Information High-Way)> is different from others. Since the concept is a kind of metaphor, there is not a real lexical hyponymy relation. If we can cluster the relations into meaningful groups, such as, metaphor group and non-metaphor group, it is possible for us to verify parts of the relation group instead of all relations.

We notice that a prefix <pref> of a concept <cpt>=<pref><suf> is typically a term that forms parts of other concepts in our concept set. Given a <pref>, H(<pref>) denotes all chunks that appears before <pref> in other concepts and T(<pref>) denotes all chunks that appears after <pref> in other concepts. The two statistical information, that provided by concept set context, can be used to define the similarity of two prefixes.

Definition 11. Prefix Similarity is a quantity for measuring the similarity of two prefixes within a concept-set context. It is the average of Crossover Coefficients of Head Similarity and Tail Similarity.

$$Sim(<x>,<y>) = (\frac{|H(<x>) \cap H(<y>)|}{\min(|H(<x>)|,|H(<y>)|)} + \frac{|T(<x>) \cap T(<y>)|}{\min(|T(<x>)|,|T(<y>)|)}) \tag{12}$$

K-cluster technique, which is the simplest unsupervised learning algorithm, enables us to cluster data according to a given number of clusters k (MacQueen, 1967). With the ease to control cluster number, we can then flexibly choose a specific grain to cluster our relations. We perform a k-cluster algorithm on concept set using prefix similarity. In the case above, there are 1210 concepts containing “信息” in our 800,000 concept set. Other prefix terms rarely appear and share some terms such as <*><收费站>. Given k=2, the prefix <信息> will be placed in a separate group through clustering, while the rest four prefixes are grouped into one cluster. Hence, we only need to judge two hyponymy relations respectively from each cluster. From the empirical study, the best k-value is a median proportion of the size of the target concept sub-set.

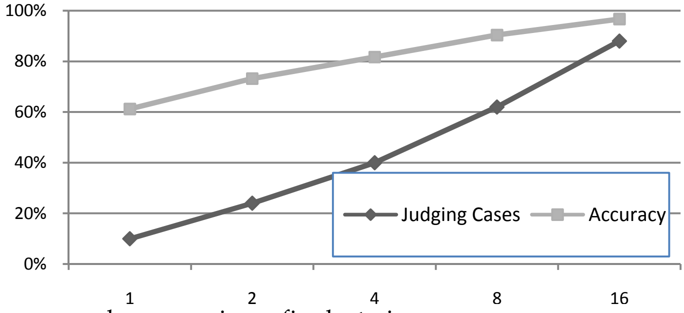


Fig. 5. Judging cases and accuracy in prefix clustering

This step is optional comparing to other modules employed in our framework, and sometimes it may lower the precision of the system. Figure 5 describes our judging cases and accuracy in a 1000-sized sub tree of a CST built by an 800,000 concept set. When setting the K-value to be 8, we will have an accuracy of 90.4% by judging 62% relation cases. Remarkably, not only does the percentage of judging cases depend on K-value, it also relates to the structure of targeting CST. However, prefix clustering will significantly improve the efficiency of human judgment during verification phase.

10. Discovering Hierarchical Lexical Hyponymy

Given a concept set *C*, we use the CST clustering technique to build a CST. Then we compute the statistics of patterns described in Sect. 6 and store them in each CST node. We apply the

SCI rules to extract class concept candidates ***T′***, and add them to ***C***, enlarging our original concept set. We verify the unverified candidates in ***T′*** with the Google-base verification described in Sect. 7, and get a class concept set ***T***. In lexical hyponymy relation candidate set ***H′***, we remove all the relations that have hypernymy concepts in ***T***-***T′***.

Lexical hyponymy relations are generated as follows: For a given concept node <*cpt*>, set *{<s-cpt*$_1$*>… <s-cpt*$_n$*>}* is used to denote all the verified class concept nodes it goes through in CST, and we have *HISA(<s-cpt*$_i$*>,<s-cpt*$_j$*>)* ($i<j$). Put all generated relations to ***H′***. As the original concept set changed, we update statistical information of each node, and keep performing steps above until the status of each node remains unchanged. Finally we cluster the prefix according to Sect.7 and judge one relation candidate in each cluster in ***H′***, resulting our final *hierarchical* lexical hyponymy relation set ***H***. The pseudo code of acquiring process is given in Figure 6.

```
Acquiring a large-scale concept set C.
Constructing CST using CST clustering.
While(Convergence){
    Compute statistical information of inner nodes.
For each concept node <cpt> in CST {
Apply SCI rules.
            Get all class concept candidates T′
C←T′
           T← Verify unverified candidates in T′
            Remove hyponymy of invalid candidates
H′←All relation candidates in T
}
}
Perform Prefix Clustering in H′
Judging Relations in H′, resulting H
```

Fig. 6. Acquiring hierarchical lexical hyponymy relations

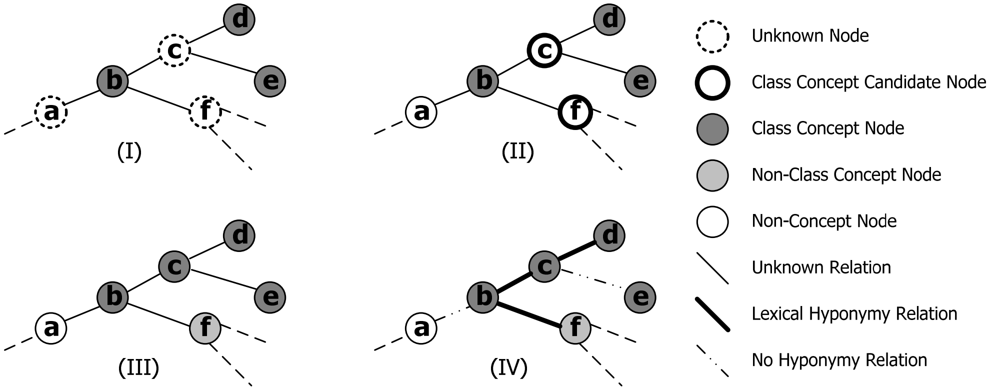


Fig. 7. An example of hierarchical acquisition process

To better illustrate this acquisition process, an example is given in Figure 7. Nodes {*a*, *b*, *c*, *d*, *e*, *f*} are *suffix chunk nodes* in a Common Suffix Tree. A suffix chunk node represents a lexical chunk of string starting from the corresponding CST node leading to the root. In (I), we have already known that *b*, *d*, *e* are class concept nodes and the rest are unknown nodes. Through suffix analysis, *a* is proved to be a non-concept and *b*, *c* are identified to be class concept candidates, as shown in (II). The candidates are then verified by the class concept classifier. In (III), *c* is classified as class concept and *d* is classified as non-class concept. Hyponymy relation candidates are *HISA(d, c)*, *HISA(e, c)*, *HISA(d, b)*, *HISA(e, b)*, *HISA(f, b)*, where *HISA(d, b)* and *HISA(e, b)* are derived from *transitivity* of hyponymy relation. *HISA(e, c)* is judged as a non-hyponymy relation, leading that *HISA(e, b)* to be removed, as shown in (IV).

11. Experiment

11.1 Concept Extraction

The concept extraction part of our system is called *Concept Extractor* (CptEx) and uses the following formulae to evaluate its performance:

$$p = \frac{\| m_a \cap m_m \|}{\| m_a \|}, r = \frac{\| m_a \cap m_m \|}{\| m_m \|}, F - Measure = \frac{2 \times p \times r}{p + r} \qquad (8)$$

where m_a are the concepts CptEx extracts and m_{m} are the ones built manually. To calculate the performance, we selected 1000 chunks from the raw corpus and label the concepts in them manually. We compare the results based on CICMs with those based the Syntax Models and the POS Models as shown in Table 6:

Measurement	Syntax Models	POS Models	CICM
p	98.5%	86.1%	89.1%
r	1.2%	87.8%	84.2%
F-measure	2.3%	86.9%	86.6%

Table 6. Performance of CptEx

Having adopted CICMs to distinguish concepts from the chunks extracted by lexico-patterns, the precision rate drops down to 89.1% while the recall rate flies to 84.2%. The precision rate reduces because there are still some improper CICMs which will confirm fake concepts.
Compared with POS Models, CICMs has a higher accuracy rate because we consider more factors to clarify the inner constructive rules rather than using part of speech only. On the other hand, our stricter models result in a lower recall rate.

11.2 Relation Mining

Our lexical hyponymy relation discovery is being evaluated through 5 concept sets of the size of 10000, 50000, 100000, 400000, and 800000, respectively. To compare their performances under different settings, we use the resulting lexical hyponymy relations acquired, followed by a human judgment with a k=10 prefix clustering. The same evaluation system as the last section is used to evaluate the performance of our system:

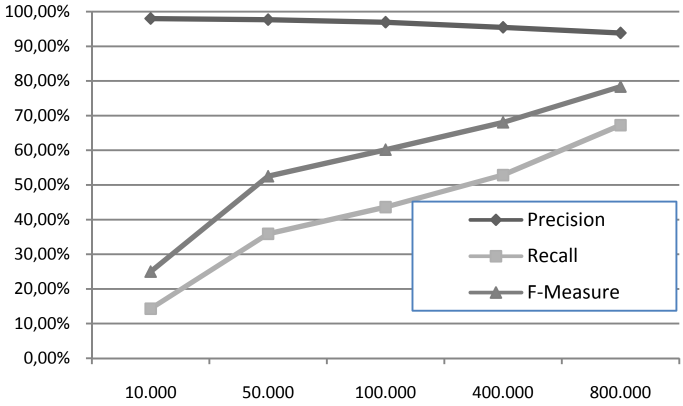


Fig. 8. System performance with different concept set size

From the acquisition result shown in Fig.8, we can discover that F-measure incrementally increases coincides the larger concept-set size, from 24.93 in 10000-sized concept set, climbing to 78.34 in 800000 one. Precision lower slightly and recall increase significantly with a larger concept set. As the size of concept set enlarges, more statistical information emerges, and at the same time more suffix concepts are extracted as class concepts, some of which form lexical hyponymy relations, causing a higher recall, while some other relations are invalid, leading to a lower precision. Under the concept-set with a size of 800000, the precision is 93.8% and recall reaches to 67.24%. The recall can be even higher when given a larger concept set.

In our concept set, we discover noise due to exocentric compounds, in which the suffix concepts are not hypernymy concepts. So far, no effort has been done to verify Chinese exocentric structures and the difficulty of linguistic usage makes it hard to analyze semantic relation within Chinese lexical concepts, which inevitably lower the precision of our framework.

Single-gram hypernymy concepts, such as ‘计’, are likely to cause ambiguity. In our concept set, we find a large number of concepts ended with suffixes like {“硬度计”,“光度计”, “温度计”, “速度计”, “长度计”, “高度计”}. The mutual information between “度” and “计” is very high, leading the algorithm adopting SPI rule to wrongly mark the chunk “度计”, rather than “计”, as a class concept candidate. This problem might be solved if we could avoid the information sparsity by further enlarging the concept set.

The precision of class concept verification module is an important factor to the performance of whole system. We can further obtain a larger feature space and enhance the performance by employing advanced learning techniques such as SVM and Naïve Bayes Network.

Final precision of the framework is affected by our prefix clustering judgment, however, when the concept set becomes larger and thus more relations are extracted, it is inevitable for us to adopt that judgment.

12. Conclusion

We have described a new approach for automatic acquisition of concepts from text based on Syntax Models and CICMs of concepts. This method extracted a large number of candidate concepts using lexico-patterns firstly, and then learned CICMs to identify more concepts accordingly. Experiments have shown that our approach is efficient and effective. We test the method in a 160G free text corpus, and the outcome indicates the utility of our method.

To discover the inner relationships of the concept set, we propose a novel approach to discover lexical hyponymy relations in a large-scale concept set and make the acquisition of lexical hyponymy relations possible. In this method we cluster a concept set into a common suffix tree firstly, and then use the proposed statistical suffix identification rules to extract class concept candidates in the inner nodes of the common suffix tree. We then design a Google-base symbolic class concept verifier. Finally we extract Lexical hyponymy relations and judge them after the prefix clustering process. Experimental result has shown that our approach is efficient and can correctly acquire most lexical hyponymy relations in a large-scale concept set.

In the concept extraction part there are still some more works be done to get better performance for there are some improper CICMs. We plan to validate concepts in an open corpus such as in the World Wide Web in the future. In the relation discovery future work will be concentrated on the extraction of single-gram suffixes, which covers a large part of lexical hyponymy relations. On the other hand, through inner cross verification within a concept set, an approach that automatically verifies hyponymy relation is coming soon.

13. Reference

Acquemin, C. & Bourigault, D. (2000). Term Extraction and Automatic Indexing. Oxford University Press, Oxford(2000)

Agirre, E.; Ansa, O.; Hovy, E. & Martinez, D. (2004). Enriching very large ontologies using the WWW. In: Proc. of the ECAI 2004 Workshop on Ontology Learning.

Ando, M.; Sekine, S. & Ishizaki, S.(2003). “Automatic Extraction of Hyponyms from Newspaper Using Lexicon-syntactic Patterns”, In IPSJ SIG Technical Report 2003-NL-157, pp.77-83, 2003

Cao C., Feng, Q. and et al. (2002). Progress in the Development of National Knowledge Infrastructure. Journal of Computer Science & Technology, Vol.17, No.5, 1¨C16, May, 2002.

Caraballo, S. (1999). “Automatic Construction of a Hypernym-babeled Noun Hierarchy from Text”, In proceedings of 37th Annual Meeting of the Association for Computational Linguistics, Maryland, pp120-126, Jun 1999.

Chen, W.; Zhu, J. & Yao, T. (2003). Automatic learning field words by bootstrapping. In: Proc. of the JSCL. Beijing: Tsinghua University Press, 2003. pp. 67--72

Cilibrasi, R.L.& Vitanyi, P.(2007). The Google Similarity Distance. Knowledge and Data Engineering. IEEE Transactions 19(3), pp370–383, 2007

Dong, Z. & Dong, Q. (2006). HowNet and the computation of meaning. World Scientific Publishing Co., Inc. 2006.

Du, B.; Tian, H.; Wang, L. & Lu, R. (2005), Design of domain-specific term extractor based on multi-strategy. Computer Engineering, 31(14):159¨C160, 2005

Du, X. & Li, M. (2006). "A Survey of Ontology Learning Research", Journal of Software, Vol. 17 No.9 pp. 1837-1847, Sep. 2006.

Gelfand, B.; Wulfekuler, M. & Punch. W. F. (1998). Automated concept extraction from plain text. In: AAAI 1998 Workshop on Text Categorization, pp. 13--17, Madison, WI, 1998.

Gusfield,D. (1997). Algorithms on Strings, Trees and Sequences : Computer Science and Computational Biology[M]. Cambridge University Press, 1997.

Hearst, M.A. (1992). Automatic acquisition of hyponyms from large text corpora. Proceedings of the 14th International Conference on Computational Linguistics (COLING), pp. 539--545, 1992.

Hinneburg, A. & Keim, D. (1998). An efficient approach to clustering in large multimedia databases with noise. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, 1998.

Laurence, S. & Margolis, E. (1999). Concepts: Core Readings, Cambridge, Mass. MIT Press. 1999.

Liu L. & et al. (2005). Acquiring Hyponymy Relations from Large Chinese Corpus, WSEAS Transactions on Business and Economics, Vol.2, No.4, pp.211-218, 2005

Liu, L.; Cao, C.; Wang, H.& Chen, W. (2006). A Method of Hyponym Acquisition Based on "isa" Pattern, J. of Computer Science. pp146-151.2006

Lu, C.; Liang, Z.& Guo, A. (1992). The semantic networks: a knowledge representation of Chinese information process. In: ICCIP'92. pp. 50--57

MacQueen,J.(1967). Some Methods for classification and Analysis of Multivariate Observations. In proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, pp281-297, 1967.

Nakata, K.; Voss, A.; Juhnke, M. & Kreifelts, T. (1998). Collaborative Concept Extraction from Documents. In U. Reimer, editor, Proc. Second International Conference on Practical Aspects of Knowledge Management (PAKM 98), Basel, 1998.

Ramirez, P.M. & Mattmann, C.A. (2004). ACE: improving search engines via Automatic Concept Extraction. In: Proceedings of the 2004 IEEE International Conference(2004) pp. 229--234.

Rydin,S. (2002). Building a hyponymy lexicon with hierarchical structures, In Proceedings of the Workship of the ACL Special Interest Group on the lexicon (SIGLEX), Philadelphia, July 2002, pp. 26-33, 2002

Tian, G. (2007). Research os Self-Supervised Knowledge Acquisition from Text based on Constrained Chinese Corpora. A dissertation submitted to Graduate University of the Chinese Academy of Sciences for the degree of Doctor of Philosophy. Beijing China, May 2007.

Velardi, P.; Fabriani, P. & Missikoff, M. (2001). Using text processing techniques to automatically enrich a domain ontology. In: Proc. Of the FOIS. New York: ACM Press, 2001. pp. 270--284.

Yu. L. (2006). A Research on Acquisition and Verification of Concepts from Large-Scale Chinese Corpora. A dissertation Submitted to Graduate School of the Chinese academy of Sciences for the degree of master. Beijing China, May 2006

Zelenko, D.; Aone, C. & Richardella, A.(2003). "Kernel Methods for Relation Extraction", Journal of Machine Learning Research, No.3, pp .1083-1106, 2003

Zhang, C. & Hao, T. (2005). The State of the Art and Difficulties in Automatic Chinese Word Segmentation. Journal of Chinese System Simulation. Vol.17 No.1 138¨C147. 2005.

Zhang, C. et al. (2007). Extracting Hyponymy Relations from domain-specific free texts, In proceedings of the 9th Int. Conf. on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007.

Zhang, Y.; Gong, L.; Wang, Y. & Yin, Z. (2003). An Effective Concept Extraction Method for Improving Text Classification Performance. Geo-Spatial Information Science. Vol. 6, No.4, 2003

Zheng J. & Lu J. (2005). Study of an improved keywords distillation method. Computer Engineering, 31:194¨C196, 2005

Zhou, J.; Wang, S. & Cao,C. (2007), A Google-Based Statistical Acquisition Model of Chinese Lexical Concepts. In proceedings of KSEM07, pp243-254. 2007

20

Cognitive Learning for Sentence Understanding

Yi Guo, Zhiqing Shao
Department of Computer Science and Engineering
East China University of Science and Technology
China

1. Introduction

In the research field of natural language understanding, sentence stands a very prominent position in text processing. The process of sentence understanding involves computing the meaning of a sentence based on analysis of meanings of its individual words. Research procedures in sentence understanding examine the representations and processes that connect the identification of individual words in text reading (Culter, 1995; Balota, 1994) with mapping sentence meanings to relevant mental models (Johnson-Laird, 1983) or discourse representations (Kintsch, 1988; van Eijck & Kamp, 1997).

The task of sentence understanding includes two stages, sentence parsing and semantic processing. Sentence parsing resides in the fundamental level, while semantic understanding involves lexcial and higher discourse analysis. Sentence understanding has compact connections with human cognition, thus this chaper will introduce how cognitive models are integrated, with machine learning algorithms (or models), into the procedures of sentence parsing and semantic processing.

2. Statistical Learning Review

Over the past decade, statistical learning, a means to discover hidden structures or patterns by analyzing statistical properties of the input, has emerged a general candidate mechanism by which a wide range of linguistic experience can be acquired (Saffran, 2003).

Statistical learning, as a type of implicit learning, has been demonstrated across a variety of natural and artificial language learning situations, including learning of information that is potentially highly relevant to sentence comprehension processes, such as using function words to delineate phrases (Green, 1979), integrating prosodic and morphological cues in the learning of phrase structure (Morgan et al., 1987), parsing each natural language sentence (Charniak, 1997) to a hierarchical structure which presents how words hookup together to form constituents, discovering phonological and distributional cues to lexical categories (Monaghan et al., 2005), locating syntactic phrase boundaries (Saffran, 2001; 2003), and detecting long-distance relationships between words (Gómez, 2002; Onnis et al., 2003).

Misyak & Christiansen (2007) revealed that statistical learning ability was a stronger predictor of relative clause comprehension than the reading span measure, and suggested that statistical learning may play a strong role in the accumulation of linguistic experience relevant for sentence processing.

Moreover, within natural language comprehension and production studies, there is clear evidence that prior experience of a given syntactic structure affects (1) comprehension of similar structures and (2) the probability that a speaker will utter a sentence with the same or similar structure, even when there is no meaning overlap between sentences (Ferreira & Bock, 2006).

Syntactic priming has been described as stemming from statistical learning at the syntactic level (Bock & Griffin, 2000; Chang et al., 2006) or at the syntactic–semantic interface (Chang et al., 2003), which can be viewed as examples of statistical learning of information relevant to sentence processing.

Above research works have testified the significance of statistical learning for natural language processing, including sentence comprehension, and also explicitly pointed out the performance bottlenecks (Monaghan et al., 2005; Dell & Bock, 2006; Misyak & Christiansen, 2007) of statistical processing technologies. Since human, rather than the computer software and hardware, is the core subject to process and understand natural language, it is essential to survey pivotal research works regarding human cognition.

3. Cognitive Concepts Highlight

Sentences convey not only lexico-semantic information for each word, but sentence meaning based on syntactic structures (Townsend & Bever, 2001; Friederici, 2002), which has elucidated the importance of syntactic structures for sentences.

Recursion is a unique human component of the faculty of language (Hauser et al., 2002), which is also known as the property of discrete infinity, the ability to generate an infinite range of discrete expressions from a finite set of elements. Sentences are indeed such infinite expressions generated from a limited set of words, signs, or letters; and syntactic mechanisms (Chomsky, 2000) have been applied to instantiate this property.

Thus, the processing of syntactic structures plays a critical role in the selective integration of lexico-semantic information into sentence meaning. Syntactic analyses are performed in the service of semantics, and sentence meaning is derived from syntactic analyses of the sentence structures.

As mentioned before, the procedure of sentence understanding includes sentence parsing at a fundamental level, and semantic understanding at lexcial and higher discourse analysis. This section will highlight several cognitive concepts regarding sentence parsing and emantic understanding.

3.1 Syntax-First and Interactive Models

How human beings parse sentences, especially for syntactically ambiguous sentences, has been a long-history cognitive research topic attracting research efforts for decades in the field of cognitive psychology. In cognitive psychology, behavioristic experiments have been popularly implemented to explore the sentence-analyzing mechanism, which is also called "parser", especially in the case that human beings cannot automatically constitute the meaning of a sentence.

With respect to syntactic and semantic processing in sentence comprehension, two main classes of cognitive models have been proposed to account for the behavioral data: Syntax-First and Interactive models.

Syntax-First models (Fodor, 1983; Frazier & Fodor, 1978; Kako & Wagner, 2001) claims that, (1) syntax plays the main part whereas semantics is only a supporting role, (2) the parser initially builds a syntactic structure based on word category information, which is independent from lexical or semantic information, and (3) thematic role assignment takes place during a second stage. If the initial syntactic structure cannot be mapped onto the thematic structure, the final stage will require a re-analysis.

Interactive models (Bates & Mac-Whinney, 1987; MacDonald et al., 1994; Marslen-Wislon & Tyler, 1980; Taraban & McClelland, 1988) state that syntactic and semantic processes actually interact with each other at an early stage, and both syntax and semantics work together to determine the meaning of a sentence. Despite the agreement that syntactic and semantic information has to be integrated within a short period of time, the two model classes differ in their views on the temporal structure of the integration processes.

Syntax and semantics are two indispensable properties of sentences. The eye-tracking studies (Tanenhaus & Trueswell, 1995) have supported the conclusion that syntax and semantics interact during parsing, which denotes that meaning affects early processing. These behavioristic experiments have convinced that the interactionist approach (Trueswell et al., 1994) is rational and effective to simulate human parsing and semantic understanding mechanism.

3.2 The Garden Path Model and Alternatives

Theories of sentence processing have illustrated various perspectives on when comprehenders initiate semantic interpretation of an incoming word. One of the most prominent and influential models of sentence processing is the garden path model (Ferreira & Clifton, 1986; Frazier & Rayner, 1982; Rayner et al., 1983), which states that semantic interpretation generally followes the construction of a syntactic analysis.

The syntactic analysis applies appropriate syntactic parsing strategy together with information of major syntactic category (e.g. noun, verb, adjective, etc.) of incoming words. Semantic interpretation can proceed after he construction of a syntactic analysis. The strict temporal ordering of syntactic analysis and semantic interpretation produce the fact that semantic information cannot influence the construction of a syntactic analysis. The effects of semantic information observed during the resolution of syntactic ambiguity have been interpreted as reflecting processes occurring after an initial syntactic analysis (Ferreira & Clifton, 1986; Kennison, 2001; Speer & Clifton, 1998).

From the perspective of the garden path theory, the results of the present research can be viewed as supporting the claim that certain aspects of high-level integrative semantic processing for an incoming word occur only after the comprehender determines the word's syntactic analysis.

The most prominent alternatives to the garden path model include interactive and constraint-based approaches to sentence processing. These approaches stated that language comprehension can be achieved through highly interactive and parallel processing (MacDonald et al., 1994; Sedivy et al., 1999; Tanenhaus et al., 1995; Taraban & McClelland, 1988; Trueswell & Tanenhaus, 1994; Trueswell et al., 1993). Word-specific (lexical) information will produce candidate syntactic frames, which are activated in parallel.

Although semantic interpretations are constructed upon syntactic frames (MacDonald et al., 1994), semantic information can influence the activation of syntactic frames. As a consequence, syntactic and semantic analysis may influence each other.

3.3 The Brain-Based Model

In the brain-based model (Friederici, 2002), language comprehension is divided into three functionally and temporally separable processing steps: (1) initial local structure building in the first phase; (2) lexical-semantic and thematic processes in the second phase; and (3) syntactic integration and revision in the third phase. For an integrative view of language processing, recent brain image research (Friederici & Kotz, 2003) provides support evidence that syntax-first aspects take place in an early time window and the interactive aspects happen in a later time window.

3.4 Working Memory and Semantic Memory

An early study (Fodor, 1983) of sentence understanding hypothesized a cognitive architecture focusing on a component building grammatical structures of sentence processing. Later research works (Caplan & Waters, 1999; Gibson, 1998; Just & Carpenter, 1992; Zurif et al., 1995) involve various executive resources facilitating sentence processing, such as working memory (WM), which contains specific sentence features and acts as temporary storage for phrasal information manipulation during the processing of long-distance syntactic dependencies in a sentence. During the course of sentence processing, working memory may help maintain, in a linear or non-linear manner, crucial components of a sentence in an active state until the correct grammatical relationships are established (Lewis et al., 2006).

Tulving (1972) first introduced semantic memory (SM), which refers to the general knowledge of concepts and facts, including word meaning, and involves encoding and retrieval of information in multiple domains (Hart et al., 2007). The essence of semantic memory is that contents are not statically bound to any particular instance of experience as in episodic memory. Instead, semantic memory stores is the gist of experience, an abstract structure applicable to a wide range of experiential objects, and delineates categorical and functional relationships between such objects.

4. Simple Recurrent Networks (SRNs)

Hadley (1994) proposed that systematic behavior is a matter of learning and generalization; thus, a neural network trained on a limited number of sentences should to be able to process all possible sentences in a generalize manner. Moreover, since people learn systematic language behavior from exposure to only a small fraction of possible sentences, a neural network should similarly be able to learn from a relatively small proportion of possible sentences, if it is to be considered cognitively plausible.

Simple Recurrent Networks (SRNs) (Elman, 1991) has been widely applied in basic connectionist approaches (parallel distributed processing) for language learning. SRN has been implemented to employ the functions of working memory (MacDonald et al., 2001; MacDonald & Christiansen, 2002).

The SRN architecture (as illustrated in Fig. 1.) includes the activations from the recurrent layer (RL, the hidden layer) as the context layer (CL) in the input layer (IL), aiming at processing inputs that consist of sequences of patterns of variable length. This architecture allows the network to include information connected with all the previous steps in a sequence in its processing of the current stage. The architecture will remember what has gone before, forgetting gradually as it progresses through the sequence.

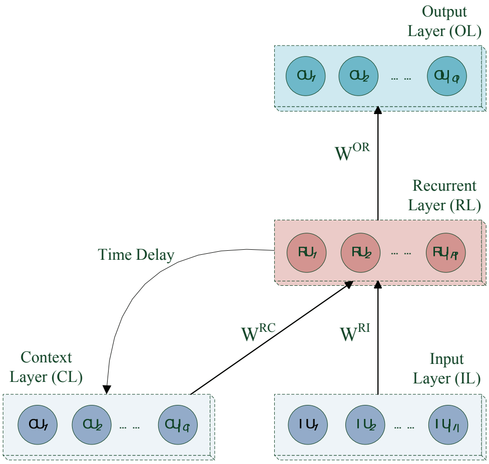


Fig.1. Architecture of Simple Recurrent Networks

Symbols	*Definition*
IU	A unit of input layer
RU	A unit of recurrent layer
CU	A unit of context layer
OU	A unit of output layer
$\lvert I \rvert$	The number of units in IL
$\lvert R \rvert$	The number of units in RL
$\lvert C \rvert$	The number of units in CL
$\lvert O \rvert$	The number of units in OL
$\boldsymbol{W^{RI}}$	The weight vector from IL to RL
$\boldsymbol{W^{RC}}$	The weight vector from CL to RL
$\boldsymbol{W^{OR}}$	The weight vector from RL to OL

Table 1. Definition of SRN Symbols

Symbols in Fig. 1. are defined in table 1: the first order weight matrices W^{RI} and W^{OR} fully connect the units of the input layer (IL) , the recurrent layer (RL) and the output layer (OL) respectively, as in the feed forward multilayer perceptron (MLP). The current activities of recurrent units $RU^{(t)}$ are fed back through time delay connections to the context layer, which is presented as $\boldsymbol{CU^{(t+1)}} = \boldsymbol{RU^{(t)}}$.

Therefore, each unit in recurrent layer is fed by activities of all recurrent units from previous time step through recurrent weight matrix W^{RC}. The context layer, which is composed with activities of recurrent units from previous time step, can be viewed as an extension of input layer to the recurrent layer. Above working procedure represents the memory of the network via holding contextual information from previous time steps.

The weight matrices W^{RI} , W^{RC} and W^{OR} are presented as equations (1) to (3)

$$W^{RI} = [\,(w_1^{RI})^T, (w_2^{RI})^T, \ldots, (w_{|R|}^{RI})^T\,] = \begin{bmatrix} w_{11}^{ri} & w_{12}^{ri} & \cdots & w_{1,|R|}^{ri} \\ w_{21}^{ri} & w_{22}^{ri} & \cdots & w_{1,|R|}^{ri} \\ \vdots & \vdots & \ddots & \vdots \\ w_{|I|,1}^{ri} & w_{|I|,2}^{ri} & \cdots & w_{|I|,|R|}^{ri} \end{bmatrix} \qquad (1)$$

$$W^{RC} = [\,(w_1^{RC})^T, (w_2^{RC})^T, \ldots, (w_{|R|}^{RC})^T\,] = \begin{bmatrix} w_{11}^{rc} & w_{12}^{rc} & \cdots & w_{1,|R|}^{rc} \\ w_{21}^{rc} & w_{22}^{rc} & \cdots & w_{1,|R|}^{rc} \\ \vdots & \vdots & \ddots & \vdots \\ w_{|C|,1}^{rc} & w_{|C|,2}^{rc} & \cdots & w_{|C|,|R|}^{rc} \end{bmatrix} \qquad (2)$$

$$W^{OR} = [\,(w_1^{OR})^T, (w_2^{OR})^T, \ldots, (w_{|O|}^{OR})^T\,] = \begin{bmatrix} w_{11}^{or} & w_{12}^{or} & \cdots & w_{1,|O|}^{or} \\ w_{21}^{or} & w_{22}^{or} & \cdots & w_{1,|O|}^{or} \\ \vdots & \vdots & \ddots & \vdots \\ w_{|R|,1}^{or} & w_{|R|,2}^{or} & \cdots & w_{|R|,|O|}^{or} \end{bmatrix} \qquad (3)$$

In above formulations, where $(w_k^{RI})^T$ is the transpose of w_k^{RI} for the instance of W^{RI} , where w_k^{RI} is a row vector, $(w_k^{RI})^T$ is the column vector of the same elements. The vector w_k^{RI} = (w_{1k}^{ri}, w_{2k}^{ri} ,... $w_{|I|,k}^{ri}$) represents the weights from all the input layer units to the recurrent (hidden) layer unit RU_k . The same conclusion applies with W^{RC} and W^{OR}

Given an input pattern in time ***t***, $\boldsymbol{IU}^{(t)}$ = ($IU_1^{(t)}, IU_2^{(t)}, \ldots, IU_{|I|}^{(t)}$), and recurrent activities $\boldsymbol{RU}^{(t)}$ = ($RU_1^{(t)}, RU_2^{(t)}, \ldots, RU_{|R|}^{(t)}$)), for the *i*th recurrent unit, the net input $RU_i'^{(t)}$ and output activity $RU_i^{(t)}$ are calculated as equations (4) and (5).

$$RU_i'^{(t)} = \boldsymbol{IU}^{(t)} \cdot (w_i^{RI})^T + \boldsymbol{RU}^{(t\text{-}1)} \cdot (w_i^{RC})^T = \sum_{j=1}^{|I|} IU_j^{(t)}\, w_{ji}^{ri} + \sum_{j=1}^{|R|} RU_j^{(t-1)} w_{ji}^{rc} \qquad (4)$$

$$RU_i^{(t)} = f(RU_i'^{(t)}) \qquad (5)$$

For the *k*th output unit, its net input $OU_k'^{(t)}$ and output activity $OU_k^{(t)}$ are calculated as equations (6) and (7).

$$OU_k'^{(t)} = \mathbf{RU}^{(t)} \cdot (w_k^{OR})^T = \sum_{j=1}^{|R|} RU_j^{(t)} w_{jk}^{or} \tag{6}$$

$$OU_k^{(t)} = f(OU_k'^{(t)}) \tag{7}$$

Here, the activation function *f* applies the logistic sigmoid function (Eq. 8).

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} \tag{8}$$

5. Cognitive Learning with Machine

Most current cognitive models of language processing agree that sentence comprehension involves different types of constraints (Jackendoff, 2002) in which syntactic and semantic (conceptual) information deserve the most salient consideration.

From one point of view, separable, independent but partly sequential processes construct distinct syntactic and semantic representations of a sentence (Berwick & Weinberg, 1984; Ferreira & Clifton, 1986). The opposed view is that syntactic and semantic constraints directly and simultaneously interact with each other at the message-level representation of the input (Johnson-Laird, 1983; Marslen-Wilson & Tyler, 1987; McClelland et al., 1989).

There also exist other proposals in between above fully independent and fully interactive models. Frazier (1987) suggests that syntactic analysis is autonomous and independent from semantic variables in initial stage(s), but is affected by semantic variables at later stage(s); in the contrary, syntactic analysis can influence semantic integration from the very beginning of processing. Meanwhile, Trueswell et al., (1994) claims that semantic information affects and leads syntactic analysis of the utterance in an immediate and direct mannner.

Abve and several other diverging proposals can be testified by using event-related brain potentials (ERPs), measurements of brain activity, which are elicited during the process of sentence comprehension. Different reliable ERP components have been employed to prove the distinction between the processing of syntactic and semantic information during sentence understanding. The extent and type of interaction of ERPs are taken as evidence for the interplay occurring between syntactic and semantic analyses during sentence comprehension.

This section will focus on how syntactic parsing and semantic processing are implemented with sentence processing machinery.

5.1 Sentence Parsing

The processing of syntactic structures plays a critical role in the selective integration of lexico-semantic information into sentence meaning. Syntactic analyses are performed in the service of semantics, and sentence meaning is derived from syntactic analyses of the sentence structures.

As discussed in section 3.1, the behavior experiments proved that semantics and syntax work together in sentence parsing to clarify the meaning. As a conclusion, semantics should be assigned an equal prominent role as syntax to improve parsing results. Thus, how to incorporate semantics with syntax simultaneously is the dominant challenge in sentence parsing.

In recent a few years, the research works of natural language processing (NLP) have strived toward the elaboration of huge linguistic dictionaries and ontologies (Knight et al., 1995; Miller et al., 1990; Sugumaran & Storey, 2002), even including relations between concepts and common sense. The exploitation and implementation of such dictionaries and ontologies has fulfilled some *understanding* requirements.

Kapetanios et al. (2005) proposed to implement the process of parsing natural language queries with an ontology, which preserved extensional semantics, such as domain terms, operators and operations. Since the context of terms circumscribed by the real-world semantics can be expressed by the ontology, it also will alleviate the semantic parsing. Context of terms is defined by the interrelationships expressed with an ontology as well as by the intentional meaning expressed with annotations.

Considering the impacts of linguistic dictionaries and ontologies in NLP, our solution for interactionist parsing, CIParser, takes WordNet (Miller et al., 1990) as the linguistic dictionary, and designs a corresponding ontology, *WNOnto* (as defined in Guo & Shao (2008)), referring to a W3C working draft (van Assem et al., 2006). Since nouns and verbs are more dominant in parsing sentences into phrases, they are the word types deliberately chosen for semantical analysis with WordNet. Therefore, the design of WNOnto grounds on nouns and verbs, which also benefits time efficiency in machine learning and parsing.

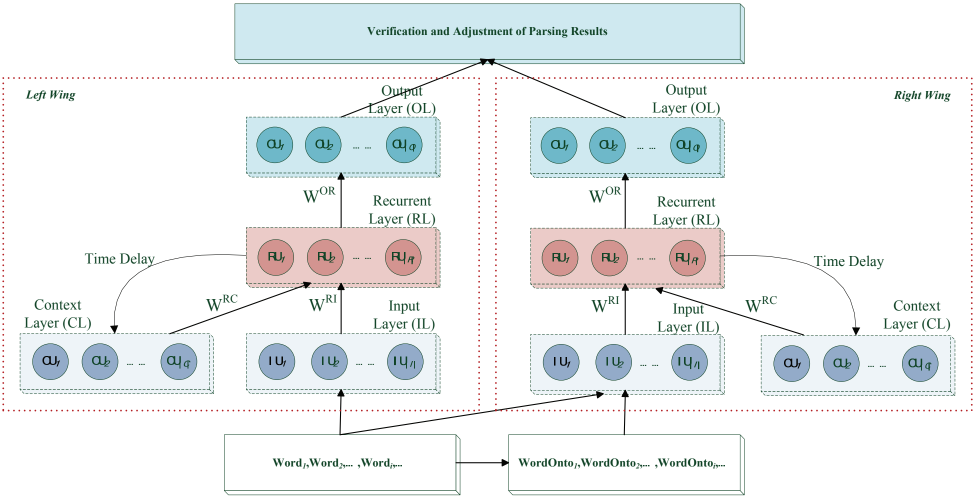


Fig. 2. Architecture of CIParser

Based on the architecture of SRN (figure 1), our CIParser is designed as illustrated in Fig. 2. The left wing is a classical SRN as described in section 4: all the input units in IL are single words from original sentences; the activations from RL of the previous time step produce the CL for the current stage; the units of IL and CL respectively multiplying matrices of W^{RI} and W^{RC} compose the input of RL; the activations of RL multiplying W^{OR} produce the input units of OL in current stage.

All the grammatical information is implicitly preserved in its pattern of link weights. Moreover, there are fewer independence assumptions. The SRN itself decides what to pay attention to and what to ignore. Statistical issues, such as combining multiple estimators or smoothing for sparse data, are handled in the training procedure. "One-size-fits-all" is a common feature of machine learning techniques.

The right wing is structurally identical as the left wing, except that the input units in IL include not only single words from sentences but also individual ontologies, *WordOnto*s, produced according to WNOnto with querying results from WordNet. In another word, each input unit of IL is composed with (1) a single word and (2) a corresponding ontology (only for a noun or verb). Here, any noun or verb has been appended with its semantical information from WordNet in the ontology manner.

The syntactic structure of a natural language sentence is a hierarchical structure, which represents how the words connect together to form constituents, such as phrases and even clauses. This structure is normally specified with a constituent-tree, in which the constituents are nodes or leaves and the hierarchical structure is denoted with parent-child relationships.

In the final processing phase, "Verification and Adjustment of Parsing Results", the parsing results of left and right wings are verified against each other in case that either wing takes too long time to deliver a parsing result. In the case of both wings producing parsing results, we have followed a selection rule that the tree containing more constituents wins, which has been strictly followed in later experiments. The application of phrases to identify structural constituents in our CIParser also offers the competence to generalize machine learned information across structural constituents.

As we know that (1) people has language processing constraints in constructions, such as center embedding (Chomsky, 1959), and (2) people can only activate a limited number of information units in memory at any one time (Miller, 1956), we introduced working memory (Baddeley & Susan, 2006) into our CIParser. Baddeley et al. (2006) defined working memory as a limited capacity system for temporary storage and manipulation of information for complex tasks such as comprehension, learning and reasoning. In this paper, we add the storage task of working memory to our CIParser to simulate human processing features.

The nature of SRN decides that each new input, a word or/and its ontology, of the network, will also be input of the network in a new state, which indicates that information is computed through all of these states in every subsequent time period. However, the constraints on the depth of center embedding (Chomsky, 1959) implies that a limited number of these states will be referred to by following parts of the constituent-tree in any given time period.

In CIParser, we construct a queue with limited length to simulate the active units in human memory. When the SRNs arrive at a new state, this state will be queued from head to tail. When a new state comes to the queue fully filled with previous network states, the oldest state leaves the queue at tail and the new one enters the head. This queue mechanism presents that, when the number of states exceeds the queue length, the oldest state will be forgotten. This mechanism also helps the CIParser to focus on active states and to achieve precise computing results efficiently.

Guo & Shao (2008) has designed and constructed experiments for training and examining CIParser in sentence parsing. The experiments demonstrate that the SRN-based CIParser may be used for connectionist language learning with structured output representations.

The performance of CIParser is evaluated in terms of traditional measures, *Precision* and *Recall* of constituents with the famous SUSANNE Corpus.

The experimental results demonstrate that the CIParser has comparability with the state-of-the-art parsing techniques based on statistical language learning. Guo & Shao (2008) also pointed out that (1) thinking of the parsing efficiency, only the semantic information of nouns and verbs are considered in current stage; (2) involving other word types (e.g. adverb and adjective) will be future research efforts.

5.2 Semantic Processing

As we know, several knowledge repositories, e.g. WordNet (Miller et al., 1990) and Cyc (Lenat, 2006), have been developed to support programs (or agents) to increase the intelligence of specified tasks. Meanwhile, other existing repositories are domain dependent and only represent information about certain aspects of the domains.

WordNet, as a linguistic repository, does not have the capability to capture the semantic relationships or integrity constraints between concepts. As linguistic repositories lack semantic knowledge, query expansion cannot deal with several issues: (1) knowledge related to the domain of the query, (2) common sense inferences, or (3) the semantic relationships in which the concepts of the query can participate.

The Cyc ontology is a semantic repository developed to capture and represent common sense, but can not represent linguistic relationships of the concepts (e.g. whether two concepts are synonyms). Semantic repositories need linguistic knowledge to identify relevant concepts from the repository that represent a given term used in the query. Thus, a semantic repository, as Cyc, can be extended with linguistic information from the WordNet lexicon, and factual information from the World Wide Web.

In section 5.1, we have illustrated a model for sentence parsing, and we will construct another model (as Fig. 3.) for semantic processing in this section. In order to implement semantic processing in sentence understanding, we have to consider semantic repositories to represent semantic information; the integration of linguistic and semantic information could be useful to increase the contexts where knowledge in these repositories can be used successfully.

In step one, each original sentence will be first processed by CIParser to obtain a corresponding syntactic structure, e.g. a constituent tree. In step two, as the sentence is processed word by word, open and closed class words are segregated into distinct processing streams. The Grammatical Relations Mapping module will integrate constituent information for each word or phrase with strict mapping operations. In step three, the Linguistic Relations Construction module constructs linguistic relationships (e.g. synonyms, antonyms, hypernyms, and hyponyms) of the concepts in a sentence with referent provided by WordNet. In step four, the Semantic Relations Construction module captures the semantic relationships or integrity constraints between concepts, so as to successfully deal with domain knowledge and common sense inferences. Finally, in step five, all the structured data (instances of ontolgy in XML format) from previous processing steps are used to fulfill the appropriate components of the meaning structure, the Sentence-Meaning Construction Index (SMCI). Obviously, SMCI contains four types, lexical, syntactic, grammatical and semantic (or conceptual) of information.

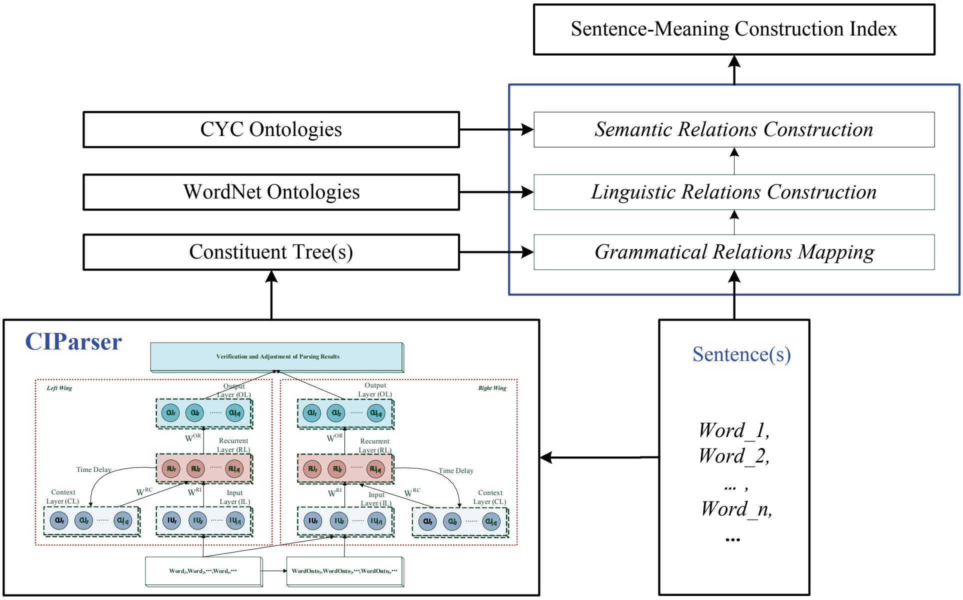

Sentence-Meaning Construction Index
CYC Ontologies
WordNet Ontologies
Constituent Tree(s)
Semantic Relations Construction
Linguistic Relations Construction
Grammatical Relations Mapping
CIParser
Verification and Adjustment of Parsing Results
Left Wing
Right Wing
Output Layer (OL)
Recurrent Layer (RL)
Input Layer (IL)
Context Layer (CL)
Time Delay
W^{OR}
W^{RC}
W^{RI}
OU_1 OU_2 ...... OU_q
RU_1 RU_2 ...... RU_q
IU_1 IU_2 ...... IU_n
CU_1 CU_2 ...... CU_q
$Word_1, Word_2, \cdots, Word_i, \cdots$
$WordOnto_1, WordOnto_2, \cdots, WordOnto_i, \cdots$
Sentence(s)
Word_1,
Word_2,
... ,
Word_n,
...

Fig. 3. A Computing Model of Semantic Processing for Sentence Understanding

The above model is able to store and retrieve different sentence-meaning construction appropriate for different sentences. The requirement is that each individual sentence should yield a unique construction index. The construction indices are used in a working memory or an associative memory to store and retrieve the correct sentence-meaning construction index.

6. Conclusion

This chaper starts with a review of classical and traditional statistical learning approaches. As sentence understanding has latent compact connections with human cognition, this chaper also highlights relevant cognitive concepts or models in sentence understanding domain. Afterwards, this chapter described the completion of sentence understanding task from two aspects, sentence parsing and semantic processing, and how cognitive models are integrated, with machine learning algorithms (or models), into the procedures of sentence parsing and semantic processing.

The CIParser has been evaluated and proven comparablr with the state-of-the-art parsing techniques based on statistical language learning. Another computing model of Semantic Processing for Sentence Understanding (Fig. 3.) also has been constructed to deliver Sentence-Meaning Construction Index (SMCI) for each sentence. With SMCI, a sentence can be understood in four dimensions, which are lexical, syntactic, grammatical and semantic (or conceptual) dimensions.

Cognitive learning with machines for sentence understanding has just started with minor productions, in which our works took SRNs as an initial model of artificial neural networks.

In an artificial language learning task (next-word prediction), van der Velde et al. (2004) evaluated a simple recurrent network (SRN) and claimed that the SRN failed to process novel sentences appropriately, for example, by correctly distinguishing between nouns and verbs. However, Frank (2006) extended above simulations and showed that, although limitations had arisen from overfitting in large networks (van der Velde et al., 2004), an identical SRN still can display some generalization performance in the condition that the lexicon size was increased properly. Moreover, Frank (2006) demonstrated that generalization could be further improved by employing the echo-state network (ESN) (Jaeger, 2003), an alternative network that requires less training (due to fixed input and recurrent weights) and is less prone to overfitting.

Recurrent Self-Organizing Networks (RSON) (Farkaš & Crocker, 2006), coupled with two types of a single-layer prediction module, had demonstrated salient benefit in learning temporal context representations. In the task of next-word prediction, RSON achieved the best performance, which turned out to be more robust and faster to train than SRN and higher prediction accuracy than ESN. As a conclusiong, further investigation will take ESN and RSON as neural network models, and we believe that comparison and evalation works among SRNs, ESNs, and RSONs are also venturing and promising directions.

7. Acknowledgement

This research work has been partly funded with the Returned Scholar Research Funding of Chinese Ministry of Education (MoE).

8. References

Baddeley, A. & Susan, J. P. (2006). *Working Memory: An Overview. Working Memory and Education*, Academic Press, ISBN:0125544650, Burlington, pp.1-31.

Balota, D. A. (1994). Visual word recognition: the journey from features to meaning, In: *Handbook of Psycholinguistics*, Gernsbacher, M. A., (Ed.), pp.303-358, Academic Press, ISBN-10: 0122808908, New York

Bates, E. & MacWhinney, B. (1987). Competition, variation and language learning, In: MacWhinney, B. (Ed.), *Mechanisms of Language Acquisition*, Erlbaum, Hillsdale, NJ, pp. 157–194.

Berwick, R. & Weinberg, A. (1984). *The Grammatical Basis of Linguistic Performance*, MIT Press, Cambridge, MA.

Bock, K. & Griffin, Z. M. (2000). The persistence of structural priming: Transient activation or implicit learning? *Journal of Experimental Psychology: General*, Vol.129, pp.177–192.

Caplan, D. & Waters, G. S. (1999). Verbal working memory and sentence comprehension, *Behavioral and Brain Sciences*, Vol.22, pp.77–126.

Chang, F.; Bock, K. & Goldberg, A. (2003). Do thematic roles leave traces of their places? *Cognition*, No. 90, pp.29–49.

Chang, F.; Dell, G. S. & Bock, K. (2006). Becoming syntactic, *Psychological Review*, No.113, pp.234–272.

Charniak, E. (1997). Statistical Techniques for Natural Language Parsing, *AI Magazine*, No.18, pp. 33-43.

Chomsky, N. (1959). On Certain Formal Properties of Grammars, *Information and Control*, Vol. 2, pp. 137-167.

Chomsky, N. (2000). *New Horizons in the Study of Language and Mind*, Cambridge University Press, Cambridge, UK.

Culter, A. (1995). Spoke word recognition and production, In: *Speech, Language, and Communication (Handbook of Perception and Cognition)*, Miller, J. L. and Eimas, P. D., (Eds.), pp. 97-136, Academic Press, ISBN-10: 0124977707, New York

Elman, J. L. (1991). Distributed Representations, Simple Recurrent Networks, and Grammatical Structure, *Machine Learning*, Vol. 7, pp. 195-225.

Farkaš , I. & Crocker, M. (2006). Recurrent networks and natural language: exploiting self-organization, in: *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum, Hillsdale, NJ.

Ferreira, F. & Clifton, C. (1986). The independence of syntactic processing, *Journal of Memory and Language*, No. 25, pp. 348–368.

Ferreira, V. S. & Bock, K. (2006). The functions of structural priming. Language and Cognitive Processes, 21, 1011–1029.

Fodor, J.A. (1983). *The Modularity of Mind: An Essay on Faculty Psychology*, MIT Press, Cambridge, MA.

Frank, S. (2006). Learn more by training less: systematicity in sentence processing by recurrent networks, *Connection Science*, Vol.18, No.3, pp. 287–302.

Frazier, L. & Fodor, J.D. (1978). The sausage machine: a new two-stage model of the parser, *Cognition*, Vol. 6, pp. 291–325.

Frazier, L. & Rayner, K. (1982). Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences, *Cognitive Psychology*, Vol. 14, pp. 178–210.

Frazier, L. (1987). Sentence processing: a tutorial review, In: Coltheart, M. (Ed.), *Attention and Performance XII*, Erlbaum, London, pp. 559–585.

Friederici, A.D. (2002). Towards a neural basis of auditory sentence processing, *Trends of Cognition Science*, No. 6, pp.78-84.

Friederici, A.D. & Kotz, S.A. (2003). The brain basis of syntactic process: functional imaging and lesion studies, *NeuroImage*, Vol. 20, pp. 8–17.

Gibson, E. (1998). Linguistic complexity: Locality of syntactic dependencies, *Cognition*, No. 68, pp. 1–76.

Gómez, R. (2002). Word frequency effects in priming performance in young and older adults, *Journals of Gerontology*, Series B: Psychological Sciences and Social Sciences, 57B(3), pp. 233–240.

Green, T. R. (1979). The necessity of syntax markers: Two experiments with artificial languages, *Journal of Verbal Learning and Verbal Behavior*, Vol. 18, pp.481–496.

Guo, Y. & Shao, Z. (2008). The Cognitive Interactionist Approach of Sentence Parsing with Simple Recurrent Neural Networks, In: *Proceedings of International Symposium on Intelligent Information Technology Application 2008* (IITA 2008), IEEE Computer Society Press, Shanghai, China, pp.118-122.

Hadley, R. (1994). Systematicity in connectionist language learning, *Mind Language*, Vol. 9, No.3, pp. 247–272.

Hauser, M.D.; Chomsky, N. & Fitch, W.T. (2002). The faculty of language: what is it, who has it, and how did it evolve? *Science*, No. 298, pp.1569-1579.

Jackendoff, R. (2002). Foundations of language: brain. Meaning, Grammar, Evolution, *OUP*, Oxford, UK.

Jaeger, H. (2003). Adaptive nonlinear system identification with echo state networks, in: *Advances in Neural Information Processing Systems*, Vol. 15, MIT Press, Cambridge, MA.

Johnson-Laird, P. N. (1983). *Mental Models*, Harvard University Press, ISBN-10: 0674568818, Cambridge, MA.

Just, M. A. & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory, *Psychological Review*, Vol. 99, pp. 122–149.

Kako, E. & Wagner, L. (2001). The semantics of syntactic structures, *Trends in Cognitive Sciences*, Vol. 5, No. 3, pp. 102-118.

Kapetanios, E.; Baer, D. & Groenewoud, P. (2005). Simplifying syntactic and semantic parsing of NL-based queries in advanced application domains, *Data & Knowledge Engineering*, Vol. 55, No.1, pp. 38-58.

Kennison, S. M. (2001). Limitations on the use of verb information in sentence comprehension, *Psychonomic Bulletin and Review*, Vol. 8, pp. 132–138.

Kintsch, W. (1988). The use of knowledge in discourse processing: a construction-integration model, *Psychological Review*, No. 95, pp.163-182, ISSN: 0033-295X

Knight, K.; Chancer, I. & Haines, M. (1995). Filling knowledge gaps in a broad-coverage machine translation system, In : *Proceedings of IJCAI'95*, pp.1390-1397, Montreal, Canada.

Lenat, D.(2006). Hal's Legacy: 2001's Computer as Dream and Reality. From 2001 to 2001: Common Sense and the Mind of HAL, http://www.cyc.com/cyc/technology/halslegacy.html, Retrieved on 2006-09-26.

Lewis, R. L.; Vasishth, S., & VanDyke, J. A. (2006). Computational principles of working memory in sentence comprehension, *Trends in Cognitive Sciences*, Viol. 10, pp. 447–454.

MacDonald, M.C.; Pearlmutter, N.J. & Seidenberg, M.S. (1994). The lexical nature of syntactic ambiguity resolution, *Psychological Review*, No. 101, pp. 676–703.

MacDonald, M. C.; Almor, A. ; Henderson, V. W.; Kempler, D., & Andersen, E. (2001). Assessing working memory and language comprehension in Alzheimer's disease, *Brain & Language*, No. 78, pp. 17–42.

MacDonald, M. C. ; & Christiansen, M. H. (2002). Reassessing working memory: Comment on Just and Carpenter (1992) and Waters and Caplan (1996), *Psychological Review*, No. 109, pp. 35–54.

Marslen-Wislon, W. & Tyler, L.K. (1980). The temporal structure of spoken language understanding, *Cognition*, Vol. 8, pp. 1–71.

Marslen-Wilson, W.D. & Tyler, L.K. (1987). Against modularity, In: Gareld, J.L. (Ed.), *Modularity in Knowledge Representation and Natural-Language Understanding*, MIT Press, Cambridge, MA, pp. 37–62.

McClelland, J.L. ; St John, M. & Taraban, R. (1989). Sentence comprehension: a parallel distributed processing approach, *Language and Cognition Processes*, Vol. 4, pp. 287–336.

Miller, G. A. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, *Psychological Review*, No. 63, pp. 81-97.

Miller, G. A.; Beckwith, R.; Fellbaum, C.; Gross, D., & Miller, K. J. (1990). Introduction to WordNet: An On-Line Lexical Database, *International Journal of Lexicography*, Vol. 3, No. 4, pp. 235-312.

Misyak, J. B. & Christiansen, M. H. (2007). Extending statistical learning farther and further: Long-distance dependencies, and individual differences in statistical learning and language, In: D. S. McNamara & J. G. Trafton (Eds.), *Proceedings of the 29th annual cognitive science society conference*, pp. 1307-1312, Austin, TX: Cognitive Science Society.

Monaghan, P.; Chater, N. & Christiansen, M. H. (2005). The differential role of phonological and distributional cues in grammatical categorisation, *Cognition*, Vol. 96, pp. 143–182.

Morgan, J. L.; Meier, R. P. & Newport, E. L. (1987). Structural packaging in the input to language learning: Contributions of prosodic and morphological marking of phrases to the acquisition of language, *Cognitive Psychology*, Vol. 19, pp. 498–550.

Onnis, L.; Christiansen, M.; Chater, N. & Gómez, R. (2003). Reduction of uncertainty in human sequential learning: Evidence from artificial language learning, In: *Proceedings of the 25th annual conference of the cognitive science society*, Mahwah, NJ: Lawrence Erlbaum, pp. 886–891.

Rayner, K.; Carlson, M., & Frazier, L. (1983). The interaction of syntax and semantics during sentence processing, *Journal of Verbal Learning and Verbal Behavior*, No. 22, pp. 358–374.

Saffran, J. R. (2001). The use of predictive dependencies in language learning, *Journal of Memory and Language*, Vol. 44, No. 4, pp. 493–515.

Saffran, J. R. (2003). Statistical language learning: Mechanisms and constraints, *Current Directions in Psychological Science*, Vol. 12, No. 4, pp. 110–114.

Sedivy, J. C.; Tanenhaus, M. K.; Chambers, C. G. & Carlson, G. N. (1999). Achieving incremental semanticinterpretation through contextual representation, *Cognition*, No. 71, pp. 109–148.

Speer, S. R. & Clifton, C. (1998). Plausibility and argument structure in sentence comprehension, *Memory and Cognition*, Vol. 26, pp. 965–978.

Sugumaran, V. & Storey, V. C. (2002). An ontology based framework for generating and improving DB design, In: *Proceedings of 7th International Workshop on Applications of Natural Language to Information Systems*, pp. 1-12, Springer Verlag, Stockholm, Sweden.

Tanenhaus, M. K.; Spivey-Knowlton, M. J.; Eberhard, K. M. & Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension, *Science*, No. 268, pp. 1632–1634.

Tanenhaus, M. K. & Trueswell, J. C. (1995). *Sentence Comprehension. Speech, Language, and Communication (2nd)*, Academic Press, San Diego, pp. 217-262.

Taraban, R. & McClelland, J. R. (1988). Constituent attachment and thematic role assignment in sentence processing: Influence of content-base expectations, *Journal of Memory and Language*, Vol. 27, pp. 597–632.

Townsend, D. J. & Bever, T.G. (2001). *Sentence Comprehension: The Integration of Habits and Rules*, The MIT Press, Cambridge, MA.

Trueswell, J. & Tanenhaus, M. K. (1994). Toward a lexicalist framework for constraint-based syntactic ambiguity resolution, In: C. Clifton, L. Frazier, & K. Rayner (Eds.), *Perspectives on sentence processing*, pp. 155–179, Hillsdale, NJ: Lawrence Erlbaum Associates.

Trueswell, J. C.; Tanenhaus, M. K. & Garnsey, S. M. (1994). Semantic Influences On Parsing: Use of Thematic Role Information in Syntactic Ambiguity Resolution, *Journal of Memory and Language*, Vol. 33, No. 3, pp. 285-318.

Trueswell, J. C.; Tanenhaus, M. K. & Kello, C. (1993). Verb-specific constraints in sentence processing: Separating effects of lexical preference from garden-paths, *Journal of Experimental Psychology: Learning, Memory, and Cognition*, Vol. 19, pp. 528–553.

Tulving, E. (1972). Episodic and semantic memory, In : E. Tulving & W. Donaldson (Eds.), *Organization of memory*, New York: Academic Press.

van Assem, M.; Gangemi, A. & Schreiber, G. (2006). *RDF/OWL Representation of WordNet*, W3C Working Draft 19 June 2006, http://www.w3.org/TR/wordnet-rdf/#intrown.

van der Velde, F.; van der Voort, G.; van der Kleij; de Kamps, M.(2004). Lack of combinatorial productivity in language processing with simple recurrent networks, *Connection Science*, Vol. 16, No. 1, pp. 21–46.

van Eijck, J. & Kamp, H. (1997). Representing discourse in context, In: *Handbook of Logic and Language*, van Bentham, J. And ter Meulen, A., (Eds), pp.179-237, Elsevier, ISBN-10: 044481714X, Amsterdam

Zurif, E. B.; Swinney, D.; Prather, P.; WingWeld, A. & Brownell, H. H. (1995). The allocation of memory resources during sentence comprehension: Evidence from the elderly, *Journal of Psycholinguistic Research*, Vol. 24, pp. 165–182.

21

A Hebbian Learning Approach for Diffusion Tensor Analysis & Tractography

Dilek Göksel Duru
Bogazici University
Turkey

1. Introduction

The principle significance of an artificial neural network is that it learns and improves through that learning. The definition of the learning process in neural networks is of great importance. The neural network is stimulated and regarding to these stimulations the free parameters of the network change in its internal structure. As a result the neural network replies in a new way. Based on a basic learning algorithm namely Hebbian learning, a solution to the problem of resolving uncertainty areas in diffusion tensor magnetic resonance image (DTMRI) analysis is represented. Diffusion tensor imaging (DTI) is a developing and promising medical imaging modality allowing the determination of in-vivo tissue properties noninvasively upon the random movement of the water molecules. The method is unique in its ability being a noninvasive modality which is a great opportunity to explore various white matter pathologies and healthy brain mapping for neuroanatomy research. In neuroscience applications DTI is mostly used addressing brain’s fiber tractography, reconstructing the connectivity map. Clinical evaluation of fiber tracking results is a major problem in the field. Noise, partial volume effects, inefficiency of numerical implementations by reconstructing the intersecting tracts are some of the reasons for the need of standardized fiber tract atlas. Also misregistration caused by eddy currents, ghosting due to motion artifacts, and signal loss due to susceptibility variations may all affect the calculated tractography results.

The proposed method based on the Hebbian learning provides an instance of non-supervised and competitive learning in a neurobiological aspect as a solution to the tracking problem of the intersecting axonal structures. The main contribution of the study is to describe a tracking approach via a special class of artificial neural networks namely the Hebbian learning with improved reliability.

2. Diffusion tensor imaging

Essential concepts necessary to understand DTMRI are explained in this section. The utility of the diffusion tensor is that it provides the direction in three dimensional space in which the rate of diffusion is greatest (Basser et al., 2000). The developing imaging modality is almost a routine MR technique analyzing tissue anisotropy characteristics, connectivity and alterations of human brain neural tracts.

The discrete diffusion tensor and diffusivity trajectory estimation between neighboring image pixels are used to trace out the fiber pathways namely the tracts. The process of determining the neural tracts especially white matter structures by diffusion tensor analysis is commonly known as *tractography*. Fiber tractography is able to provide both quantitative and qualitative information aiming to clarify the anatomical architecture of brain's fibers and advance our knowledge of fiber connectivity maps (Ding et al., 2003). There are some limiting cases in DTI analysis and fiber tracking. One of the critical problems in estimating these brain maps is the existence of intersecting tracts within the tissue. As a consequence of this fact, axonal structures in the image voxels with more than one diffusivity direction can not be clearly defined, where the generally the diffusion tensor model becomes inaccurate to define the uncertainties (Bammer, 2003; Ciccarelli et al., 2003).

Current researchs are involved in multi-tensor mixture models (Tuch et al., 2002) and higher order tensor models (Basser et al., 1994; Basser, 2002). Some techniques such as q-space imaging (Callaghan et al. 1988; Basser, 2002), and high angular resolution diffusion imaging (Frank, 2002; Tuch et al., 2002) are enhanced in resolving such multidiffusivities within a voxel. Jones employed the so called "cone of uncertainty" as a construction method where the tensor's principal eigenvector has a confidence interval in which one helps to define the uncertainty regions as a cone with a probability distribution instead of a discrete diffusivity determination (Jones, 2003). In spite of having some proposed methods for determination of the intersecting diffusivities (Westin et al., 1999; Pajevic & Pierpaoli, 2000; Poupon et al., 2000; Tuch et al., 2002), still the connection is not precisely defined, and there isn't any gold standard yet (Westin et al., 1999; LeBihan et al., 2006). So depending on the proposed Hebbian learning rule approach, we aim to clarify the tracts in the intersections in order to eliminate the uncertainty.

3.1 Diffusion tensor theory

Diffusion weighted images are the raw data source for the calculation of the diffusion tensor measured using the Stejskal-Tanner equation (Basser & LeBihan, 1992; Basser et al., 1994), where $|g|$ is the strength of the diffusion gradient pulses, S_0 is the RF signal received for a measurement without diffusion gradient pulses, and S_i is the signal received with diffusion gradient pulses. The *diffusion tensor* $\boldsymbol{D}$ is gained by systematically application of the diffusion weighted gradients in multiple directions. The mathematical expression D is a real, symmetric second order tensor, represented in matrix form as a real, symmetric 3x3 matrix (Eq. 1).

$$S_i = S_0 e^{-b\hat{g}_i^T D \hat{g}_i} \tag{1}$$

Getting the six unique elements of the diffusion tensor D requires at least six diffusion weighted measurements in non-collinear measurement directions g along with a non-diffusion-weighted measurement S_0 based on the three-dimensional Gaussian Stejskal-

Tanner model (Eq.2). The linear system of $n \geq 6$ diffusion weighted measurements constraining the diffusion tensor D may be represented in matrix form (Basser, 2002).

$$\begin{bmatrix} x_1^2 & y_1^2 & z_1^2 & 2x_1y_1 & 2y_1z_1 & 2x_1z_1 \\ x_2^2 & y_2^2 & z_2^2 & 2x_1y_1 & 2y_2z_2 & 2x_2z_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & y_n^2 & z_n^2 & 2x_ny_n & 2y_nz_n & 2x_nz_n \end{bmatrix} \begin{bmatrix} D_{xx} \\ D_{yy} \\ D_{zz} \\ D_{xy} \\ D_{xz} \\ D_{yz} \end{bmatrix} = \begin{bmatrix} -\frac{1}{b}\ln\frac{S_1}{S_0} \\ -\frac{1}{b}\ln\frac{S_2}{S_0} \\ \vdots \\ -\frac{1}{b}\ln\frac{S_n}{S_0} \end{bmatrix} \tag{2}$$

In the linear system of equations $Ad = s$ of equation 2, A is the encoding matrix containing the $n \geq 6$ unit normalized gradient measurement directions, d is a vector specifying the 6 unique elements of the diffusion tensor D (Eq.3), and s is a vector containing natural logarithmic scaled RF signal loss resulting from the Brownian motion of spins (Berg, 1983).

$$D = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \tag{3}$$

Diagonalization of the diffusion tensor yields its eigensystem with its eigenvectors and eigenvalues λ_1, λ_2, λ_3 listed in decreasing order. The eigensystem of D is calculated in every pixel for DT analysis. Different research groups have studied mathematical explanations of the diffusion tensor (Lori et al., 2002; Ciccarelli et al., 2003; Khader & Narayana, 2003). The scalar diffusion tensor eigenvalues and their derivatives defining the anisotropy values are correlated to the underlying tissue in the region of interest (ROI). The directional eigenvectors are relevant to the anisotropy spatially and orientationally (Pierpaoli et al., 2002). The largest eigenvector actually corresponds to the major molecular motion of water indicating the principal orientation of the analyzed axonal structures. Most of the diffusion tensor analyses rely in assigning the major eigenvector as the direction of the largest water diffusion called the principal diffusivity for reconstructing the 3D trajectories of human brain fiber bundles (Westin et al., 1999; Basser et al., 2000; Poupon et al., 2000; Khader et al., 2003; Taylor et al., 2004). The approach is adopted in our implementation too. Using the computational diagonalization the eigensystem of the 3 by 3 symmetric D is achieved (Borisenko & Tarapov, 1979; Szafer et al., 1995; Göksel & Özkan, 2006). The eigenvectors e_i and the corresponding eigenvalues λ_i are the solutions of the equation (5), where the eigenvectors e_i (Eq. 4) are the principal diffusion directions $(i = 1, 2, 3)$.

$$D\vec{e}_i = \lambda_i\vec{e}_i \tag{4}$$

$$|\ D - \lambda I\,| = 0 \tag{5}$$

The calculated eigenvectors are ordered descending, and an ordered orthogonal basis with the first eigenvector having the direction of largest variance of the data is created (Goksel & Ozkan, 2006). In our sample, this leads to the principal diffusivity, and so the most appropriate diffusivity directions can be determined (Borisenko & Tarapov, 1979). The first principal component λ_1 has maximum variance, and thus its weighting coefficients give the direction of the maximum diffusion weighted signal, or largest principal diffusivity (Basser et al., 2000). The weighting coefficients of the second and third principal components λ_2 and λ_3 give the directions of the intermediate and smallest principal diffusivity respectively.

Estimating the fiber tract maps follow the implementation of the selected post processing methods, in this study the Hebbian learning rule, to resolve the related eigensystem. To begin the tracking process, generally a starting pixel also called a seed point is selected to focus on the desired region of interest and to avoid calculation overload in consideration of working on the whole brain DTMR data. Starting at the seed point coordinates, similar fiber orientation vectors are traced out upon a predefined similarity constraint until the selected ROI is fully covered. Specific tracts related to the investigated ROIs can be visualized by choosing the regions/seed points according to anatomical structures picked on the brain atlas where the selection can be made either on segmented DT brain map or unsegmented and full brain volume.

3. Hebbian learning for pattern association

3.1 Hebb's hypothesis

Hebb's rule is the earliest and the simplest of the learning rules for a neural network. The basic neural net model provides knowledge about the synaptic modifications and learning procedure between nodes in a pattern. The technique relies in representing the activity between correlated nodes according to their related weighting. The Organization of Behavior is expressed by Donald Hebb (1949) as: ``When the axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased''. This Hebbian learning rule is used to store patterns in artificial neural network models of associative memory. The Hebb's postulate of learning implies both temporal and spatial constraints on learning.

The ability of the Hebb's rule to determine the weights for the correct output related to all the training pattern is influenced by three factors. One is the existence of such weights. Linear independency is a must to provide the existence of the weights. The second factor is the correlation factor. The quality of the resulting weight matrix of the Hebbian learning depends on the orthogonality of the input vectors. Finally, the weights of the Hebb's process represent the simultaneous activation of the stimulated nodes unit by unit, which lead the rule make also called correlation training or encoding (Fausett, 1994). The rule is sometimes called the activity product rule also, because of the correlational characteristic of Hebb's hypothesis.

The idea selecting the Hebbian learning to resolve the uncertainty problem in diffusion tensor tractography (DTT) relies in the proficiency of the neural networks as classifiers especially in non-linear real world problems. The Hebbian learning algorithm is performed locally, which makes the application a plausible theory for biological learning methods. That is also making Hebbian learning process ideal in DTT.

In this study, the input pattern is actually the eigensystem defining the principal diffusivity of the fibers in DTMRI. As previously mentioned, the learning theory of Hebb relies in the increase of the weights between neighboring nodes by simultaneous activation. In other words, the weights between the nodes of the input pattern in Hebbian learning are representing the relationship between these nodes. The modification of the weights and the implementation will be explained in the next subtitles.

3.2 Mathematical model of Hebb's rule

The formulation of the Hebbian learning follows with a synaptic weight w_{kj} of neuron k with F a function of both input signal x_j and output signal y_k (Haykin, 1999):

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n)) \tag{6}$$

The Hebb's formula has many forms, the simplest form expressed with the weight modification is given as (Haykin, 1999):

$$\Delta w_{kj}(n) = \eta \ y_k(n)x_j(n) \tag{7}$$

The synaptic adjustment Δw_{kj} is applied to the synaptic weight vector w_{kj} at time step n with a learning rate parameter η. This proceeds from one step in the learning algorithm to another.

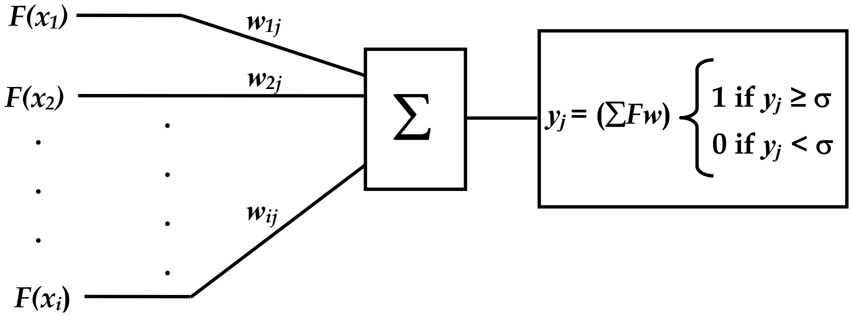


Fig. 1. The principal of the Hebbian learning: The activation function with a threshold σ is defining the output in other words activated, allowed vectors of the examined pattern

Generally, modification of a weight w_{ij} from an excited neuron x_i to a destination neuron y_j having a learning rate η, the Hebb's learning rule is defined formally as in Eq.(7) and expressed graphically as in Fig.1. Correlated nodes will have strong negative or positive weights regarding to their tendency being opposite or not, where uncorrelated nodes will have weights near zero. As a result, a neuron map is obtained with weights encoding the stationary probability density function of the input pattern vectors (Haykin, 1999). The procedure is an instance of unsupervised and competitive learning process. The advantages of the procedure is that it is a local learning rule, simple with very little computational requirement and biological plausibility.

As a basis of the proposed Hebbian method, independent component analysis (ICA) is applied to solve the intersecting fiber problem in DTT literature (Arfanakis et al., 2002; Sungheon et al., 2005; Jeong-Won & Singh, 2006). The application has provided useful information about the diffusion properties of brain structures without estimation of the diffusion tensor. But ICA should not be considered as a fiber tracking method. The technique maps structures. The advantage is that it may be used for visualization of particular structures without any user specific a priori information. Relying on the successful implementation of the ICA method in the literature, the basic Hebbian learning is proposed for the post processing of the diffusion weighted MR images.

3.3 The Hebbian rule in DTI

The fiber directions in living tissue should be compared carefully, thus simulation studies are done addressing verification and validation of the analysis. By generating the algorithm, first all the weights w_i=0 (i=1,…,n) are initialized. For each input training vector, the input units and the output unit are activated, while new weights are adjusted regarding Eq.(7). The weight modification Δw_{kj} follows until the ROI is covered according to the predefined constraints.

Our approach relies in the assumption that the axon follows a unique path. Each element in the Hebbian input pattern represents a voxel in the ROI, and each voxel is related with its neighboring voxels. To clarify the idea and the implementation steps, a sample synthetic fiber tract ROI with its eigenvectors in every pixel is given in Fig.2:

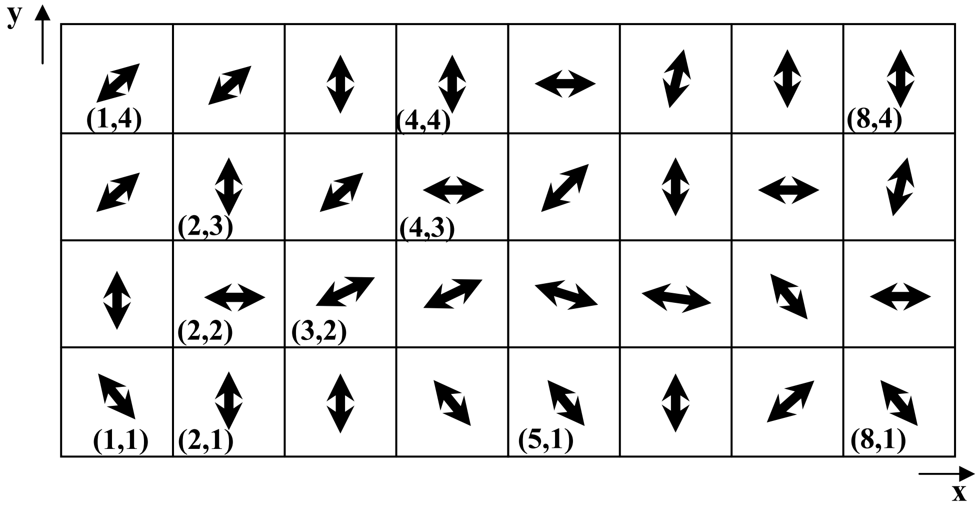


Fig. 2. A segment of synthetic fiber tract pattern illustrating the diffusion tensor orientations in each pixel with computed eigenvectors.

The seed point in Fig.2 is the voxel with the coordinate (4,2). The 8 neighboring nodes of the starting point are first investigated and in each step the weight is been updated. For the same sample, Gaussian noise is added to the pattern, and again weights are updated starting from the same seed point. As a result, the Hebb rule defines the green path as the winning path (Fig.3). Depending on the threshold function varying branches can be determined for the same ROI (Fig.4). For simulation studies various threshold functions are selected to verify the algorithm, and to define the simulated paths more precisely. In living tissue, the selection of the threshold constraint is depending actually on the human brain atlas and anatomical structure knowledge.

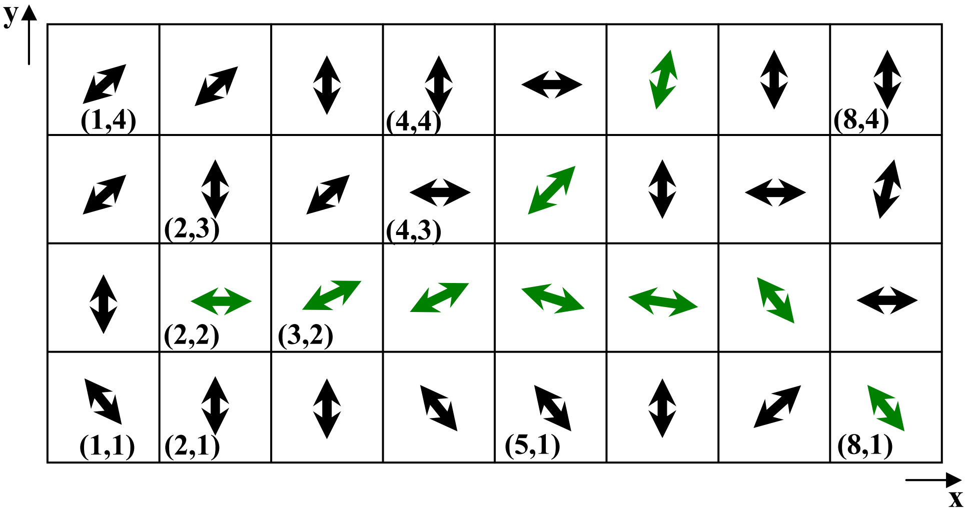


Fig. 3. The green branching path represents the winning tracts estimated by Hebb's learning

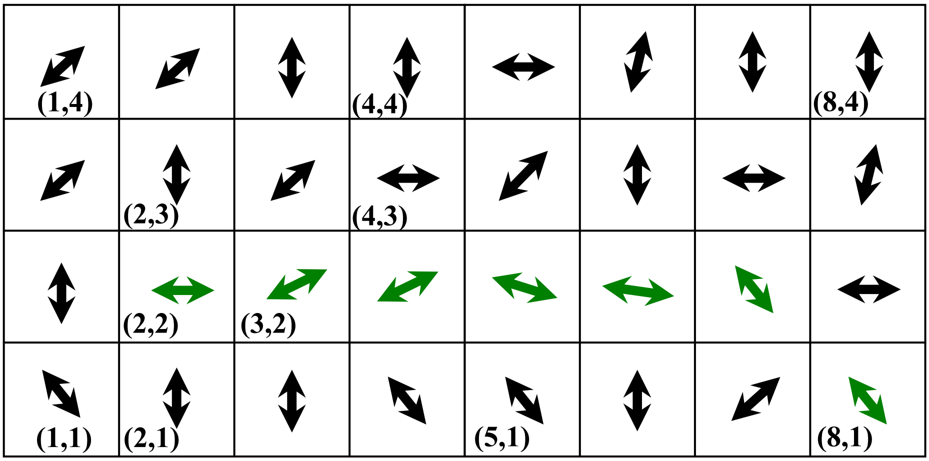


Fig. 4. For the same ROI, changing the threshold function results a non-branching trajectory

The real data sets of brain diffusion MR images are used for the validation of the algorithm. The starting point and the activation function's threshold are selected upon the knowledge of white matter fiber atlas and the most common pathology regions in the literature.

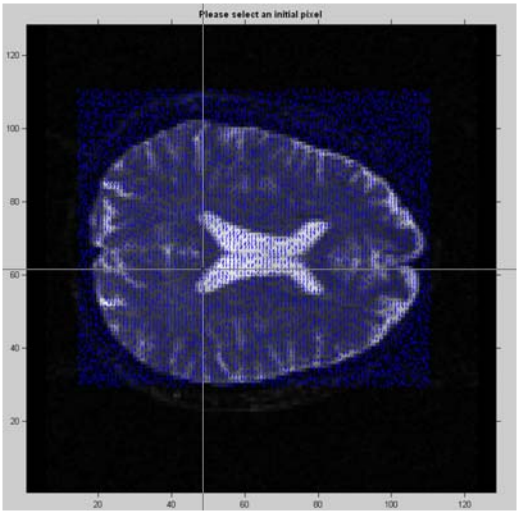


Fig. 5. Eigenvectors of the whole slice is represented on the registered anatomic MR image. The manually executed seed point selection is seen on the figure.

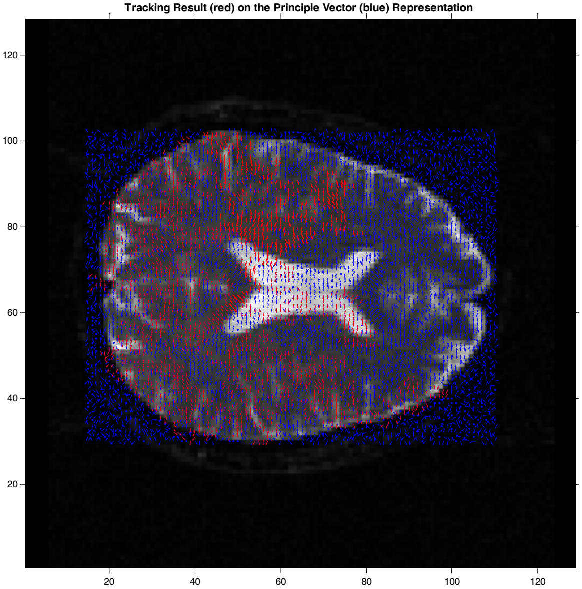

Tracking Result (red) on the Principle Vector (blue) Representation
120
100
80
60
40
20
20
40
60
80
100
120

Fig. 6. Implementation of the algorithm at the starting point shown in Fig. 5 with a threshold function σ_1 allowing a wide range of neighbors as winning nodes results the represented axial slice registered with the anatomic MR image

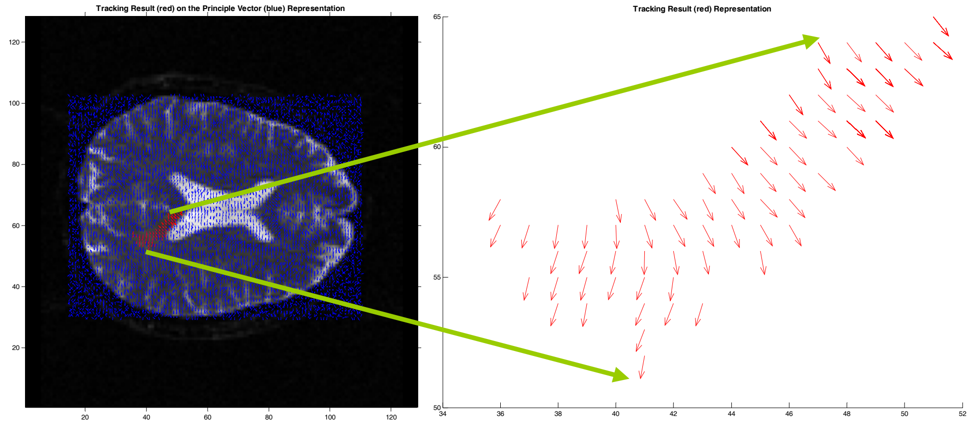

Tracking Result (red) on the Principle Vector (blue) Representation
120
100
80
60
40
20
20
40
60
80
100
120
Tracking Result (red) Representation
65
60
55
50
34
36
38
40
42
44
46
48
50
52

Fig. 7. Implementation of the algorithm with an activation function having a threshold σ_2 allowing a limited number of neighbors of the seed point shown in Fig. 5 as winning nodes

4. Discussion

The special class of artificial neural networks namely Hebbian learning is proposed for the analysis of a relatively new 3D imaging technique DTMRI's raw data. One of the major problems in DTI literature is the absence of a gold standard in fiber tractography. Intersections of two or more fiber tracts yield to erroneous estimates of the diffusivity and the fiber orientation. The anatomical fiber maps determined by diffusion tensor analysis are having still an unclear accuracy because of the inefficiency of the tensor model used to define the uncertainty regions such as crossing diffusivities in a single voxel. The practical accuracy of DT analysis and tractography vary upon the limitations of data quality and signal-to-noise ratio (SNR) (Mangin et al., 2002). In this proposed study the critical uncertainty problem is tried to be eliminated by adequate analysis tool. The method is first implemented on synthetic tract pattern (Fig. 2). The weight modifications yield to determine the weighted connections between the neighboring pixels (Fig. 3). The application results give promising tract estimations (Fig. 4) based on the threshold determination of the activation functions. Some real data analyses are done as represented in the implementation section 3.3 in Fig. 6 and Fig. 7. Still the proposed rule has to be implemented on 3D brain volume for validation studies.

Post processing reconstruction can reduce the sensitivity of tractography, so in Hebb application automated mapping and tracking after seed point selection is achieved and the method relies in basic learning algorithm which is quite an accepted procedure in defining the anatomical brain mapping. The applicability of the Hebbian rule to the uncertainty problem is verified by examining the updated weight changes by defining a fiber path.

The assumptions made in the determination of the diffusion tensor analysis are of great importance because the error tolerance and the general limitations of all the sequent applications including tractography are highly dependent on these.

5. Conclusion

Diffusion tensor imaging analysis and related tractography is highly promising for the detection and identification of brain fiber tracts especially the white matter paths. The implementation is helpful for better understanding of anatomical and pathological brain maps, the neural connectivity, neuropsychiatric diseases and the neural circuitry. The diffusion anisotropy in biological tissues still needs clarification on the fundamentals of its mechanisms. DTI is able to locate the intersecting fiber tracts but poor in identifying them, therefore post processing methods are of great importance (Lee, 2005; LeBihan, 2006). To solve the determination and visualization problems of fiber tracking especially in uncertainty regions, methods should be developed. The applied algorithm based on Hebbian learning is proposed to eliminate this uncertainty in intersecting pixels, and to define the fiber paths more secure and precisely upon the updated weightings.

The choice of training patterns plays a significant role in implementing the Hebb rule. The Hebbian approach aims basically the elimination of the impairment of the tensor modeling and the correlation of the brain fiber mapping to artificial neural network. The proposed method shows promising results to modify the fiber tractography and estimate the voxel diffusivity and neural map. The study must be accomplished in 3D human DTT. The assumptions made during the analysis need to be updated with a radiologist not to miss the clinical needs and to eliminate erroneous pathological approaches. Also engineering and clinical views should be obtained on the tractography results and these should be verified with brain atlases. The currently promising Hebb's rule will then be a qualified tool in DTMR analysis. Besides post processing enhancements, improvements will also be made with the development of faster sequences and higher field imaging.

6. References

Arfanakis, K.; Cordes, D.; Haughton, V.M.; Carew, J.D.; Meyerand, M.E. (2002). Independent component analysis applied to diffusion tensor MRI. *Magn Reson Med,* 47, 2, (Feb. 2002) 354-63, 0740-3194

Bammer, R. (2003). Basic principles of diffusion-weighted imaging. *European Journal of Radiology*, 45, 3, (March 2003) 169-184, 0720-048X

Berg, H.C. (1983). *Random Walks in Biology*, Princeton University Press, 0691000646, Princeton, NJ

Basser, P.J.; Le Bihan, D. (1992). Fiber orientation mapping in an anisotropic medium with NMR diffusion spectroscopy. *Proceedings of the 11th Annual Meeting of the SMRM*, pp. 1221, August 1992, Berlin

Basser, P.J.; Mattiello, J.; Le Bihan, D. (1994). MR diffusion tensor spectroscopy and imaging. *Biophysical Journal*, 66, 1, (January 1994) 259-267, 0006-3495

Basser, P.J.; Pajevic, S.; Pierpaoli, C.; Duda, J. & Aldroubi, A. (2000). In Vivo Fiber Tractography Using DT-MRI Data. *Magnetic Resonance in Medicine*, 44, 4, (October 2000) 625–632, 0740-3194

Basser, P.J. (2002). Relationships Between Diffusion Tensor and q-Space MRI. *Magnetic Resonance in Medicine*, 47, 2, (February 2002) 392–397, 0740-3194

Borisenko, A.I.; Tarapov, I.E. (1979). *Vector and Tensor Analysis With Applications*, Dover Publications, Inc., 0486638332, New York, NY

Callaghan, P.T.; Eccles, C.D. & Xia, Y. (1988). NMR microscopy displacements: k-space and q-space imaging. *Journal of Physics E: Sci. Instrum.*, 21, 8, (August 1988) 820-822, 0022-3735

Ciccarelli, O.; Parker, G.J.M.; Wheeler-Kingshott, C.A.M.; Barker, G.J.; Boulby, P.A.; Miller, D.H. & Thompson, A.J. (2003). From diffusion tractography to quantitative white matter tract measures: a reproducibility study. *Neurolmage*, 18, 2, (February 2003) 348-359, 1053-8119

Ding, Z.; Gore, J.C. & Anderson, A.W. (2003). Classification and quantification of neuronal fiber pathways using diffusion tensor MRI. *Magnetic Resonance in Medicine*, 49, 4, (April 2003) 716-721, 0740-3194

Fausett, L. (1994). *Fundamentals of Neural Networks,* Prentice Hall, 0-13-042250-9, New Jersey

Frank, L.R. (2002). Characterization of anisotropy in high angular resolution diffusion weighted MRI. *Magn. Res. Med.,* 47, 6, (June 2002) 1083-1099, 0740-3194

Goksel, D. & Ozkan, M. (2006). Towards rapid analysis of diffusion tensor MR imaging. *European Radiology*, 16, (Feb 2006) 1, 286, 0938-7994

Göksel, D. & Özkan, M. (2006). Simulated Diffusion Tensor MR Imaging (DT-MRI) Dataset for Verification and Validation of DT Analyzing Tools. *Proceeding Book of 5th International Conference on Advanced Engineering Design – AED 2006*, pp.122-129, 80-86059-44-8, Prague CTU, June 2006

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation,* Prentice Hall, 0-13-273350-1, New Jersey

Jeong-Won, J.; Singh, M. (2006). Multiple fibers per voxel detection by fast independent component analysis. *Proceedings of 3rd IEEE International Symposium on Biomedical Imaging ISBI,* pp.49-52, 0-7803-9576-X, Arlington Virginia, April 2006, IEEE Press

Jones, D.K. (2003). Determining and visualizing uncertainty in estimates of fiber orientation from diffusion tensor MRI. *Magn. Reson. Med.*, 49, 1, (January 2003) 7–12, 0740-3194

Khader, M.H. & Narayana, P.A. (2003). Computation of the Fractional Anisotropy and Mean Diffusivity Maps Without Tensor Decoding and Diagonalization: Theoretical Analysis and Validation. *Magnetic Resonance in Medicine*, 50, 3, (September 2003) 589–598, 0740-3194

Le Bihan, D.; Poupon, C.; Amadon, A. & Lethimonnier, F. (2006). Artifacts and Pitfalls in Diffusion MRI. *J. Magn. Res. Imaging*, 24, 3, (September 2006) 478–488, 1053-1807

Lee, S.K. (2005). Diffusion tensor MRI: clinical applications and pitfalls. *Fortschr Röntgenstr*, 34, pp.64, 1438-9029

Lori, N.F.; Akbudak, E.; Shimony, J.S. et.al. (2002). Diffusion Tensor fiber tracking of human brain connectivity: acquisition methods, reliability analysis and biological results. *NMR Biomed.*, 15, 7-8, (Nov-Dec 2002) 494-515, 0952-3480

Mangin, J.F.; Poupon, C.; Clark, C.; Le Bihan, D. & Bloch, I. (2003). Distortion correction and robust tensor estimation for MR diffusion imaging. *Medical Image Analysis*, 6, 3, (September 2002) 191-198, 1361-8415

Pajevic, S. & Pierpaoli, C. (2000). Color Schemes to Represent the Orientation of Anisotropic Tissues From Diffusion Tensor Data: Application to White Matter Fiber Tract Mapping in the Human Brain. *Magnetic Resonance in Medicine*, 42, 6, (June 2000) 526–540, 0740-3194

Poupon, C.; Clark, C.A.; Frouin, V.; Regis, J.; Bloch, I.; Le Bihan, D. & Mangin, J.F. (2000). Regularization of diffusion-based direction maps for the tracking of brain white matter fascicles. *Neurolmage*, 12, 2, (August 2000) 184-195, 1053-8119

Sungheon, K.; Jeong-Won, J.; Singh, M. (2005). Estimation of multiple fiber orientations from diffusion tensor MRI using independent component analysis. *IEEE Transactions on Nuclear Science*, 52, 1, (Feb. 2005) 266-273, 0018-9499

Szafer, A.; Zhong, J.; Anderson, A.W.; Gore, J.C. (1995). Diffusion-weighted imaging in tissues: theoretical models. *NMR in Biomedicine*, 8, 7-8, (Nov-Dec 1995) 289-96, 0952-3480

Taylor, W.D.; Hsu, E.; Krishnan, R.R. & MacFall, J.R. (2004). Diffusion tensor imaging: Background, potential, and utility in psychiatric research. *Biological Psychiatry*, 55, 3, (February 2004) 201-207, 0006-3223

Tuch, D.S.; Reese, T.G.; Wiegell, M.R.; Makris, N.; Belliveau, J.W.; Wedeen, V.J. (2002). High angular resolution diffusion imaging reveals intravoxel white matter fiber heterogeneity. *Magn Reson Med*, 48, 4, (October 2002) 577-82, 0740-3194

Westin, C.F.; Maier, S.E.; Khidhir, B.; Everett, P.; Jolesz, F.A. & Kikinis, R. (1999). Image Processing for Diffusion Tensor Magnetic Resonance Imaging, *Proceedings of Medical Image Computing and Computer-Assisted Intervention, Lecture Notes in Computer Science*, pp. 441-452, 3-540-66503-X, September 1999, Springer-Verlag, Cambridge

22

A Novel Credit Assignment to a Rule with Probabilistic State Transition

Wataru Uemura
Ryukoku University
Japan

1. Introduction

In this chapter, we introduce profit sharing method (Grefenstette, 1988) (Miyazaki et al., 1994a) which is a reinforcement learning method. Profit sharing can work well on the partially observable Markov decision process (POMDP) where a learning agent cannot distinguish an observation between states which need another action, because it is a typical non-bootstrap method, and its Q-value is usually handled accumulatively. So we study profit sharing as the next generation reinforcement learning system. First we discuss how to assign the credit to a rule on POMDP. The conventional reinforcement function of profit sharing does not consider POMDP. So we propose a novel credit assignment which considers the condition of the reward distribution on POMDP. Secondly, we discuss the probabilistic state transition on MDP. Profit sharing does not work well on the probabilistic state transition. We propose a novel learning method which considers the probabilistic state transition. It is similar to the Monte Carlo method. We therefore discuss the Q-values of our proposed method. In an environment with deterministic state transitions, we show the same performance for both conventional profit sharing and the proposed method. We also show the good performance of the proposed method against the conventional profit sharing.

In this chapter, we discuss the learning in POMDP and the probabilistic state transition. We show the advantages and disadvantages of the profit sharing method. We propose a novel learning method which has the same advantages and solves the disadvantages.

We propose how to handle the Q-values in an action-selection. Section 2 introduces the conventional reinforcement learning methods and profit sharing method. We propose the novel learning method in Section 3. Section 4 shows the results and finally Section 5 concludes this chapter.

2. Reinforcement learning system

In a reinforcement learning system (Sutton, 1990), a learning agent gets a reward if and only if it reaches the goal state. An agent learns a better policy by repeated trial and error. We just describe the goal condition, so an agent must learn how to go from the start state to the goal state by the interaction between an agent and an environment. At time t, an agent observes the **observation** o_t at the **state** s_t, and selects an **action** a_t by the **policy**. After selecting the action a_t, the environment will change from the state s_t to the next state s_{t+1}. When the next state s_{t+1} is the goal state, the agent gets the reward r_{t+1}, and if the next state s_{t+1} is not the goal state, the reward r_{t+1} will be equal to 0, or less than 0 which means the penalty.

In Markov decision process (**MDP**) (Sutton, 1990), the probability $P_{st,st+1}$, which is the state transition probability from the state s_t to the state s_{t+1} by the action a_t, is decided by only the state s_t and the action a_t. If an agent cannot get the all of the state, then some other states are observed with the same observation. We call this a partially observable Markov decision process (**POMDP**) (Miyazaki et. al, 1998) (Whitehead & Balland, 1990). In a POMDP environment, an agent must select two or more actions at the same observation.

2.1 Q-learning

We introduce Q-learning (Watkins & Dayan, 1992) which estimate the rule's value as a Q-value. The Q-value means the expected return which is updated as follow:

$$\begin{aligned} Q(s_t, a_t) \quad &\leftarrow \quad Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_{a_{t+1} \in A(s_{t+1})} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right], \\ &= \quad (1-\alpha)Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_{a_{t+1} \in A(s_{t+1})} Q(s_{t+1}, a_{t+1}) \right] \end{aligned} \tag{1}$$

where α is the learning rate, and γ is the discount rate. After many trials, Q-value will reach the estimate value of its rule. Thus, an agent selects the rule which has the highest Q-value of its state in order to get the optimal policy. Using Q-learning, an agent can update Q-value per action-selection without the reward. We call this **on-line updating**. Thus we can set the any value as initial Q-value. We call this **optimistic initial value**.

In a POMDP environment the combination of $Q(o_t,a_t)$ and $Q(o_{t+1},a_{t+1})$ is effected by alias problems, where the observation o_t and o_{t+1} means the observation at the state s_t and s_{t+1} respectively, so Q-value cannot reach the optimal value. For example, $Q(o_1,a_1)$ has the high-value at the state s_1, on the other hand $Q(o_1,a_1)$ has the low-value at the state s_2, then $Q(o_1,a_1)$ has no aim.

2.2 Profit sharing

In this section, we introduce profit sharing (Grefenstette, 1988) (Miyazaki et. al, 1994a) (Miyazaki et. al, 1994b) which is a reinforcement learning method. A profit sharing method has some advantages over other learning methods which mean that it can learn in MDP and also POMDP environments. In profit sharing, the agent distributes the reward to the selected rules (called an **episode**) when it reaches the goal state. The distributed function $f(x)$ is called a **reinforcement function**, and in MDP (Miyazaki et. al, 1994a) (Miyazaki et. al, 1994b) it should be formed by

$$C\sum_{j=i}^{W} f(j) < f(i-1) \qquad (i=1,\ldots,W), \tag{2}$$

where C is the maximum number of conflicting effective rules, and W is the maximum length of episodes. We usually use the reinforcement function that implements Equation 2 as follow:

$$f(x) = 1/L^x, \tag{3}$$

where x is the number of steps from the goal state, and L is the number of actions at each state.

3. Novel profit sharing

3.1 Reward distribution in a POMDP

Profit sharing uses the estimate value of rules in selecting rules. The estimate value does not be correct value when an aliasing observation confuses the agent observation capability. The conventional reinforcement function of profit sharing has this problem. The reinforcement of profit sharing is expressed by

$$\omega(o_x, a_x) \leftarrow \omega(o_x, a_x) + r \times f(x), \tag{4}$$

where o_x is the observation from the state s_x. In Equation 4, there is no problem because profit sharing does not use the relationship between observations. Profit sharing does not correctly estimate rules if and only if a rule (o_x,a_x) is equal to a rule ($o_{x'}$,$a_{x'}$), a state s_x is not equal to a state $s_{x'}$, and an action a_x is not equal to an action $a_{x'}$. We discuss this case.

The case of the problem in profit sharing is that an agent confuses between a reinforcement rule and a non-reinforcement one. For example, at Figure 1a an agent has to suppress the rule (s_{t1},a_j) than the rule (s_{t1},a_i). At Figure 1b an agent can not distinguish the state s_{t1} and the state s_{t2} from observation o (=o_{t1}=o_{t2}). If the agent suppresses the rule (o_{t1},a_j) than the rule (o_{t1},a_i) at the state s_{t1}, its suppression will reinforce the rule (o_{t2},a_i) to make a loop at the state s_{t2}. Both the rule (o,a_i) and the rule (o,a_j) at Figure 1b are needed to receive a reward and must not be suppressed. None of needed rules for a reward must be suppressed. On MDP it is needless for an agent to think of the rule suppression because there is not aliasing state (like Figure 1b). On POMDP it is need for an agent to think of the rule suppression. All rule for a reward should be reinforced equally. All rule in an episode should be reinforced equally at each state, because an agent can see no difference between Figure 1a and Figure 1b with one episode.

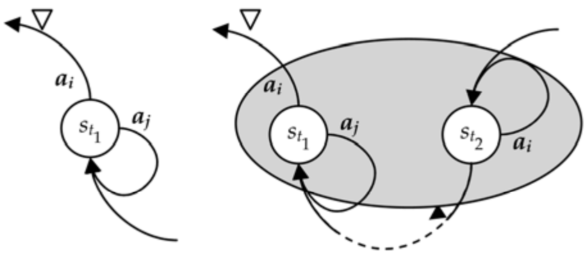
Thorem 1:

On POMDP the condition to distribute correctly the reward is

$$f(x) = \begin{cases} \alpha_{o_x} & \text{first reinforcement of rule x} \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where rule x is reinforced by the function $f(x)$. α_{ox} has to take the constant value at each observation o_x.

We propose the Episode-based Profit Sharing (EPS) that fills the need for the correct distribution on POMDP. The reinforcement function of EPS is



(a) A non-aliasing state. **(b) Aliasing states.**

Fig. 1. Aliasing states and a non-aliasing state.

$$f(x) = \begin{cases} 1/L^{w} & \text{first reinforcement of rule } x \\ 0 & \textit{otherwise,} \end{cases} \tag{6}$$

where L is the number of non-detour rules at a state, then the number of rule-1 is sufficient for L. We show that EPS can suppress the reinforcement of rules that make a loop.

If the environment has aliasing states, then the reinforcement function to distribute correctly rewards needs *Theorem* 1. The perceptual aliasing problem does not affect EPS because EPS can fill the needs from *Theorem* 1. So we have no need to think about the affectable of the aliasing states. We show the two case, one is that only one state makes a loop, and the other case is that multiple states make a loop. Next we propose the sub-episode method that reinforces rules with part of an episode. When part of an episode can be used always, the reinforcement function matches a geometrical decreasing function, that is the conventional function.

(a) a loop consisting of single state

Now we discuss the case that one observation makes a loop. The reinforcement value is written as Δ. The difference of reinforcement values between a non-detour rule and a detour rule is

$$\Delta(o, \text{non-detoure rule}) > \Delta(o, \text{detour rule}). \tag{7}$$

So EPS can suppress the reinforcement of rules that make a loop in the case of single state.

(b) a loop consisting of multiple states

Now we discuss the case that has two or more observations in order to make a loop. The difference of reinforcement values between a non-detour rule and a detour rule is

$$\Delta(o_l, \text{non-detoure rule}) > \Delta(o_l, \text{detour rule}). \tag{8}$$

EPS can suppress the reinforcement of rules that make a loop in the case of multiple states. So we can show the suppression proof of EPS.

(c) using part of an episode

We discuss about the sub-episodes (o_i,a_i), (o_{i+1},a_{i+1}), ..., (o_t,a_t) (i=1,2,...,t-1) which are the parts of an episode (o_1,a_1), (o_2,a_2), ..., (o_t,a_t). An agent can learn from the sub-episodes which start at the time i. To use sub-episodes has to fill the needs for *Theorem* 1 in order to distribute correctly rewards on POMDP. When an agent can see no difference between the observation o_{k1} and the observation o_{k2} affected by perceptual aliasing, there may be some difference between the state s_{k1} and the state s_{k2}. In this case, the agent can not use the sub-episode which has the rule (o_k,a_k) is the start rule in order to fill the needs for *Theorem* 1 ($k_1 < k \leq k_2$). That is to say that the agent can use the sub-episode starting at the rule (o_k,a_k) ($k \leq k_1$ or $k_2 < k$). It is the same when two or many observations are affected by perceptual aliasing. The rules between the observation o_{k1} and the observation o_{k2} are defined as rules *on an observation loop*. The flag to mean whether the rule (o_k,a_k) is on an observation loop or not is d_k which is defined as

$$d_k = \begin{cases} 0 & \text{ok is on an observation loop.} \\ 1 & \text{otherwise,} \end{cases} \tag{9}$$

An agent can reinforce rules using the length t-i+1 of the sub-episode (o_i,a_i), (o_{i+1},a_{i+1}), ..., (o_t,a_t). Now the amount $f(x)$ of reinforcement for rule (o_x,a_x) is

$$f(x) = \sum_{k=x}^{W} \frac{1}{L^k} d_k \cdot \tag{10}$$

Figure 2 shows this reuse sub-episodes. So the reinforcement function of EPS with sub-episodes is

$$f(x) = \begin{cases} \sum_{k=x}^{W} \frac{1}{L^k} d_k & \text{first reinforcement of rule } x \\ 0 & \textit{otherwise.} \end{cases} \tag{11}$$

The reinforcement function on MDP that is d_k=1 ($\forall k = 1,2,\ldots,W$) and has no same rules in an episode is

$$f(x) = \sum_{k=x}^{W} \frac{1}{L^k} \cdot \tag{12}$$

Given $W \rightarrow \infty$, the reinforcement function $f(x)$ becomes the geometrical decreasing function with a common ratio $1/L$. This function matches the conventional function.

3.2 Online updating

Usually softmax action selection is used for profit sharing because its Q-value means the accumulation of past rule values, for example, roulette distribution, Boltzmann distribution, and Gibbs distribution. In a POMDP environment, in some states, the agent cannot recognize that the observation there is not similar to the observation of another state. In other words, it gets the same observation in the other states. This problem is called an **alias problem** (Whitehead & Ballland, 1990).

Profit sharing is robust in a POMDP environment for two reasons. One is that updating the Q-value is non-bootstrapping. Non-bootstrapping means that the agent does not use Q-values which are in other states in order to estimate the Q-value. Updating the equation for profit sharing is as follows:

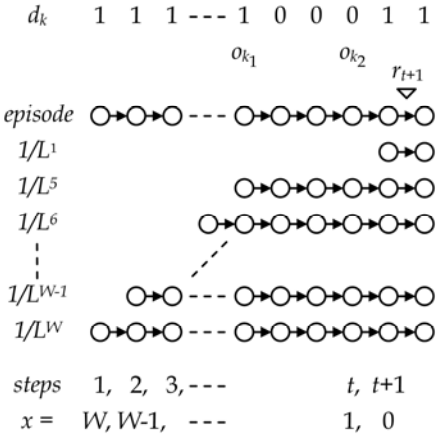


Fig. 2. Reuse sub-episodes (when $o_{k_1} = o_{k_2}$).

$$\omega(s_x, a_x) \leftarrow \omega(s_x, a_x) + f(x), \quad \text{For all value of } x \text{ in the episode.} \tag{13}$$

where $\omega(s_x,\ a_x)$ is the Q-value of the rule (s_x,a_x). Equation 13 was proposed by Miyazaki (Miyazaki et. al, 1994a) (Miyazaki et. al, 1994b). This updating euqation does not require the Q-value of another rule. So profit sharing is a non-bootstrapping method.

The second reason is that the action selection of profit sharing is a softmax action selection. In order to solve the alias problem, the agent must not select one action always often one observation because due to the alias problem the agent must select two or more actions. For example, in the state s_{t1} the agent gets the observation o $(=o_{t1})$, and the action which brings the agent near to the goal state is action a_i (shown at Figure 1b). On the other hand, in the state s_{t2} the agent gets the same observation o $(=o_{t2})$, however the action which bring the agent near to the next state is action a_j. Thus, the agent should not select one action for the one observation o. The agent must select both two actions, a_i and a_j, at the one observation o.

The conventional reinforcement learning methods (Watkins & Dayan, 1992) uses greedy action selection. When the action selection is greedy action selection, the agent can select the rule which has the highest Q-value of its state. Using this select method, a rule which has a secondary high Q-value is never selected. Thus the conventional reinforcement learning method does not work well in a POMDP environment. In an MDP environment, there is no aliasing states (shown in Figure 1a). So greedy action selection can work well. Using Equation 14 proposed by Miyazaki (Miyazaki et. al, 1994a) (Miyazaki et. al, 1994b) (called **accumulative profit sharing**), the agent can select two or more actions at the same observation. So accumulative profit sharing is robust in a POMDP environment.

Accumulative profit sharing, however, does not consider the probability of the state transition (Uemura et. al, 2007). For example, it distributes the same rewards whatever the state transition probability is. The expected value means $R \times P$, where R is the reward and P is the transition probability. So we should make the distributed reward nearly equal to its expected value.

A reinforcement function cannot know the state transition probability because many trials are needed to find it. Thus it is too difficult to estimate the rule-transition probability using only one episode. Some conventional reinforcement learning methods work per action selection, where the agent can update Q-values.

We propose a novel credit assignment method which considers the probabilistic state transition. Accumulative profit sharing does not consider the number of selection in the same rule. This method, therefore, distributes the same credit assignment to the rules which got the same rewards but have a different probabilistic state transition.

So we must count the number of selections in the same action, and discount the Q-value. The novel Q-value is as follows:

$$Q(s,a) \leftarrow N_r(s,a)/N_a(s,a) \times \omega(s,a), \tag{14}$$

where $N_r(s,a)$ is the number of rewards by the rule (s,a), and $N_a(s,a)$ is the number of selections of the rule (s,a).

If the state transition of rule (s,a) is always deterministic, then the number of rewards obtained $N_r(s,a)$ is almost equal to the number of selections of the rule $N_a(s,a)$. If and only if the episode has a loop, $N_a(s,a)$ becomes larger than $N_r(s,a)$. If the rule (s,a) has the probabilistic state transition, $N_r(s,a)$ / $N_a(s,a)$ means an estimated value. In other words, $N_r(s,a)$ / $N_r(s,a)$ means the experiential rule transitional probability under its learning procedure.

For example, the conventional Monte Carlo method uses the average estimate value. Its estimating function is as follows:

$$Q(s,a) \leftarrow N_r(s,a)/N_a(s,a), \tag{15}$$

This equation brings the $Q(s,a)$ to the average of rewards. This, however, is not accumulative. Thus the Monte Carlo method requires greedy action selection. Our proposed method accumulates the rewards. Thus it requires softmax action selection. It is also robust for the POMDP environment. We call our proposed method the **accumulative Monte Carlo method**.

4. Experiment

4.1 Reward distribution in a POMDP

An agent cannot know how many states affected perceptual aliasing on POMDP. So we prepare the experimental environment which has aliasing states by half of all (Figure 3). Agent can select one action from four actions (up, down, left and right) at each state. If the direction of the selected action is equal to one of the arrow in the figure, then the agent moves to the next state. The observation o_1 is observed at the state s_1, s_2, and s_3. The agent should select randomly one action from three actions except for left action because the agent must select right, down, and up at each state. At the state s_4, s_5, and s_6, the agent has to learn the action moving to the next state because the observations are equal to the states. The performance means received rewards per number of the selected actions, and the performance by the optimum policy is 10/12 = 0.833.

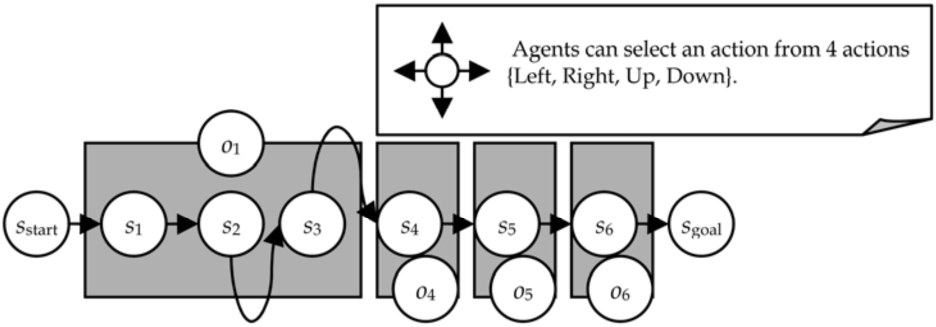


Fig. 3. The Experimental Environment in a POMDP.

Figure 4 shows the result. The action selection of Q-Learning is ε-greedy which selects the maximum Q-value in 90% probability and random actions in 10% probability. The conventional profit sharing with the geometric decreasing function is written as *PS (Decrease)*. The performance of PS (Decrease) becomes worse but the proposed profit sharing, EPS, can learn more policy.

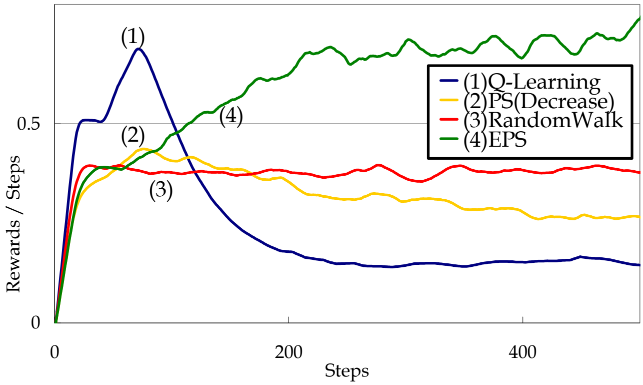


Fig. 4. The learning performances at POMDP.

4.2 Online updating

We carried out experiments in a maze (Sutton, 1998) (Figure 5). An agent starts at state *S* and selects one action from 4 actions (up, down, left and right). When the agent reaches the goal state *G*, the reward *R* = 10 is received, and the agent restarts at the start state *S*. The performance is how many rewards to get per step. All actions have the same probabilistic state transition. The agent goes to the selected state by the probability *P* = 0.8, and goes to the neighbour state by the probability *P* = 0.2.

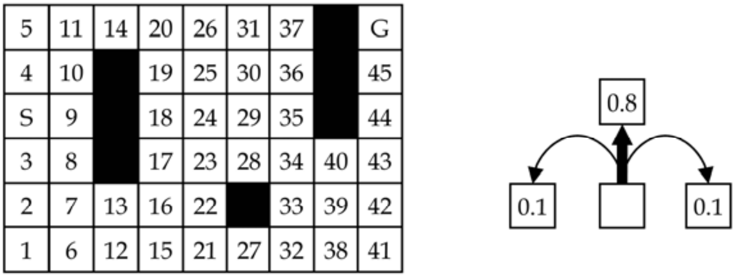


Fig. 5. The maze of Sutton with probabilistic state transitions.

The proposed method has almost the same performance as the conventional method in the non-probabilistic state transition. There is a difference if and only if the agent makes a loop in the early stage of learning. So in the first learning steps, the proposed method distributes slightly less rewards than conventional profit sharing. The performance for the probabilistic state transition is shown in Figure 6. The proposed method has better performance than the conventional method.

5. Conclusion

In this chapter, we have proposed a novel credit assignment method similar to profit sharing which considers the aliasing problem and the probabilistic state transition. We show that the condition to learn in a POMDP is to distribute equal rewards to rules at the same

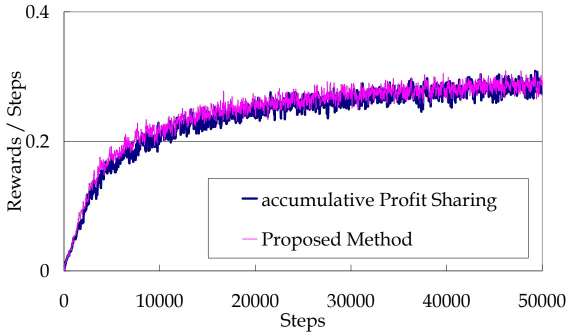


Fig. 6. The performance between conventional method and the proposed method.

state in an episode. We proposed a novel reward distribution method, called EPS, which considers this condition. Next, we consider the probabilistic state transition. If the agent experiences the same rule as the previous episode, the current episode has a loop rule, that is, its rule has a probabilistic state transition. So its rule value should be less than the previous reward. The equation $R \times P$, where R is the reward and P is the transition probability, shows the expected value. Thus the temporary rule variable should be divided by the number of its rule selection. Finally the temporary rule variable reaches to its expected value. We have proposed how to decrease the estimated values of rules per action selection.

In an environment with a deterministic state transition, we show the same performance for both conventional profit sharing and the proposed profit sharing. And we show the good performance of proposed profit sharing against the conventional profit sharing with a probabilistic state transition.

6. References

Arai, T. & Kragic, D. (1999). Name of paper, In: *Name of Book in Italics,* Name(s) of Editor(s), (Ed.), page numbers (first-last), Publisher, ISBN, Place of publication

Grefenstette, J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, Vol. 3, pp. 225 – 243.

Miyazaki, K., Yamamura, M. and Kobayashi, S. (1994). A Theory of Profit Sharing in Reinforcement Learning. *Journal of Japanese Society for Artificial Intelligence*, Vol. 9, No. 4, pp. 104 – 111.

Miyazaki, K., Yamamura, M. and Kobayashi, S. (1994). On the Rationality of Profit Sharing in Reinforcement Learning. *Proceedings of the 3rd International Conference on Fuzzy Logic*, Neural Nets and Soft Computing, pp. 285 – 288.

Miyazaki, K. and Kobayashi, S. (1998). Learning Deterministic Policies in Partially Observable Markov Decision Processes. *Proceedings of International Conference on Intelligent Autonomous System 5*, pp. 250 – 257.

Sutton, R. (1990). Integrated Architecture for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the 7th International Conference on Machine Learning*, pp. 216 – 224.

Sutton, R. (1998). Reinforcement learning – an introduction – , *the MIT Press*.

Uemura, W., Ueno A. and Tatsumi, S. (2004). The exploitation reinforcement learning method on POMDPs. *in Joint 2nd International Conference on Soft Computing and Intelligent Systems*, TUE – 1 – 3.

Uemura, W. (2007). About distributing rewards to a rule with probabilistic state transition, *in the SICE Annual Conference 2007, International Conference on Instrumentation, Control and Information Technology*, pp. 2762 – 2765.

Watkins, C. and Dayan, P. (1992). Technical note: Q-Learning, *Machine Learning*, Vol. 8, pp.279 – 292.

Whitehead, S. and Balland, D. (1990). Active perception and reinforcement learning, *Proceedings of the 7th International Conference on Machine Learning*, pp.162 – 169.