6.047 / 6.878 Computational Biology: Genomes, Networks, Evolution
Fall 2008

6.047/6.878 Computational Biology September 29, 2008

Lecture 8: Computational Gene Prediction and GHMMs

Lecturer: James E. Galagan

1 Overview of gene prediction

One of the fundamental problems in computational biology identification of genes in very long genome sequences. As we know DNA is a sequence of nucleotide molecules, or bases, which encode instructions for generation of proteins. However, not all of these bases correspond directly to amino acids. Even within a gene, a relatively small percentage of the nucleotides might actually be translated into an amino acid chain.

For example, Eukaryotic genomes contain introns, or long segments of non-coding nucleotides within a gene; the introns are discarded during processing into mature RNA leaving only the exons linked together to contribute to protein manufacturing. Also, both Prokaryotic and Eukaryotic genomes contain substantial intergenic regions, and there are many other types of segments such as introns, start and stop codons, etc. which do not code for proteins directly, but are crucial to the protein synthesis in other ways. The contiguous subsequence of bases that is finally parsed out of DNA during processing into RNA is called the coding sequence, and is comprised of the exons with all introns removed. This sequence is then deterministically translated into amino acids. Our task in gene prediction (or genome annotation) is:

Given a DNA sequence (X) with zero or more genes and some "evidence" associated with the given DNA sequence, determine a labeling (Y) that assigns to each base a label according to the functionality of that part of the gene. Some of the most basic labels are:

Intron Non-coding regions within a gene; removed during processing into mature RNA.

Exon Coding regions. 3-letter words, or codons, correspond directly to amino acids.

Intergenic Region Non-coding region located between two genes.

Start/Stop Codons Specific sequences appearing at the beginning and end of all genes.

Acceptor Site Specific sequence appearing at the end of an intron / the beginning of an exon.

Donor Site Specific sequence appearing at the beginning of an intron / the end of an exon.

With these labels, we can think of gene prediction as parsing a sequence of letters into words. In this analogy, the different regions of a DNA sequence are similar to different types of words (nouns, verbs, etc.), and just as we must follow grammatical rules when constructing sentences, there are syntactical rules for parsing DNA as well. For example, we only allow introns to occur within genes, and we expect to find start and stop codons before and after genes occur. There are likely to be many "valid" parses, and in general we are searching for the most probable legal parse of the sequence.

2 Gene prediction methods

How do we annotate genes? We would like to consider all evidence available to make the most accurate annotation possible. We can classify our evidence into two main categories: intrinsic and extrinsic. Intrinsic evidence is information gathered directly from the nucleotide sequence, or *features* of the sequence. This includes short significant subsequences like start codons, stop codons, a TATA box, etc. called *signals*, along with *content regions*, such as general statistics about the relative levels of the four nucleotide bases in a region. For example, in exon regions, all codons are not used equally, and we can translate codon frequency statistics into nucleotide frequency information. In intergenic regions or introns, there is a fundamental difference in nucleotide frequencies compared to exons resulting from the difference in evolutionary pressure between the two regions. All of this evidence must be taken in context statistically; for example, *features* are short sequences which will appear randomly throughout DNA sequences, so the presence of a start codon by itself isn't strong evidence. However, finding combinations of *signals* such as:

Start Codon → Acceptor Site → Donor Site → Stop Codon

constitues stronger evidence because it makes sense syntactically, whereas

Acceptor Site → Stop Codon → Start Codon → Donor Site

does not, as it isn't a valid parse. Other examples of measures we can incorporate are the lack of stop codons in the middle of exons, and the length of the regions (exons tend to be smaller, around 200 bases, introns can be serveral kbps in length). Apart from the evidence intrinsic to the DNA sequence, we can also incorporate some extrinsic evidence, like direct experimental evidence, BLAST hits, and the significance of each BLAST hit, HMMer hits, alignments from sequences of actual RNA to determine exons vs. introns, etc. The main question of gene prediction algorithms is how to combine all of this evidence together.

3 HMMs in Gene Prediction

The most popular method so far of combining evidence is the HMM (Hidden Markov Models). In this probabilistic model, we represent the labels as hidden states which constitute a Markov chain. For each pair of base x_i and label y_j, we also assign an emission probability to model the probability of observing base x_i when the hidden state is y_j. We denote this as $e_{y_j}(x_i)$. An important feature here is that we can ensure that the model produces only legal parses of a genome by assigng transition probabilities between states a_{jk}, representing the probability of transitioning into state k from state j. If $k = intron$ and $j = intergenic$, then it is clear that $a_{jk} = 0$, as we cannot move from an intron region into an intergenic regions.

This model is also called a generative model, since a convenient way to think of the HMM is to imagine that there is a machine with a button. And by pushing the button, one can generate a genome sequence according to some probability distribution. In gene prediction, we use maximum-likelihood training to compute state transition probabilities and emission probabilities, and then find the most likely hidden state sequence $Y = y_1 \dots y_n$. Note that

HMM in fact models a *joint distribution* over bases and hidden states, namely $P(X, Y) = P(\text{Labels}, \text{Sequence})$.

However, in gene prediction problems, we are given X and need to predict Y. This process clearly calls for a Dynamic Programming solution, namely the Viterbi algorithm. While HMMs have been applied to the gene prediction problem, there are some assumptions implicit to the model which are difficult to reconcile with real data. The most troubling problem is the distribution of state durations. For any state y_j of a HMM, we have a probability p of staying in state y_j, and probability $(1-p)$ of transitioning to another state. This gives us a geometric distribution for the state duration of y_j, described by the discrete distribution $P(D = d) = p^d \cdot (1-p)$. The problem arises from the fact that region lengths in DNA often do not resemble the geometric distribution. Another limitation of the HMM model is that we think of each state emitting a single nucleotide.

4 Introduction of Generalized Hidden Markov Models

To address these issues, we find it desirable to think of a Generalized Hidden Markov Model (GHMM) which allows us to explicitly specify several important aspects which were implicit in HMMs. We will allows states in the GHMM to emit arbitrary length sequences, and we will allow the emission probabilities to be an arbitrary function as well. Finally, we will specify transition probabilities explicitly so that they may be, for example, a function of the current length of the sequence. This allows us to model state durations according to experimental observations more realistically, resulting in higher accuracy parses. A GHMM consists of the following components which are *similar* to HMMs:

- States **Q**
- Observations **V**
- Initial State Probabilities $\pi_i = P(a_0 = i)$

and with the following *differences:*

- Duration Probabilities $f_k(d) = P(\text{state } k \text{ of length } D)$
- Emission Probabilities $e_k(X_{a,a+d}) = P_k(X_a \dots X_{a+d} | q_k, d)$

We wish to apply GHMMs again to the problem of gene annotation. Therefore, GHMMs must return the probability of a subsequence given a state and duration. Thinking about GHMMs in this way allows us to consider the set of states more abstractly without worrying about how the arbitrary functions are implemented. They could consist of anything capable of returning a probability distribution, i.e. a function, a properly set up neural network, etc.

Generalizing HMMs into this framework, however, increases the complexity of the problem of computing optimal paths such that it isn't feasible to run a Viterbi algorithm in the most general sense. The cause for the increase in the complexity of the problem arises from computing the optimal value for step i. We now must consider not only the optimal state computed for $i-1$, but also further back, as the optimization algorithm must now consider that we could have changed states just now... or we may just now realize that we changed states 4 positions ago.

This leads to a more complex recursion algorithm for the Viterbi algoritm on GHMMs. The recursion for a standard HMM and for a GHMM are both presented below for comparison:

HMM: $V_k(i) = e_k(x_i) \cdot \max_j [V_j(i-1) \cdot a_{jk}]$

GHMM: $V_k(i) = \max_j \max_d [P(x_i \dots x_{i-d}|k) \cdot V_j(i-d) \cdot P(d|k) \cdot a_{jk}]$

The GHMM recursion is more complex because we must take the maximum of both all previous states j, **and** all previous durations d. Inside the recursion the GHMM formula is straightforward:

- $P(x_i \dots x_{i-d}|k)$: Probability of seeing the sequence $(x_i \dots x_{i-d})$ of length d in state k.
- $V_j(i-d)$: Maximum probabilitiy of ending in state j at step $i-d$
- $P(d|k)$: Probability that state k has duration d.
- a_{jk} : Probability of a transition from state j to k.

Whereas the running time of the Viterbi algorithm on HMMs is $O(K^2L)$, the running time on GHMMs is much worse, $O(K^2L^3)$. The dependence on length turns out to make this algorithm unfeasible on gene sequences on the scale of the human genome.

5 GENSCAN

To address this concern and to make GHMMs useful, many implementations have been created that realize algorithmic speed-ups by selecting clever implementations of the arbitrary functions, or by making some other assumptions that help to limit the size of the expanded search space. The lecture focused on work by Chris Burge in his PhD thesis [2] on a particular implementation, called GENSCAN. This algorithm provided a good balance between capturing the complexity of the model, yet made a few key assumptions to reduce the algorithmic run time.

Some key features of GENSCAN include:

- Explicitly defined state duration probabilities
- Use a 5th order markov model to predict coding and non-coding sequences. (Look at the previous 5 nucleotides when predicting the next one.)
- Keep track of the frame of the coding sequence. When an intron interrupts a sequence, the model keeps track of the phase.
- Search both the 5' $\rightarrow$ 3' and 3' $\rightarrow$ 5' strand at the same time.
- Considers different types of exons, i.e. *initial exons* which follow the start codon, *internal exons* located between an acceptor site and a donor site, *final exons* preceeding the stop codon, or *single exons* which aren't interrupted by any introns.

- Create two main state types, **C** states and **D** states. The model is constructed so that we alternate between **C** and **D** states with each transition.

The most important aspect of GENSCAN, which provides algorithmic simplification we require to acheive acceptable running times is the concept of **C** and **D** states. This has two effects. First it reduces how far into the past we need to look. It also specifies that **D** states must have a length distribution with properties similar to the length distributions given by a HMM. Specifically, the difference between the probability of a **D** state having duration d vs. $d-1$ is given by a constant factor p_k. Therefore:

$P(\text{Length}= d|\mathbf{D} \text{ state}) = P(\text{Length}= d-1|\mathbf{D} \text{ state}) \cdot p_k.$

$f_k(d) = P(\text{state duration } d \text{ — state } k) = p_k \cdot f_k(d-1)$

This forces us to use geometric-like distributions, which we can get away with for intergenic regions, introns, UTRs, etc. However, for regions with state duration distributions radically different from the geometric distribution, or **C** states, GENSCAN specifies explicit formulas that capture these semantics.

Note: This is the draft of the notes for this lecture. The final version will include a more thorough consideration of the algorithmic details of GENSCAN.

References

[1] James Galagan. Conditional Random Fields for Computational Gene Prediction. Lecture note 6 of MIT 6.047/6.878 Computational Biology, Sept. 2007.

[2] Burge, C., Karlin, S. Prediction of complete gene structures in human genomic DNA. http://ai.stanford.edu/~serafim/cs262/Papers/GENSCAN.pdf