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

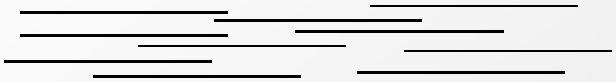
Modeling Biological Sequence using Hidden Markov Models

Lecture 6

Sept 23, 2008

Challenges in Computational Biology

4 Genome Assembly



9 Regulatory motif discovery

6 **Gene Finding**

DNA

2 Sequence alignment

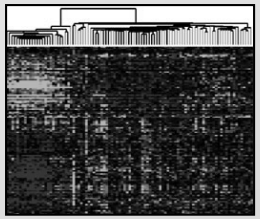
14 Comparative Genomics

10 Evolutionary Theory

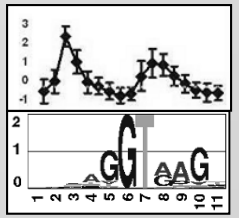
3 Database search

5 Gene expression analysis

RNA transcript

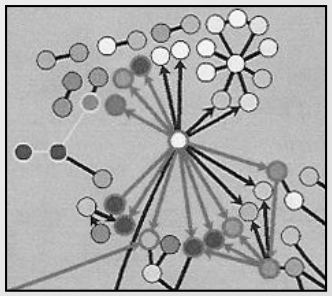


4 Cluster discovery



9 Gibbs sampling

11 Protein network analysis



12 Regulatory network inference

13 Emerging network properties

What have we learned so far?

- String searching and counting
  - Brute-force algorithm
  - W-mer indexing
- Sequence alignment
  - Dynamic programming, duality path ⇔ alignment
  - Global / local alignment, general gap penalties
- String comparison
  - Exact string match, semi-numerical matching
- Rapid database search
  - Exact matching: Hashing, BLAST
  - Inexact matching: neighborhood search, projections
- Problem set 1

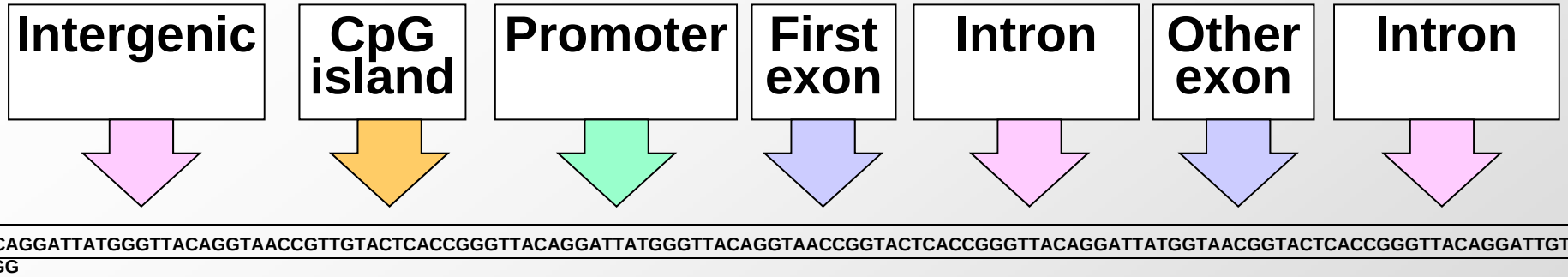
So, you find a new piece of DNA…

What do you do?

…GTACTCACCGGGTTACAGGATTATGGGTTACAGGTAACCGTT…

- Align it to things we know about
- Align it to things we don't know about
- Stare at it
  - Non-standard nucleotide composition?
  - Interesting k-mer frequencies?
  - Recurring patterns?
- Model it
  - Make some hypotheses about it
  - Build a 'generative model' to describe it
  - Find sequences of similar ***type***

This week: Modeling biological sequences
(a.k.a. What to do with a huge chunk of DNA)



- Ability to **emit** DNA sequences of a certain ***type***
  - Not exact alignment to previously known gene
  - Preserving ‘properties’ of **type**, not identical sequence
- Ability to **recognize** DNA sequences of a certain type (state)
  - What (hidden) state is most likely to have generated observations
  - Find set of states and transitions that generated a long sequence
- Ability to **learn** distinguishing characteristics of each state
  - Training our generative models on large datasets
  - Learn to classify unlabelled data

Why Probabilistic Sequence Modeling?

- Biological data is noisy
- Probability provides a calculus for manipulating models
- Not limited to yes/no answers – can provide “degrees of belief”
- Many common computational tools based on probabilistic models
- Our tools:
  - Markov Chains and Hidden Markov Models (HMMs)

Definition: Markov Chain

Definition: A ***Markov chain*** is a triplet **(*Q, p, A*)**, where:

- ***Q*** is a finite set of states. Each state corresponds to a symbol in the alphabet Σ
- ***p*** is the initial state probabilities.
- ***A*** is the state transition probabilities, denoted by a_{st} for each ***s, t*** in ***Q*.**
- For each ***s, t*** in ***Q*** the transition probability is: $a_{st} \equiv P(x_i = t | x_{i-1} = s)$

Output: The output of the model is the set of states at each instant time => the set of states are observable

Property: The probability of each symbol x_i depends only on the value of the preceding symbol x_{i-1} : $P(x_i \mid x_{i-1}, \ldots, x_1) = P(x_i \mid x_{i-1})$

Formula: The probability of the sequence:

$$P(x) = P(x_L, x_{L-1}, \ldots, x_1) = P(x_L \mid x_{L-1})\, P(x_{L-1} \mid x_{L-2}) \ldots P(x_2 \mid x_1)\, P(x_1)$$

Definitions: HMM (Hidden Markov Model)

Definition: An ***HMM*** is a 5-tuple **(*Q, V, p, A, E*)**, where:

- ***Q*** is a finite set of states, **|Q|=N**
- **V** is a finite set of observation symbols per state, **|V|=M**
- ***p*** is the initial state probabilities.
- ***A*** is the state transition probabilities, denoted by a_{st} for each ***s, t*** in ***Q*.**
  - For each ***s, t*** in ***Q*** the transition probability is: $a_{st} \equiv P(x_i = t | x_{i-1} = s)$
- **E** is a probability emission matrix, $e_{sk} \equiv P$ **(**v_k **at time** t **|** $q_t = s$**)**

Output: Only emitted symbols are observable by the system but not the underlying random walk between states -> "hidden"

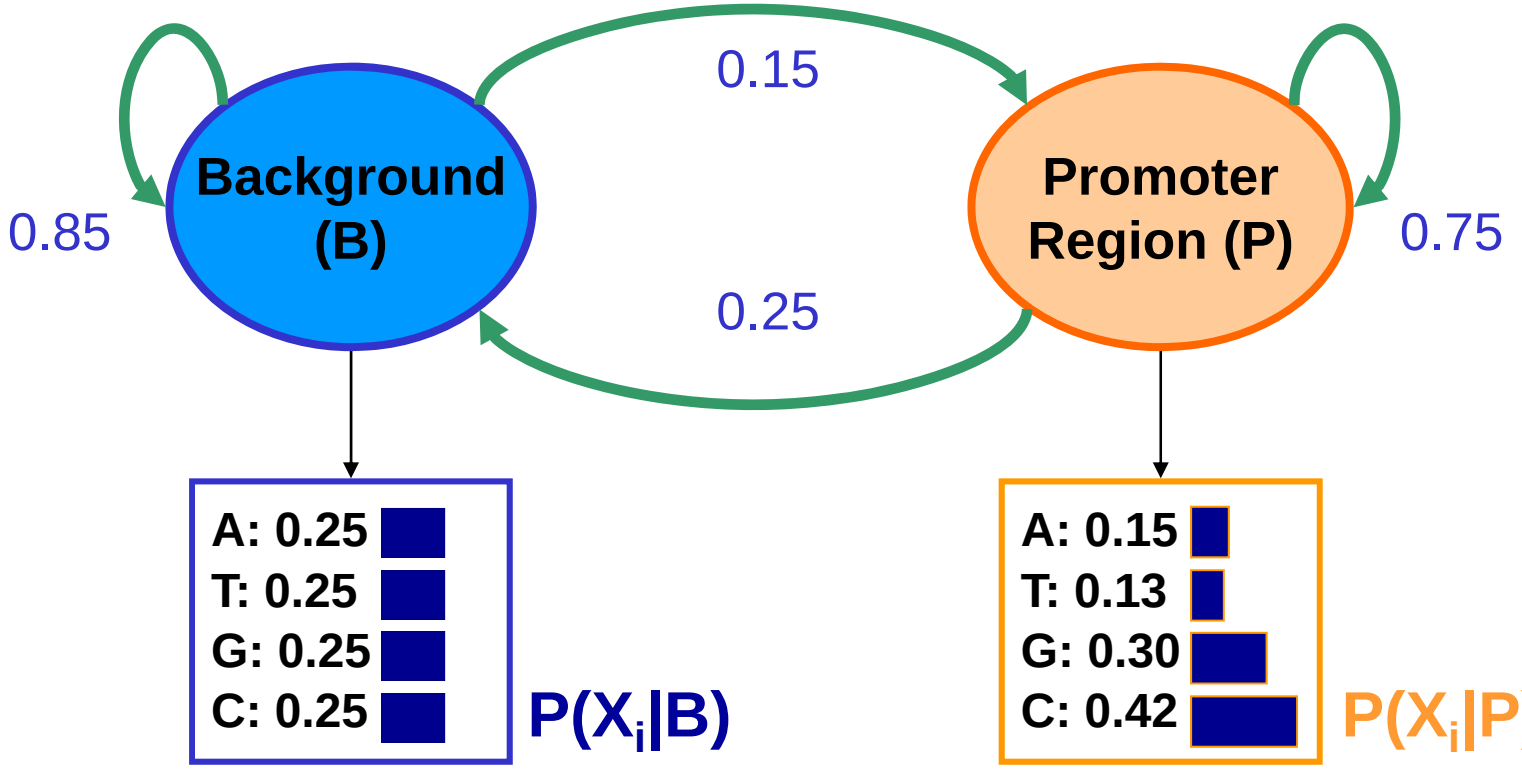
Property: Emissions and transitions are dependent on the current state only and not on the past.

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^{\wedge} = \{\pi_i \mid \pi_i = \text{argmax}_k \Sigma_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi \mid \Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi \mid \Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \Sigma_{\pi} P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

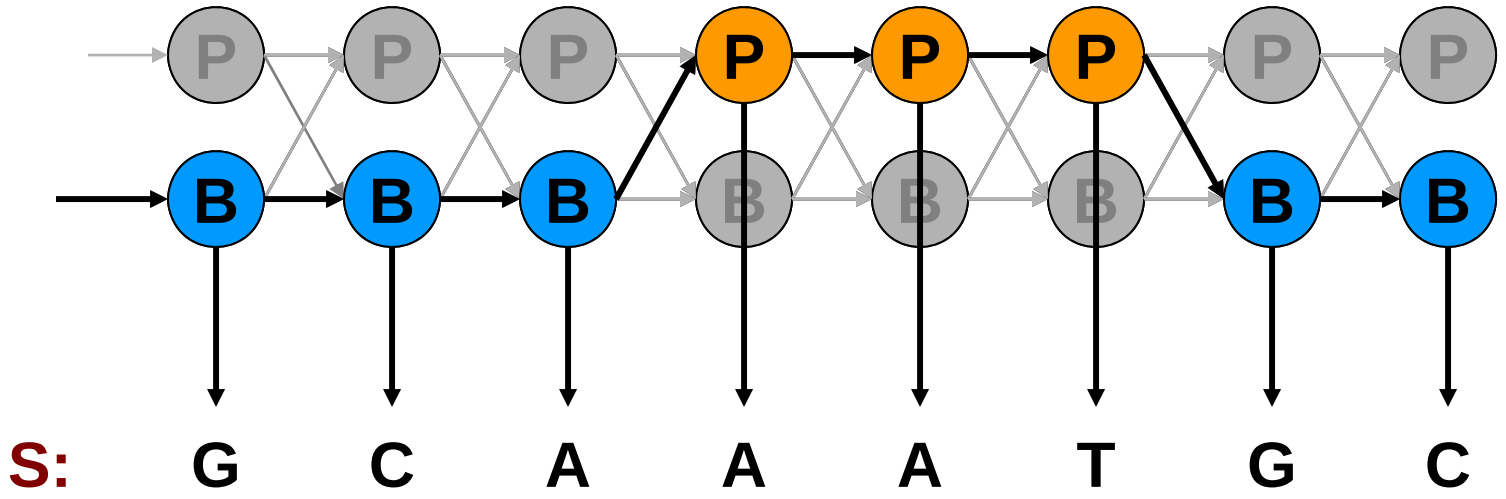
Example 1: Finding GC-rich regions

- Promoter regions frequently have higher counts of Gs and Cs
- Model genome as nucleotides drawn independently from two distributions: Background (B) and Promoters (P).
- Emission probabilities based on nucleotide composition in each.
- Transition probabilities based on relative abundance & avg. length



TAAGAATTGTGTCACACACATAAAAACCCTAAGTTAGAGGATTGAGATTGGCA
GACGATTGTTCGTGATAATAAACAAGGGGGGCATAGATCAGGCTCATATTGGC

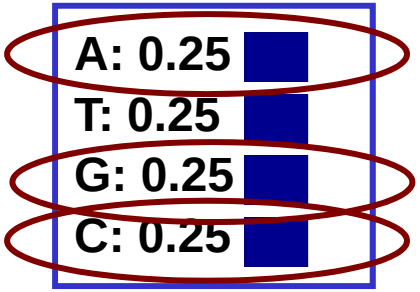
HMM as a *Generative* Model



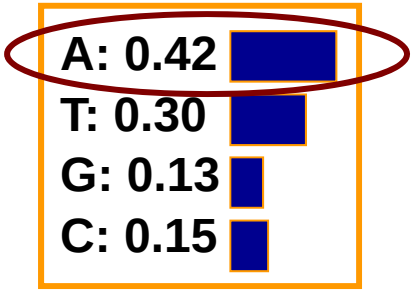
$P(L_{i+1}|L_i)$

	B_{i+1}	P_{i+1}
B_i	0.85	0.15
P_i	0.25	0.75

P(S|B)



P(S|P)



Sequence Classification

PROBLEM: Given a sequence, is it a promoter region?

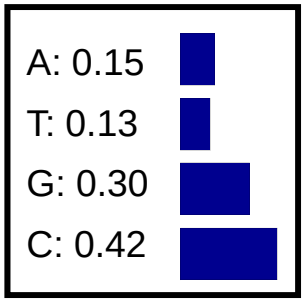
- We can calculate P(S|MP), but what is a *sufficient P value*?

SOLUTION: compare to a null model and calculate log-likelihood ratio

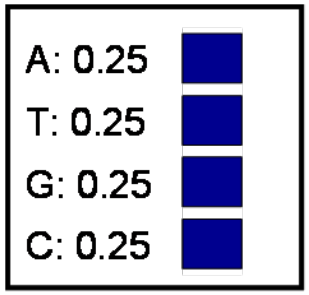
- e.g. background DNA distribution model, B

$$Score = \log \frac{P(S \mid MP)}{P(S \mid B)}$$

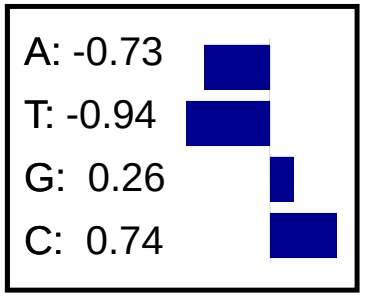
Pathogenicity Islands



Background DNA



Score Matrix

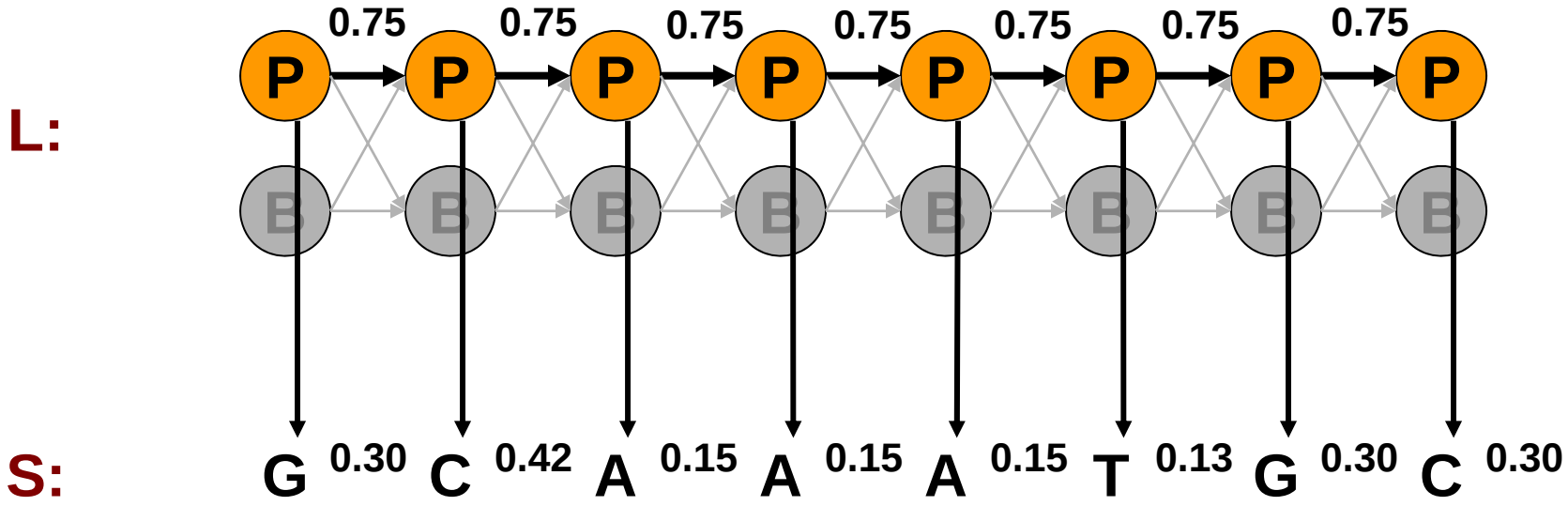


Finding GC-rich regions

- Could use the log-likelihood ratio on windows of fixed size
- Downside: have to evaluate all islands of all lengths repeatedly
- Need: a way to easily find transitions

TAAGAATTGTGTCACACACATAAAAACCCTAAGTTAGAGGATTGAGATTGGCA
GACGATTGTTCGTGATAATAAACAAGGGGGGCATAGATCAGGCTCATATTGGC

Probability of a sequence if all promoter

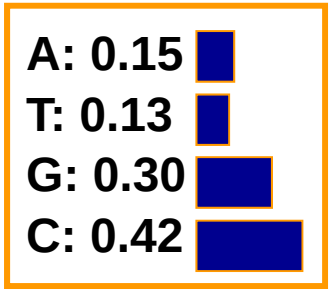


$$P(x,\pi)=a_P*e_P(G)*a_{PP}*e_P(G)*a_{PP}*e_P(C)*a_{PP}*e_P(A)*a_{PP}*\ldots$$

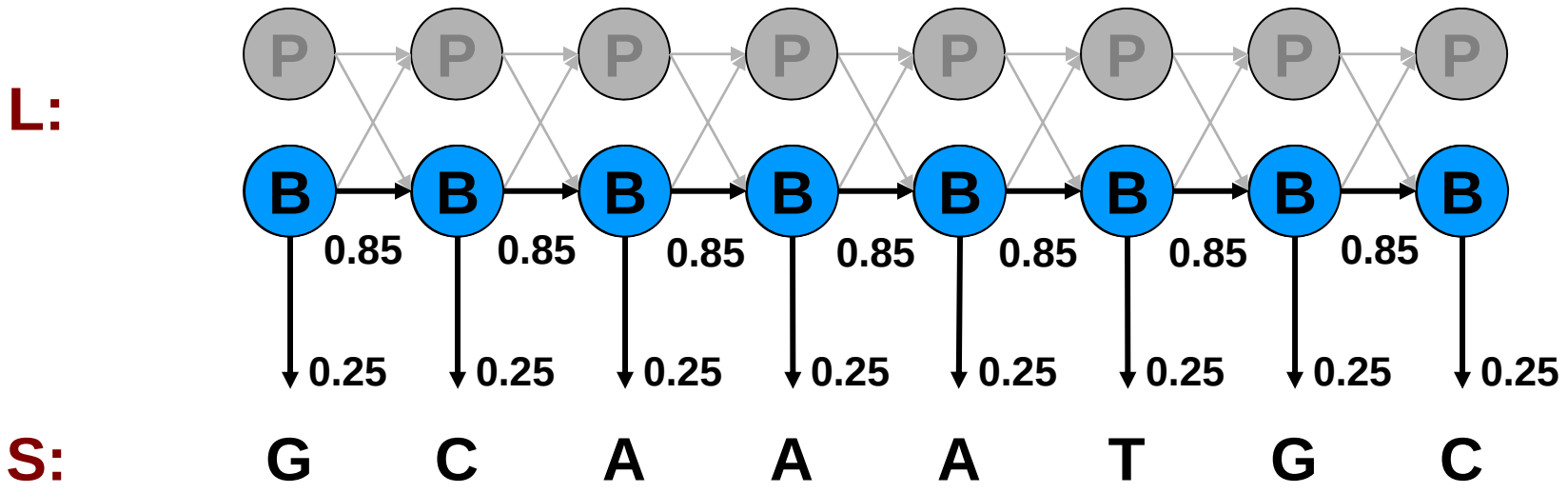
$$=a_p*(0.75)^7*(0.15)^3*(0.13)^1*(0.30)^2*(0.42)^2$$

$$=9.3*10^{-7}$$

Why is this so small?



Probability of the same sequence if all background



$$P = P(G \mid B)P(B_1 \mid B_0)P(C \mid B)P(B_2 \mid B_1)P(A \mid B)P(B_3 \mid B_2)...P(C \mid B_7)$$

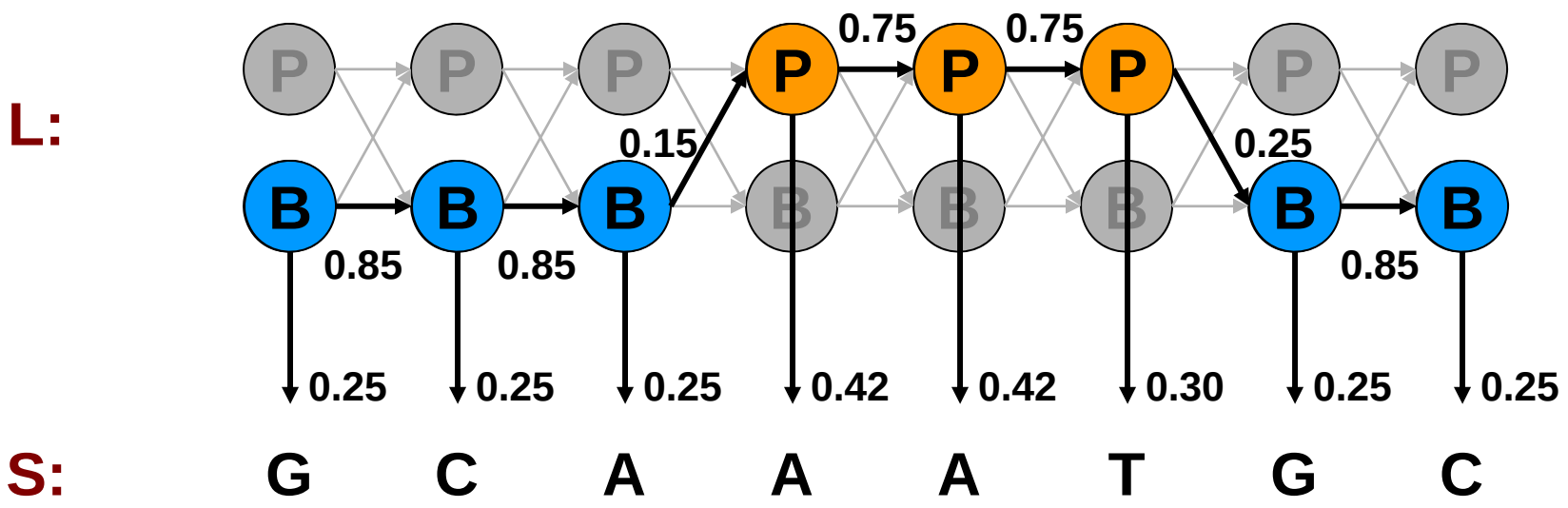
$$= (0.85)^7 \times (0.25)^8$$

$$= 4.9 \times 10^{-6}$$

Compare relative probabilities: 5-fold more likely!

A: 0.25
T: 0.25
G: 0.25
C: 0.25

Probability of the same sequence if mixed



$$P = P(G \mid B)P(B_1 \mid B_0)P(C \mid B)P(B_2 \mid B_1)P(A \mid B)P(P_3 \mid B_2)...P(C \mid B_7)$$

$$= (0.85)^3 \times (0.25)^6 \times (0.75)^2 \times (0.42)^2 \times 0.30 \times 0.15$$

$$= 6.7 \times 10^{-7}$$

Should we try all possibilities? What is the most likely path?

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^{\wedge} = \{\pi_i \mid \pi_i = \text{argmax}_k \Sigma_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi \mid \Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi \mid \Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \Sigma_{\pi} P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

3. DECODING:
What was the sequence of hidden states?

Given: Model parameters $e_i(.)$, a_{ij}

Given: Sequence of emissions x

Find: Sequence of hidden states π

Finding the optimal path

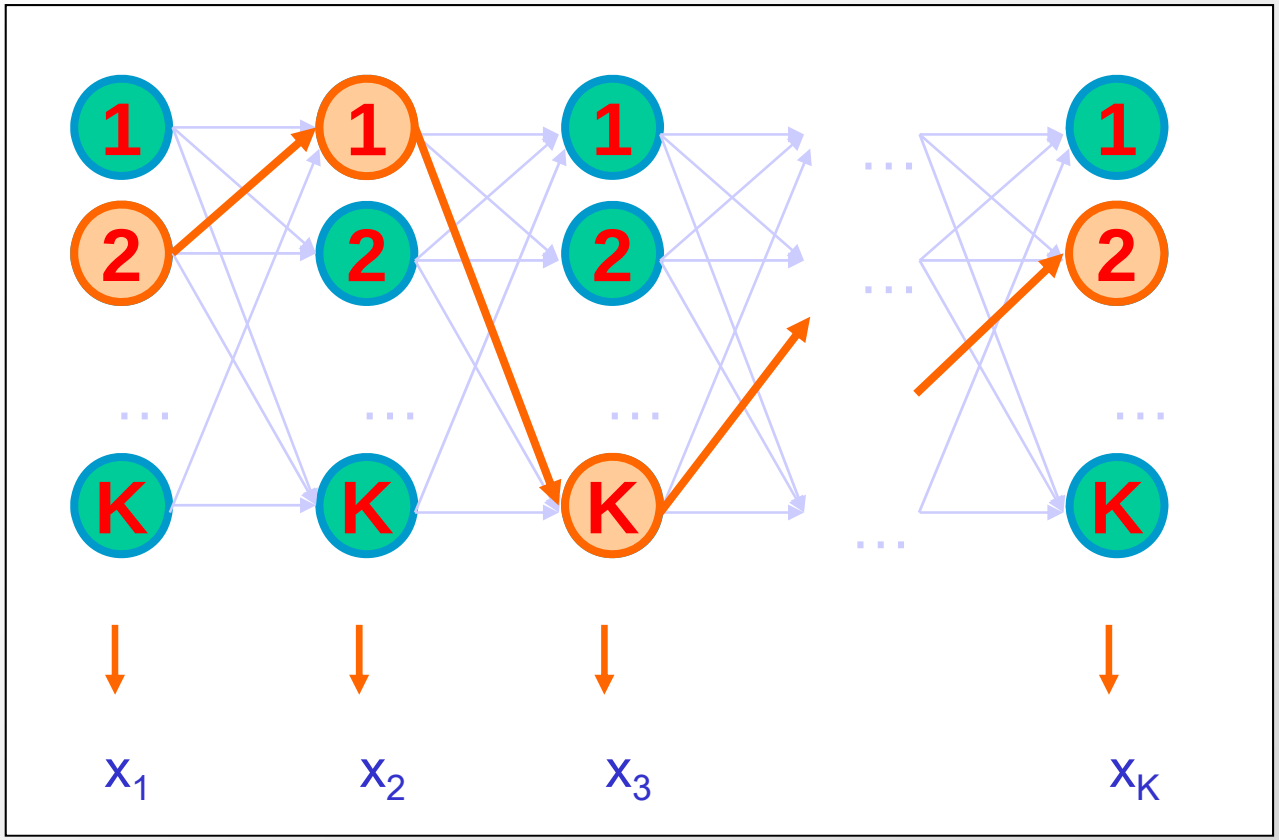
- We can now evaluate any path through hidden states, given the emitted sequences
- How do we find the best path?
- Optimal substructure! Best path through a given state is:
  - Best path to previous state
  - Best transition from previous state to this state
  - Best path to the end state

➔ Viterbi algortithm

- Define $V_k(i)$ = Probability of the most likely path through state $\pi_i=k$
- Compute $V_k(i+1)$ as a function of $\max_{k'} \{ V_{k'}(i) \}$
- $V_k(i+1) = e_k(x_{i+1}) * \max_j a_{jk} V_j(i)$

➔ Dynamic Programming

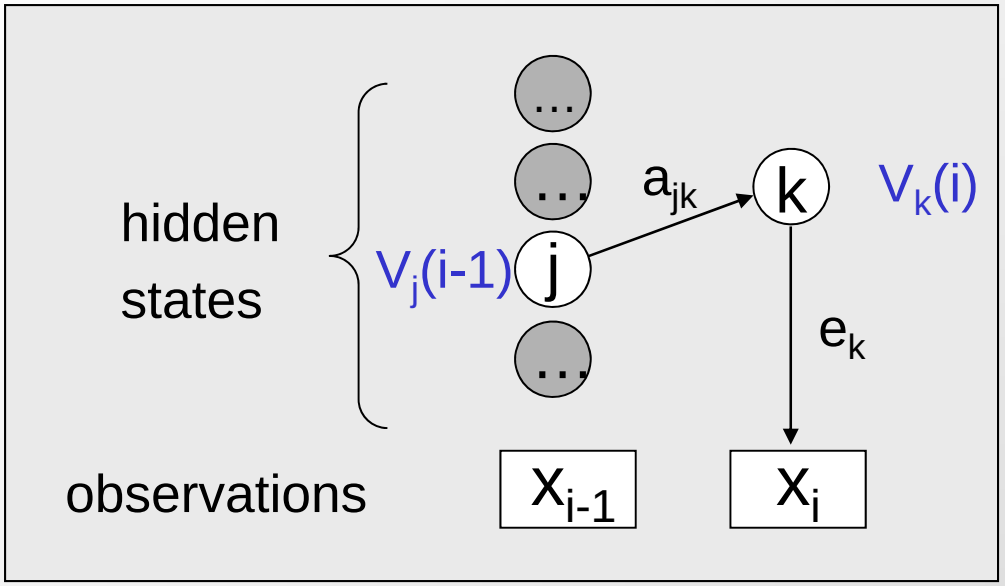
Finding the most likely path



- Find path π^* that maximizes total joint probability $P[x, \pi]$
  - $P(x,\pi) = a_{0\pi_1} * \prod_i e_{\pi_i}(x_i) \times a_{\pi_i \pi_{i+1}}$

start emission transition

Calculate maximum P(x,π) recursively



- Assume we know V_j for the previous time step (i-1)
- Calculate $V_k(i) = e_k(x_i) * \max_j (V_j(i-1) \times a_{jk})$

current max

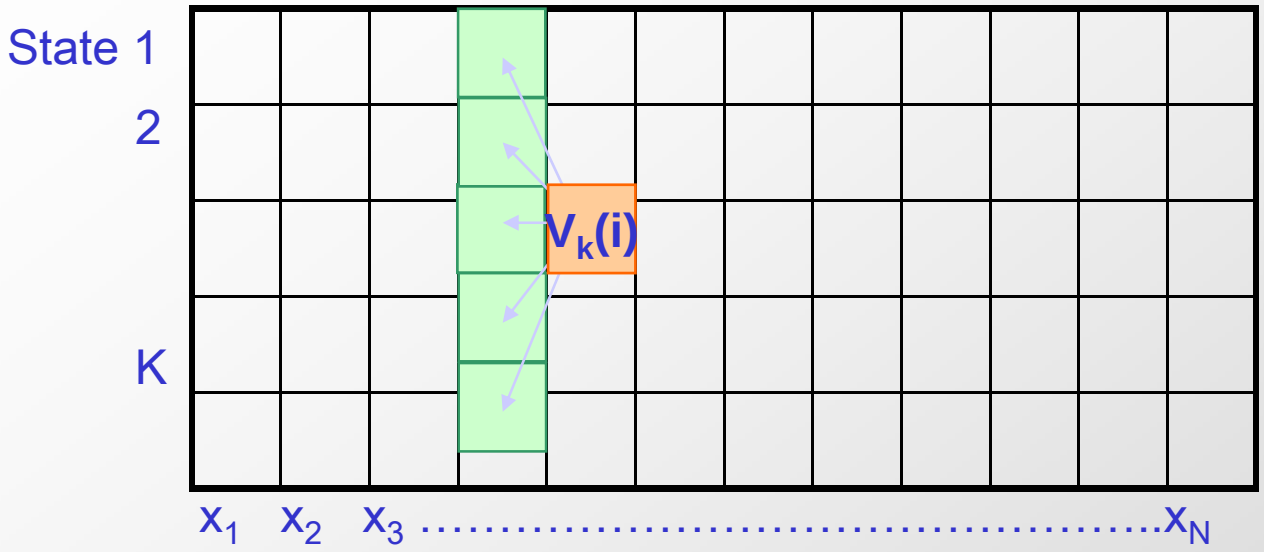
this emission

max ending in state j at step i

Transition from state j

all possible previous states j

The Viterbi Algorithm



Input: x = x1……xN

Initialization:

$V_0(0)=1$, $V_k(0) = 0$, for all $k > 0$

Iteration:

$V_k(i) = e_K(x_i) \times \max_j a_{jk} V_j(i-1)$

Termination:

$P(x, \pi^*) = \max_k V_k(N)$

Traceback:

Follow max pointers back

Similar to aligning states to seq

In practice:

Use log scores for computation

Running time and space:

Time: $O(K^2N)$

Space: $O(KN)$

The six algorithmic settings for HMMs

	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^{\wedge} = \{\pi_i \mid \pi_i = \text{argmax}_k \Sigma_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi \mid \Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi \mid \Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \Sigma_{\pi} P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

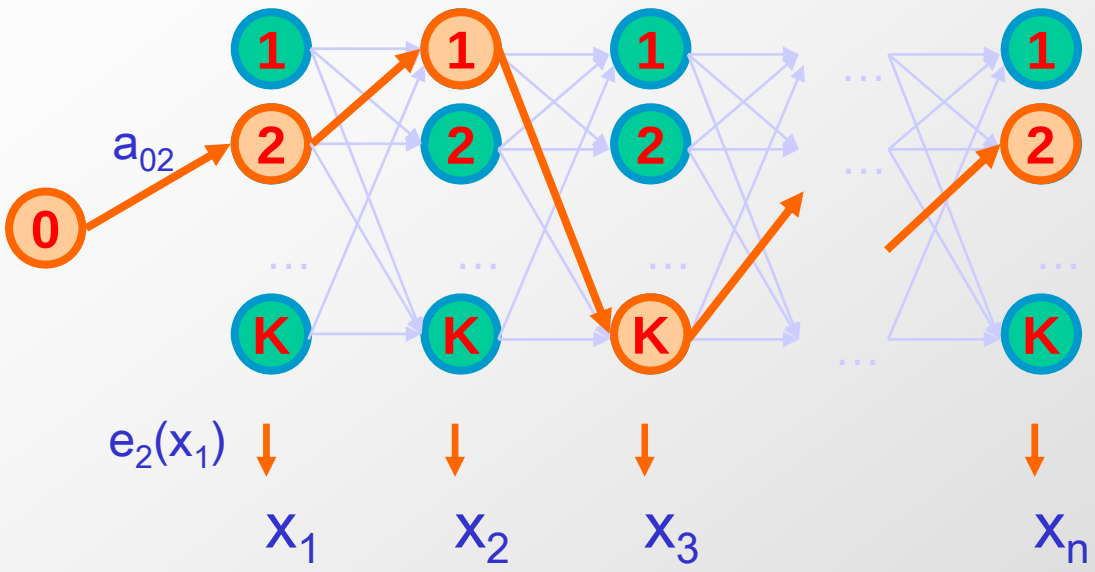
2. EVALUATION
(how well does our model capture the world)

Given: Model parameters $e_i(.)$, a_{ij}

Given: Sequence of emissions x

Find: $P(x|M)$, summed over all possible paths π

Simple: Given the model, generate some sequence x

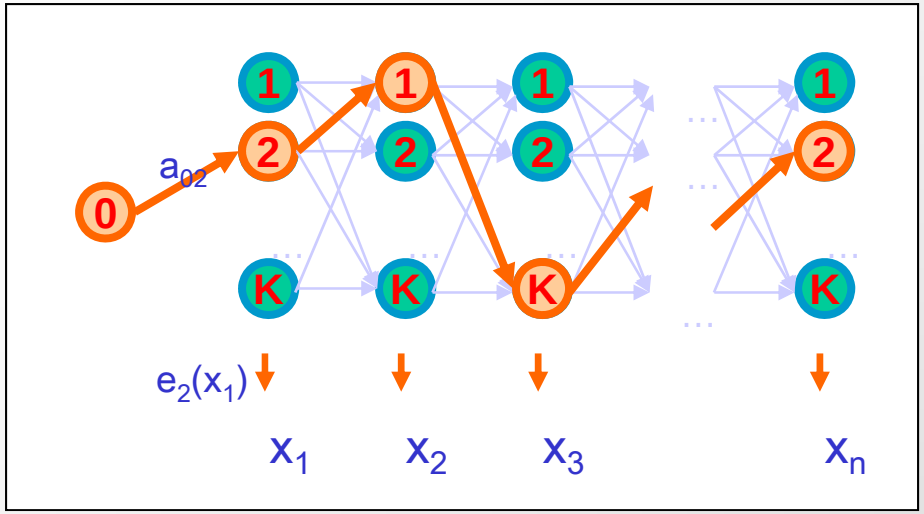


Given a HMM, we can generate a sequence of length n as follows:

1. Start at state π_1 according to prob $a_{0\pi 1}$
2. Emit letter x_1 according to prob $e_{\pi 1}(x_1)$
3. Go to state π_2 according to prob $a_{\pi 1 \pi 2}$
4. … until emitting x_n

We have some sequence x that can be emitted by p. Can calculate its likelihood.
However, in general, many different paths may emit this same sequence x.
How do we find the total probability of generating a given x, over any path?

Complex: Given x, was it generated by the model?



Given a sequence x,

What is the probability that x was generated by the model (using any path)?

– $P(x) = \sum_{\pi} P(x,\pi)$

- Challenge: exponential number of paths

Calculate probability of emission over all paths

- Each path has associated probability
  - Some paths are likely, others unlikely: sum them all up

  → Return total probability that emissions are observed, summed over all paths
  - Viterbi path is the most likely one
    - How much ‘probability mass’ does it contain?
- (cheap) alternative:
  - Calculate probability over maximum (Viterbi) path π*
  - Good approximation if Viterbi has highest density
  - BUT: incorrect
- (real) solution
  - Calculate the exact sum iteratively
    - $P(x) = \sum_{\pi} P(x,\pi)$
  - Can use dynamic programming

The Forward Algorithm – derivation

Define the forward probability:

$$f_l(i) = P(x_1 \ldots x_i, \pi_i = l)$$

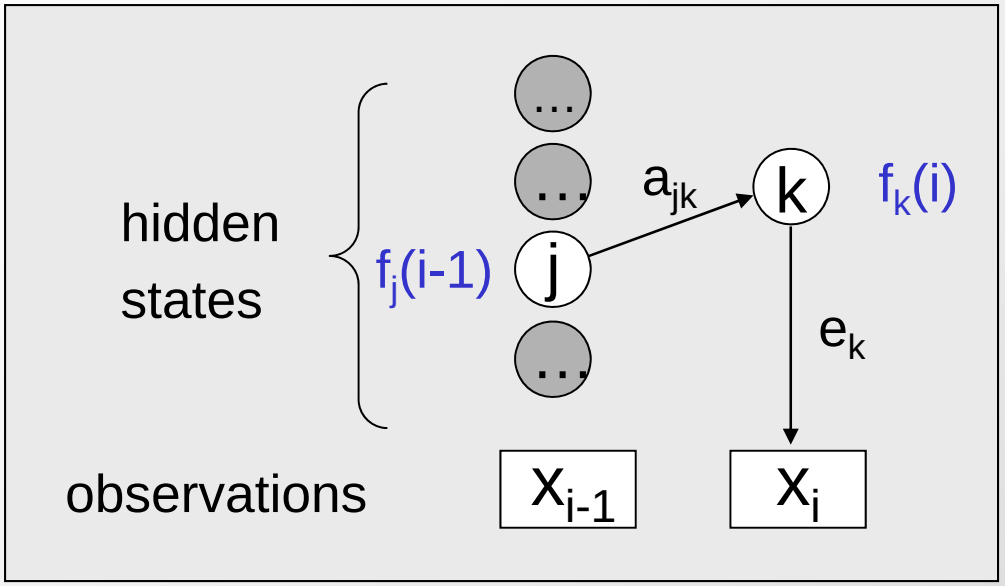
$$= \Sigma_{\pi 1 \ldots \pi i-1} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1}, \pi_i = l)\, e_l(x_i)$$

$$= \Sigma_k \boxed{\Sigma_{\pi 1 \ldots \pi i-2} P(x_1 \ldots x_{i-1}, \pi_1, \ldots, \pi_{i-2}, \pi_{i-1} = k)}\, a_{kl}\, e_l(x_i)$$

$$= \Sigma_k \boxed{f_k(i-1)}\, a_{kl}\, e_l(x_i)$$

$$= e_l(x_i)\, \Sigma_k \boxed{f_k(i-1)}\, a_{kl}$$

Calculate total probability $\Sigma_\pi P(x,\pi)$ recursively



- Assume we know f_j for the previous time step (i-1)
- Calculate $f_k(i) = e_k(x_i) * sum_j (f_j(i-1) \times a_{jk})$

updated sum

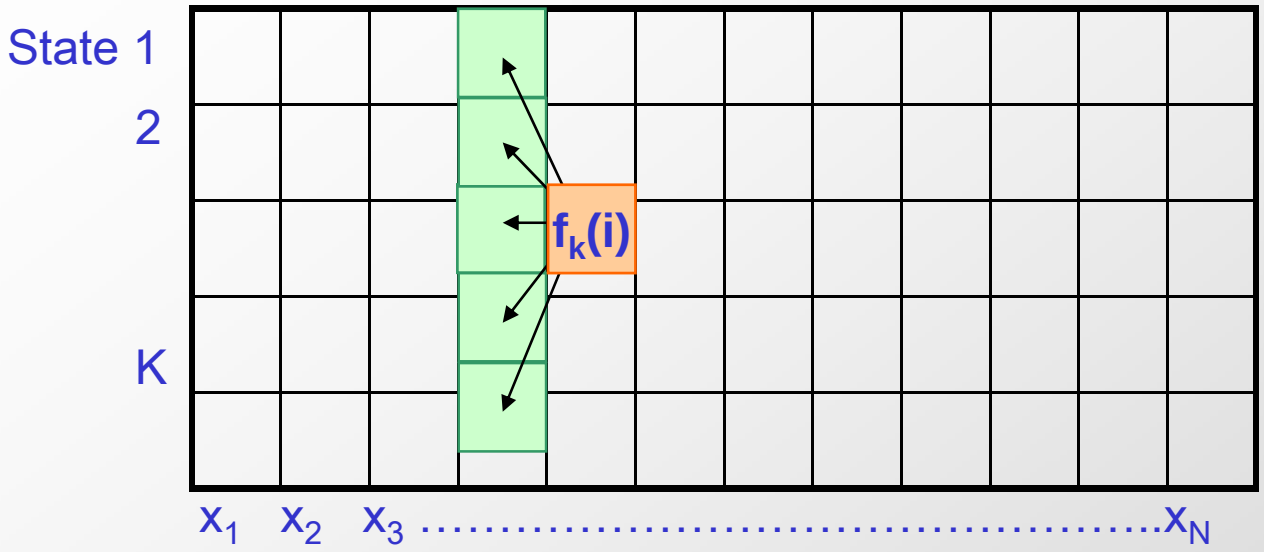
this emission

sum ending in state j at step i

transition from state j

every possible previous state j

The Forward Algorithm



Input: x = x1……xN

Initialization:

$f_0(0)=1$, $f_k(0) = 0$, for all $k > 0$

Iteration:

$f_k(i) = e_K(x_i) \times \text{sum}_j\ a_{jk}\ f_j(i\text{-}1)$

Termination:

$P(x, \pi^*) = \text{sum}_k\ f_k(N)$

In practice:

Sum of log scores is difficult

→ approximate exp(1+p+q)

→ scaling of probabilities

Running time and space:

Time: $O(K^2N)$

Space: $O(KN)$

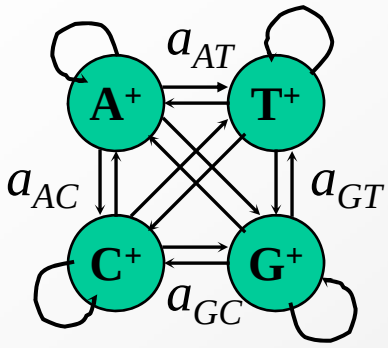
The six algorithmic settings for HMMs

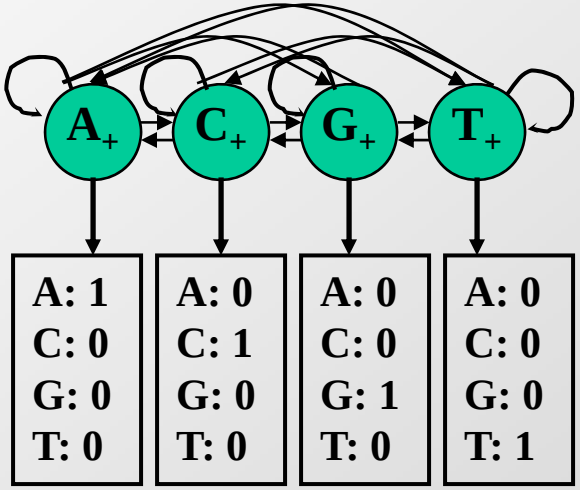
	One path	All paths
Scoring	1. Scoring x, one path $P(x,\pi)$ Prob of a path, emissions	2. Scoring x, all paths $P(x) = \sum_{\pi} P(x,\pi)$ Prob of emissions, over all paths
Decoding	3. Viterbi decoding $\pi^* = \text{argmax}_{\pi} P(x,\pi)$ Most likely path	4. Posterior decoding $\pi^{\wedge} = \{\pi_i \mid \pi_i = \text{argmax}_k \Sigma_{\pi} P(\pi_i = k \mid x)\}$ Path containing the most likely state at any time point.
Learning	5. Supervised learning, given π $\Lambda^* = \text{argmax}_{\Lambda} P(x,\pi \mid \Lambda)$ 6. Unsupervised learning. $\Lambda^* = \text{argmax}_{\Lambda} \max_{\pi} P(x,\pi \mid \Lambda)$ Viterbi training, best path	6. Unsupervised learning $\Lambda^* = \text{argmax}_{\Lambda} \Sigma_{\pi} P(x,\pi \mid \Lambda)$ Baum-Welch training, over all paths

Introducing memory

- State, emissions, only depend on **current** state
- How do you count **di-nucleotide** frequencies?
  - CpG islands
  - Codon triplets
  - Di-codon frequencies
- Introducing memory to the system
  - Expanding the number of states

Example 2: CpG islands: incorporating memory

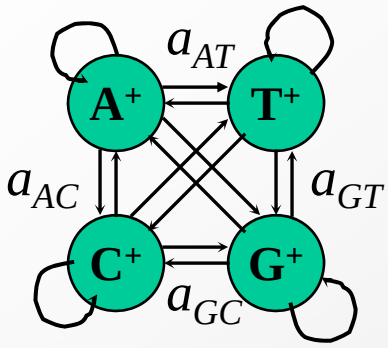


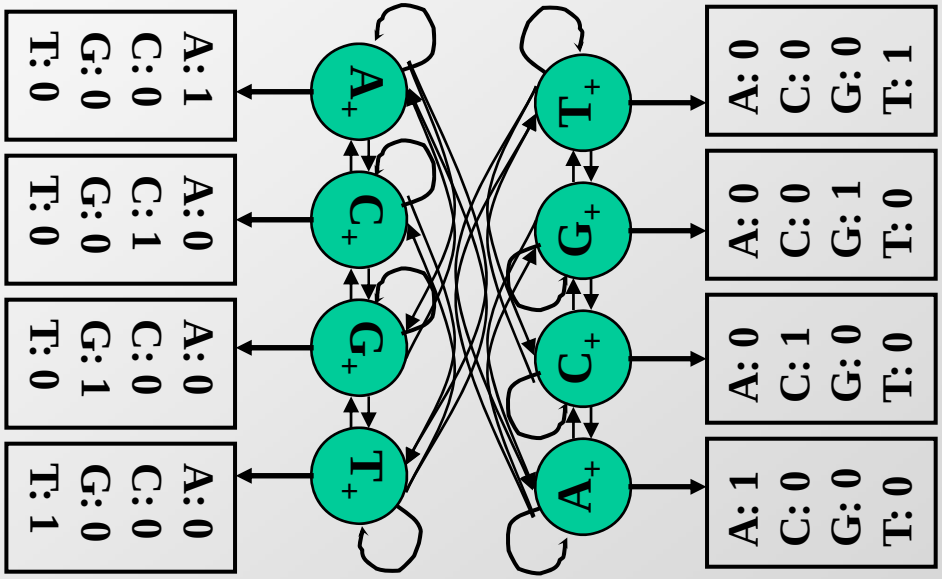


- Markov Chain
  - Q: states
  - p: initial state probabilities
  - A: transition probabilities

- HMM
  - Q: states
  - V: observations
  - p: initial state probabilities
  - A: transition probabilities
  - E: emission probabilities

Counting nucleotide transitions: Markov/HMM





- Markov Chain
  - Q: states
  - p: initial state probabilities
  - A: transition probabilities

- HMM
  - Q: states
  - V: observations
  - p: initial state probabilities
  - A: transition probabilities
  - E: emission probabilities

What have we learned ?

- Modeling sequential data
  - Recognize a ***type*** of sequence, genomic, oral, verbal, visual, etc…
- Definitions
  - Markov Chains
  - Hidden Markov Models (HMMs)
- Simple examples
  - Recognizing GC-rich regions.
  - Recognizing CpG dinucleotides
- Our first computations
  - Running the model: know model → generate sequence of a 'type'
  - Evaluation: know model, emissions, states → p?
  - Viterbi: know model, emissions → find optimal path
  - Forward: know model, emissions → total p over all paths
- Next time:
  - Posterior decoding
  - Supervised learning
  - Unsupervised learning: Baum-Welch, Viterbi training