

Lecture 23: Hidden Markov Models

Chapter 11

11/18/2008

Comp 590/Comp 790-90

Fall 2008

1

Dinucleotide Frequency



- Consider all 2-mers in a sequence
{AA,AC,AG,AT,CA,CC,CG,CT,GA,GC,GG,GT,TA,TC,TG,TT}
- Given 4 nucleotides:
each with probability of occurrence is $\sim 1/4$.
Thus, one would expect that the probability of occurrence of any given dinucleotide is $\sim 1/16$.
- However, the frequencies of dinucleotides in DNA sequences vary widely.
- In particular, CG is typically underrepresented (frequency of CG is typically $< 1/16$)



11/18/2008

Comp 590/Comp 790-90

Fall 2008

2

Example



- From a 291829 base sequence

AA	0.120214646984	GA	0.056108392614
AC	0.055409350713	GC	0.037792809463
AG	0.068848773935	GG	0.043357731266
AT	0.083425853585	GT	0.046828954041
CA	0.074369148950	TA	0.077206436668
CC	0.044927148868	TC	0.056207766218
CG	0.008179475581	TG	0.063698479926
CT	0.066857875186	TT	0.096567155996

- Expected value 0.0625
- CG is 7 times smaller than expected



Why so few CGs?



- CG is the least frequent dinucleotide because C in CG is easily *methyalted*, in other words it reacts and turns into a T afterwards
- However, methylation is suppressed around genes and transcription factor regions
- So, CG appears at *relatively* higher frequency in these important areas
- These localized areas of higher CG frequency are called **CG-islands**
- Finding the CG islands within a genome is an important problem



CG Island Analogy



- The CG islands problem can be modeled after a toy problem named *"The Fair Bet Casino"*
- The outcome of the game is determined by coin flips with two possible outcomes: **Heads** or **Tails**
- However, there are two different coins
 - A **Fair** coin: **Heads** and **Tails** with same probability $\frac{1}{2}$.
 - The **Biased** coin: **Heads** with prob. $\frac{3}{4}$, **Tails** with prob. $\frac{1}{4}$.



The "Fair Bet Casino" (cont'd)



- Thus, we define the probabilities:
 - $P(H | \text{Fair}) = P(T | \text{Fair}) = \frac{1}{2}$
 - $P(H | \text{Bias}) = \frac{3}{4}$, $P(T | \text{Bias}) = \frac{1}{4}$
 - The crooked dealer doesn't want to get caught switching between coins, so he does so infrequently
 - Changes between Fair and Biased coins with probability 10%



The Fair Bet Casino Problem



- **Input:** A sequence $x = x_1 x_2 x_3 \dots x_n$ of coin tosses made by some combination of the two possible coins (F or B).
- **Output:** A sequence $\pi = \pi_1 \pi_2 \pi_3 \dots \pi_n$, with each π_i being either F or B indicating that x_i is the result of tossing the Fair or Biased coin respectively.



Problem...



Fair Bet Casino Problem

Any observed outcome of coin tosses *could* have been generated by *either* coin, or any combination.

But, all sequences are not equally likely. What coin combination has the highest probability of generating the observed series of tosses?



Decoding Problem



P(x | fair coin) vs. P(x | biased coin)



- Suppose first, that the dealer never exchanges coins.
- Some definitions:
 - **P(x | Fair)**: prob. of the dealer generating the outcome x using the *Fair* coin.
 - **P(x | Biased)**: prob. of the dealer generating outcome x using the *Biased* coin .



P(x | fair coin) vs. P(x | biased coin)



- **P(x | Fair) = P($x_1 \dots x_n$ | Fair) =**
$$\prod_{i=1, n} p(x_i | \text{Fair}) = (1/2)^n$$
- **P(x | Biased) = P($x_1 \dots x_n$ | Biased coin) =**
$$\prod_{i=1, n} p(x_i | \text{Biased}) = (3/4)^k (1/4)^{n-k} = 3^k / 4^n$$
 - Where k is the number of *Heads* in x .



P(x | fair coin) vs. P(x | biased coin)

- When is a sequence equally likely to have come from the Fair or Biased coin?

$$P(x | \text{Fair}) = P(x | \text{Biased})$$

$$1/2^n = 3^k/4^n$$

$$2^n = 3^k$$

$$n = k \log_2 3$$

- when $k = n / \log_2 3$ ($k \sim 0.63 n$)
- So when the number of heads is greater than 63% the dealer most likely used the biased coin



Log-odds Ratio

- We can define the *log-odds ratio* as follows:

$$\begin{aligned} \log_2(P(x | \text{Fair}) / P(x | \text{Biased})) &= \\ &= \sum_{i=1}^k \log_2(p(x_i | \text{Fair}) / p(x_i | \text{Biased})) \\ &= n - k \log_2 3 \end{aligned}$$

- The log-odds ratio is convenient because we can *add* the logs of small probabilities rather than *multiply* them

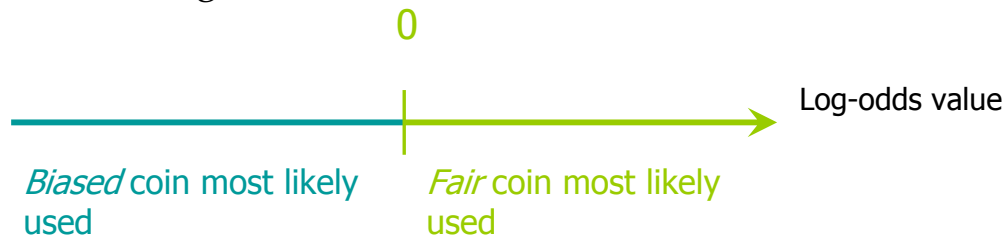


Computing Log-odds Ratio in Sliding Windows



$$x_1 x_2 \boxed{x_3 x_4 x_5 x_6 x_7} x_8 \dots x_n$$

Consider a *sliding window* of the outcome sequence.
Find the log-odds for this short window.



Disadvantages:

- the length of CG-island is not known in advance
- different window sizes may classify the same position differently



Key Elements of this Problem



- There is an unknown, *hidden*, state for each observation (Was the coin the Fair or Biased?)
- Outcomes are modeled probabilistically:
 - $P(H | \text{Fair}) = P(T | \text{Fair}) = 1/2$
 - $P(H | \text{Bias}) = 3/4, P(T | \text{Bias}) = 1/4$
- Transitions between states are modeled probabilistically:
 - $P(\pi_i = \text{Biased} | \pi_{i-1} = \text{Biased}) = a_{BB} = 0.9$
 - $P(\pi_i = \text{Biased} | \pi_{i-1} = \text{Fair}) = a_{FB} = 0.1$
 - $P(\pi_i = \text{Fair} | \pi_{i-1} = \text{Biased}) = a_{BF} = 0.1$
 - $P(\pi_i = \text{Fair} | \pi_{i-1} = \text{Fair}) = a_{FF} = 0.9$



Hidden Markov Model (HMM)



- A generalization of this class of problem
- Can be viewed as an abstract machine with k *hidden* states that emits symbols from an alphabet Σ .
- Each state has its own probability distribution, and the machine switches between states according to this probability distribution.
- While in a certain state, the machine makes 2 decisions:
 - What state should I move to next?
 - What symbol - from the alphabet Σ - should I emit?



Why “Hidden”?



- Observers can see the emitted symbols of an HMM but have *no ability to know which state the HMM is currently in*.
- Thus, the goal is to infer the *most likely hidden states of an HMM* based on the given sequence of emitted symbols.

HHHTHTHHTTTTTHTHTHTHHHTHTHTHT
BBBFFFFFFFFFFFFFFFFBBBFFFFFF?



HMM Parameters


 Σ : set of emission characters.

Ex.: $\Sigma = \{0, 1\}$ for coin tossing
(**0** for *Tails* and **1** *Heads*)


$\Sigma = \{1, 2, 3, 4, 5, 6\}$ for dice tossing

Q : set of hidden states, emitting symbols from Σ .

$Q = \{F, B\}$ for coin tossing



HMM Parameters (cont'd)


 $A = (a_{kl})$: a $|Q| \times |Q|$ matrix of probability of changing from state k to state l .

$$a_{FF} = 0.9 \quad a_{FB} = 0.1$$

$$a_{BF} = 0.1 \quad a_{BB} = 0.9$$

$E = (e_k(b))$: a $|Q| \times |\Sigma|$ matrix of probability of emitting symbol b while being in state k .

$$e_F(0) = 1/2 \quad e_F(1) = 1/2$$

$$e_B(0) = 1/4 \quad e_B(1) = 3/4$$



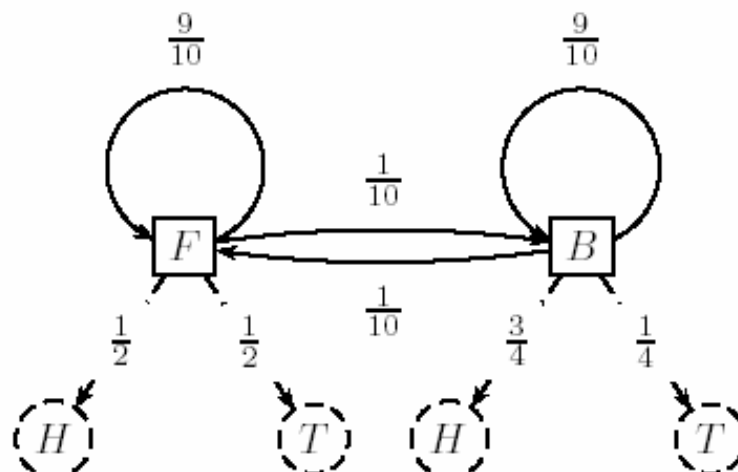
HMM for Fair Bet Casino

- The *Fair Bet Casino* in HMM terms:
 - $\Sigma = \{0, 1\}$ (0 for Tails and 1 Heads)
 - $Q = \{F, B\}$ - F for Fair & B for Biased coin.
- Transition Probabilities A , Emission Probabilities E

A	Fair	Biased
Fair	0.9	0.1
Biased	0.1	0.9

E	Tails(0)	Heads(1)
Fair	$\frac{1}{2}$	$\frac{1}{2}$
Biased	$\frac{1}{4}$	$\frac{3}{4}$

HMM for Fair Bet Casino (cont'd)



HMM model for the *Fair Bet Casino* Problem

Hidden Paths



- A **path** $\pi = \pi_1 \dots \pi_n$ in the HMM is defined as a sequence of states.
- Consider path $\pi = \text{FFFBBBBBFFF}$ and sequence $x = 01011101001$

x	=	0	1	0	1	1	1	0	1	0	0	1
π	=	F	F	F	B	B	B	B	B	F	F	F
$P(x_i \pi_i)$		$1/2$	$1/2$	$1/2$	$3/4$	$3/4$	$3/4$	$1/4$	$3/4$	$1/2$	$1/2$	$1/2$
$P(\pi_{i-1} \rightarrow \pi_i)$		$1/2$	$9/10$	$9/10$	$1/10$	$9/10$	$9/10$	$9/10$	$9/10$	$1/10$	$9/10$	$9/10$

Probability that x_i was emitted from state π_i
 Transition probability from state π_{i-1} to state π_i



$P(x | \pi)$ Calculation



- **$P(x | \pi)$** : Probability that sequence x was generated by the path π :

$$P(x | \pi) = P(\pi_0 \rightarrow \pi_1) \cdot \prod_{i=1}^n P(x_i | \pi_i) \cdot P(\pi_i \rightarrow \pi_{i+1})$$

$$= a_{\pi_0, \pi_1} \cdot \prod e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}}$$



Decoding Problem



- **Goal:** Find an optimal hidden path of state transitions given a set of observations.
- **Input:** Sequence of observations $x = x_1 \dots x_n$ generated by an HMM $M(\Sigma, Q, A, E)$
- **Output:** A path that maximizes $P(x | \pi)$ over all possible paths π .



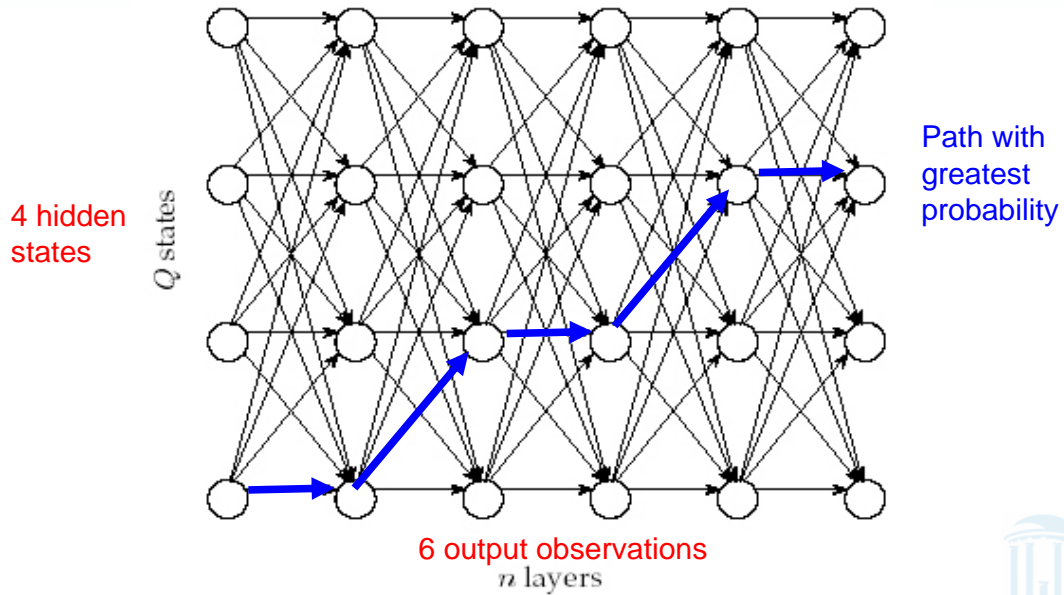
Building Manhattan for Decoding Problem



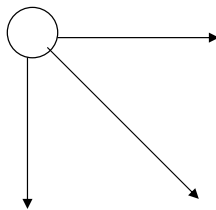
- Andrew Viterbi used the Manhattan grid (Dynamic programming) model to solve the *Decoding Problem*.
- Every choice of $\pi = \pi_1 \dots \pi_n$ corresponds to a path in the graph.
- The only valid direction in the graph is *eastward*.
- This graph has $|Q|^2(n-1)$ edges.



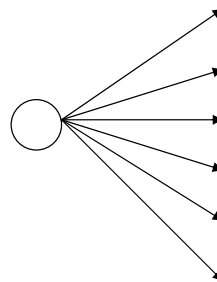
Edit Graph for Decoding Problem



Decoding Problem vs. Alignment Problem



Valid directions in the
alignment problem.



Valid directions in the
decoding problem.



Decoding Problem



- The *Decoding Problem* is reduced to finding a longest path in the *directed acyclic graph (DAG)*
- **Notes:** the length of the path is defined as the *product* of its edges' weights, not the *sum*.
(But, using the log of the weights makes it a sum again!)



Decoding Problem (cont'd)



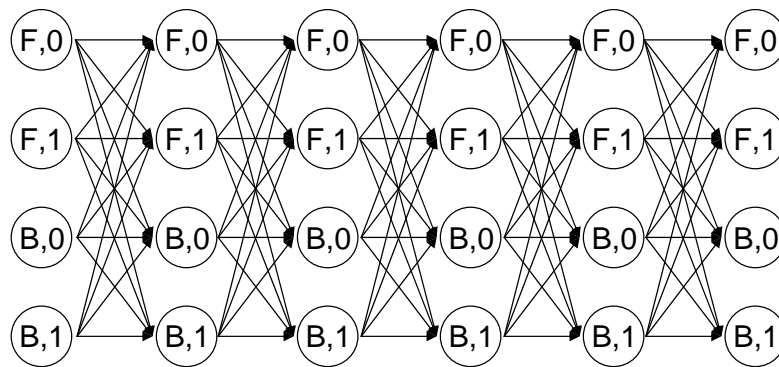
- Every path in the graph has the probability $P(x | \pi)$.
- The Viterbi algorithm finds the path that maximizes $P(x | \pi)$ among all possible paths.
- The Viterbi algorithm runs in $O(n | Q |^2)$ time.



Viterbi for Fair Bet Casino



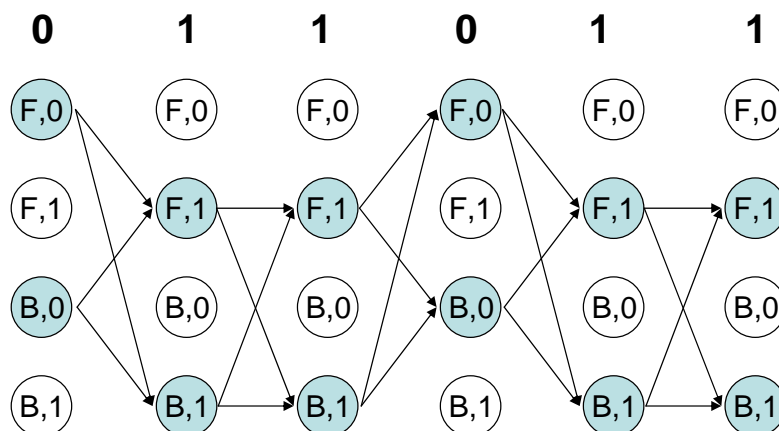
- Each vertex represents a hidden-state/observation tuple



Viterbi for Fair Bet Casino



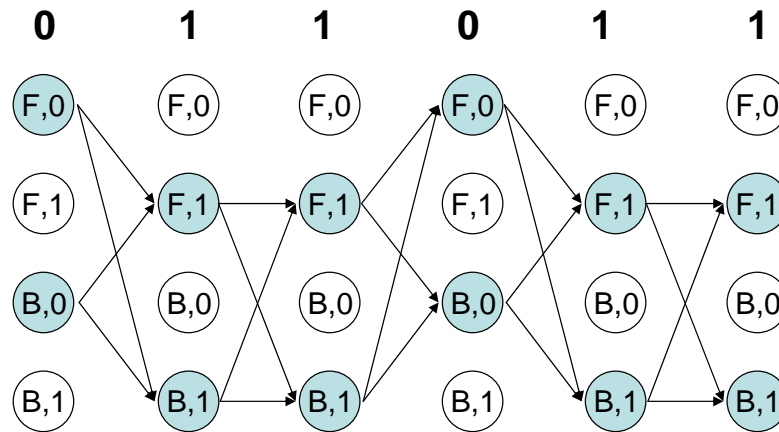
- A given observed sequence reduces the number of potential paths.



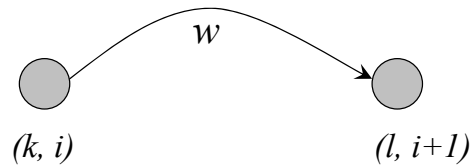
Viterbi for Fair Bet Casino



- How to label the edges?
- Goal: choose the highest probability path



Decoding Problem: weights of edges

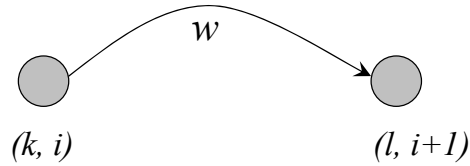


The weight w is given by:

???



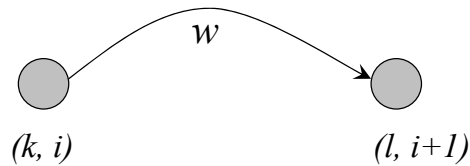
Decoding Problem: weights of edges



The weight w is given by:

The Total probability

$$P(x | \pi) = \prod_{i=0}^n e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i, \pi_{i+1}}$$



The weight w is given by

Each edge is a factor in the product

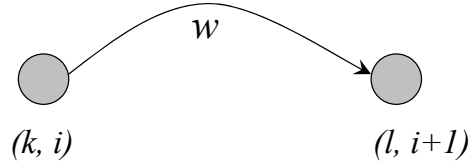
$$i\text{-th term} = e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i, \pi_{i+1}}$$



Decoding Problem: weights of edges



i -th term = $e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}} = e_l(x_{i+1}) \cdot a_{kl}$ for $\pi_i = k, \pi_{i+1} = l$



The weight $w = e_l(x_{i+1}) \cdot a_k$

Solve for the path of highest probability

Observation: a prefix is also an optimal path

Where have we seen this before?



Decoding Problem as a Dynamic Program



$$S_{l,i+1} = \max_{k \in Q} \{s_{k,i} \cdot \text{weight of edge between } (k,i) \text{ and } (l,i+1)\} =$$

$$\max_{k \in Q} \{s_{k,i} \cdot a_{kl} \cdot e_l(x_{i+1})\} =$$

$$e_l(x_{i+1}) \cdot \max_{k \in Q} \{s_{k,i} \cdot a_{kl}\}$$



Decoding Problem (cont'd)



- Initialization:
 - $s_{begin,0} = 1$
 - $s_{k,0} = 0$ for $k \neq begin$.
- Let π^* be the optimal path. Then,

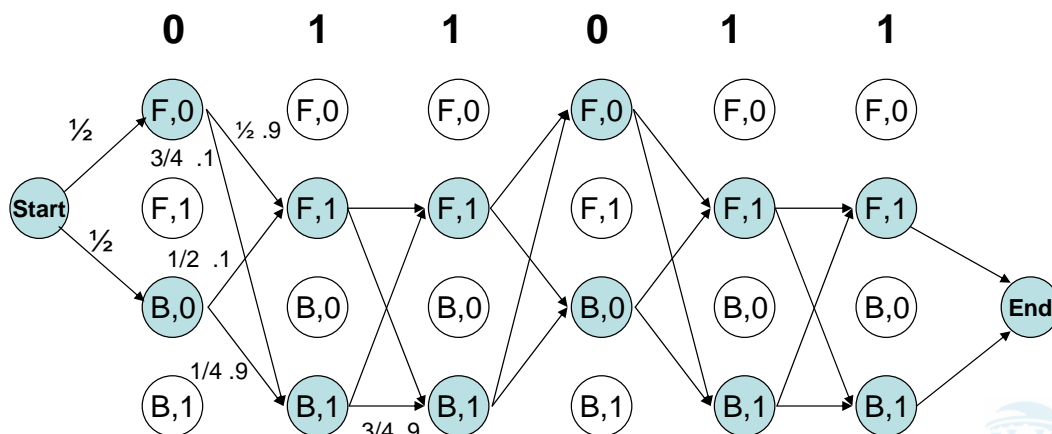
$$P(x | \pi^*) = \max_{k \in Q} \{s_{k,n} \cdot a_{k,end}\}$$



Viterbi for Fair Bet Casino



- How to label the edges?
- Goal: choose the highest probability path



Viterbi Algorithm



- Rather than addition Viterbi uses multiplication
- Covert edge weights to logs, and then it is back to addition, which has another advantage
- The value of the product can become extremely small, which leads to underflowing.
- Logs avoid underflowing.

$$s_{k,i+1} = \log e_l(x_{i+1}) + \max_{k \in Q} \{s_{k,i} + \log(a_{kl})\}$$



Forward-Backward Problem



Given: a sequence of coin tosses generated by an HMM.

Goal: find the probability that the dealer was using a biased coin at a particular time.



Forward Algorithm



- Defined $f_{k,i}$ (*forward probability*) as the probability of emitting the prefix $x_1 \dots x_i$ and reaching the state $\pi = k$.
- The recurrence for the forward algorithm is:

$$f_{k,i} = e_k(x_i) \cdot \sum_{l \in Q} f_{l,i-1} \cdot a_{lk}$$



Backward Algorithm



- However, *forward probability* is not the only factor affecting $P(\pi_i = k|x)$.
- The sequence of transitions and emissions that the HMM undergoes between π_i and π_{i+1} also affect $P(\pi_i = k|x)$.



Backward Algorithm (cont'd)



- *Backward probability* $b_{k,i} \equiv$ the probability of being in state $\pi_i = k$ and emitting the *suffix* $x_{i+1} \dots x_n$.
- The *backward algorithm's* recurrence:

$$b_{k,i} = \sum_{l \in Q} e_l(x_{i+1}) \cdot b_{l,i+1} \cdot a_{kl}$$



Backward-Forward Algorithm



- The probability that the dealer used a biased coin at any moment i is as follows:

$$P(\pi_i = k|x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \cdot b_k(i)}{P(x)}$$



HMM Parameter Estimation



- So far, we have assumed that the transition and emission probabilities are known.
- However, in most HMM applications, the probabilities are not known. It's very hard to estimate the probabilities.



HMM Parameter Estimation (cont'd)



- Let Θ be a vector combining the unknown transition and emission probabilities.
- Given training sequences x^1, \dots, x^m , let $P(x | \Theta)$ be the max. prob. of x given the assignment of param.'s Θ .
- Then our goal is to find

$$\max_{\Theta} \prod_{j=1}^m P(x_j | \Theta)$$



Finding Distant Members of a Protein Family



- A distant cousin of functionally related sequences in a protein family may have weak pairwise similarities with each member of the family and thus fail significance test.
- However, they may have weak similarities with *many* members of the family.
- The goal is to align a sequence to *all* members of the family at once.
- Family of related proteins can be represented by their multiple alignment and the corresponding profile.



Profile Representation of Protein Families



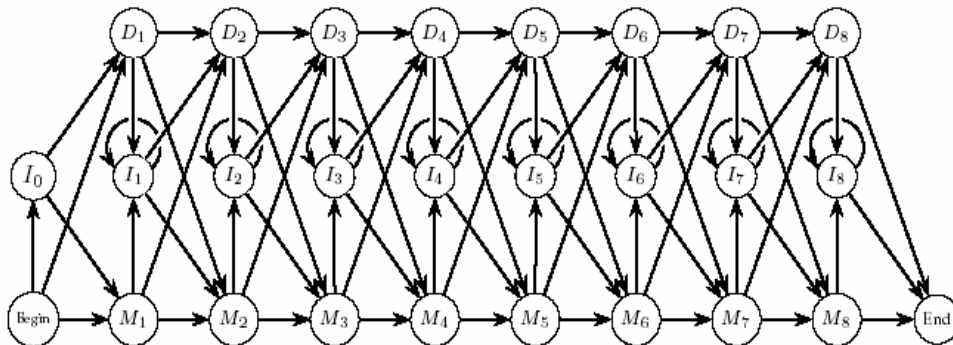
Aligned DNA sequences can be represented by a $4 \cdot n$ profile matrix reflecting the frequencies of nucleotides in every aligned position.

A		.72	.14	0	0	.72	.72	0	0
T		.14	.72	0	0	0	.14	.14	.86
G		.14	.14	.86	.44	0	.14	0	0
C		0	0	.14	.56	.28	0	.86	.14

Protein family can be represented by a $20 \cdot n$ profile representing frequencies of amino acids.



Profile HMM



A profile HMM



States of Profile HMM



- Match states $M_1 \dots M_n$ (plus *begin/end* states)
- Insertion states $I_0 I_1 \dots I_n$
- Deletion states $D_1 \dots D_n$

- Assumption:

$$e_{I_j}(a) = p(a)$$

where $p(a)$ is the frequency of the occurrence of the symbol a in all the sequences.



Transition Probabilities in Profile HMM



- $\log(a_{MI}) + \log(a_{IM}) = \text{gap initiation penalty}$
- $\log(a_{II}) = \text{gap extension penalty}$



Profile HMM Alignment



- Define $v_j^M(i)$ as the logarithmic likelihood score of the best path for matching $x_1..x_i$ to profile HMM ending with x_i emitted by the state M_j .
- $v_j^L(i)$ and $v_j^D(i)$ are defined similarly.



Profile HMM Alignment: Dynamic Programming

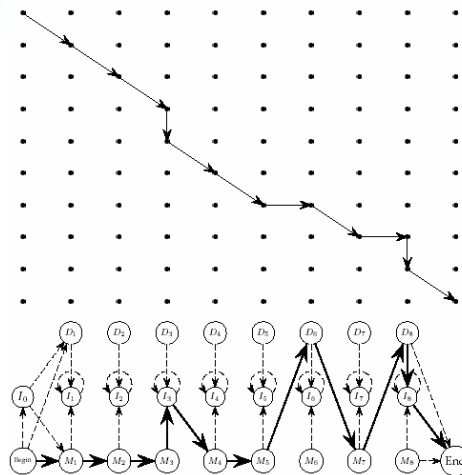


$$v_j^M(i) = \log(e_{M_j}(x_i)/p(x_i)) + \max \begin{cases} v_{j-1}^M(i-1) + \log(a_{M_{j-1}, M_j}) \\ v_{j-1}^I(i-1) + \log(a_{I_{j-1}, M_j}) \\ v_{j-1}^D(i-1) + \log(a_{D_{j-1}, M_j}) \end{cases}$$

$$v_j^I(i) = \log(e_{I_j}(x_i)/p(x_i)) + \max \begin{cases} v_j^M(i-1) + \log(a_{M_j, I_j}) \\ v_j^I(i-1) + \log(a_{I_j, I_j}) \\ v_j^D(i-1) + \log(a_{D_j, I_j}) \end{cases}$$



Paths in Edit Graph and Profile HMM



A path through an edit graph and the corresponding path through a profile HMM

