MACHINE LEARNING IN BIOINFORMATICS

Part 7: Hidden Markov Models

František Mráz
KSVI MFF UK

(Adapted slides from http://bix.ucsd.edu/bioalgorithms/presentations/Ch11_HMM.ppt and book R. Durbin, S. R. Eddy, A. Krogh, G. Mitchison: Biological sequence analysis. Cambridge University Press, 1998)

Outline

Dinucleotide frequency – CG-Islands

- 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 $\sim\frac{1}{4}$. Thus, expected probability of occurrence of a dinucleotide is $\sim\frac{1}{16}$.
- However, the frequencies of dinucleotides in DNA sequences vary widely.
- In particular, frequency of CG is typically $< \frac{1}{16}$

Example

- From a 291829 base sequence

	frequency		frequency
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
- The frequency of CG is 7 times smaller than expected

Why CG-Islands?

- CG is the least frequent dinucleotide because C in CG is easily *methylated* (that is, an H-atom is replaced by a CH_3-group) and the methyl-C has the tendency to mutate into T afterwards
- However, the methylation is suppressed around genes and transcription factor regions in a genome. So, CG appears at *relatively* higher frequency within these important areas called CG-islands
- Finding the CG islands within a genome is among the most reliable gene finding approaches
- **Classical definition:** A CG island is DNA sequence of length about 200bp with a C+G content of 50% and a ratio of observed-to-expected number of CG's that is above 0.6. (Gardiner-Garden & Frommer, 1987)

Problems

1. **Discrimination problem:** Given a short segment of genomic sequence. How can we decide whether this segment comes from a CG-island or not?

   $\longrightarrow$ Markov Model

2. **Localization problem:** Given a long segment of genomic sequence. How can we find all contained CG-islands?

   $\longrightarrow$ Hidden Markov Model

Markov Model

Definition: A (time-homogeneous) Markov model (of order 1) is a system $M = (Q, A)$ consisting of

$Q = \{s_1, \ldots, s_k\}$: a finite set of states and

$A = (a_{kl})$: a $|Q| \times |Q|$ matrix of probability of changing from state s_k to state s_l. $P(x_{i+1} = s_l,\ x_i = s_k) = a_{kl}$ with $\sum_{l \in S} a_{kl} = 1$ for all $k \in S$.

Definition: A Markov chain is a chain $x_0, x_1, \ldots, x_n, \ldots$ of random variables, which take states in the state set Q such that

$P(x_n = s \,|\cap_{j<n} x_j) = P(x_n = s\,|x_{n-1})$ is $true$ for all $n > 0$ and $s \in S$.

Definition: A Markov chain is called homogeneous, if the probabilities are not dependent on n. (At any time i the chain is in a specific state x_i and at the tick of a clock the chain changes to state x_j according to the given transition probabilities.)

Example

- Weather in Prague, daily at midday:
  - Possible states are rain, sun or clouds.
  - Transition probabilities:

	r	s	c
r	0.2	0.3	0.5
s	0.2	0.6	0.2
c	0.3	0.3	0.4

- A Markov chain would be the observation of the weather:
  `...rrrrrrccsssssscscscccrrcrcssss...`
- Types of questions that the model can answer:
  1. If it is sunny today, what is the probability that the sun will shine for the next seven days?
  2. How large is the probability, that it will rain for a month?

Modeling the begin and end states

- We must specify the initialization of the chain – an initial probability $P(x_1)$ of starting in a particular state. We can add a begin state to the model that is labeled $'Begin'$ and add this to the states set. We will always assume that $x_0 = Begin$ holds. Then the probability of the first state in the Markov chain is

$$P(x_1 = s) = a_{Begin,s} = P(s),$$

  where $P(s)$ denotes the background probability of state s.
- Similarly, we explicitly model the end of the sequence using an end state $'End'$. Thus, the probability that we end in state t is

$$P(End|x_n = t) = a_{t,End}.$$

Probability of Markov chains

- Given a sequence of states $\boldsymbol{x} = x_1, x_2, x_3, \dots, x_L$. What is the probability that a Markov chain will step through precisely this sequence of states?

$$P(x) = P(x_L, x_{L-1}, \dots, x_1)$$
$$= P(x_L | x_{L-1}, \dots, x_1)\, P(x_{L-1} | x_{L-2}, \dots, x_1) \;\dots\; P(x_1)$$

[by repeated application of $P(X,Y) = P(X|Y)P(Y)$]

$$= P(x_L|x_{L-1})P(x_{L-1}|x_{L-2}) \dots P(x_2|x_1)P(x_1)$$
$$= P(x_1)\prod_{i=2}^{L} a_{x_{i-1},x_i} = \prod_{i=1}^{L} a_{x_{i-1},x_i}$$

$$\uparrow$$

If $x_0 = Begin$

Example

```
• # Markov chain that generates CpG islands
• # (Source: DEKM98, p 50)
• # Number of states:
• 6
• # State labels (*=Begin, +=End):
• A C G T * +
• # Transition matrix:
• 0.1795 0.2735 0.4255 0.1195 0 0.002
• 0.1705 0.3665 0.2735 0.1875 0 0.002
• 0.1605 0.3385 0.3745 0.1245 0 0.002
• 0.0785 0.3545 0.3835 0.1815 0 0.002
• 0.2495 0.2495 0.2495 0.2495 0 0.002
• 0.0000 0.0000 0.0000 0.0000 0 1.000
```

Transition matrices are generally calculated from training sets.

- In our case the transition matrix $\mathbf{P}^+$ for a DNA sequence that comes from a CG-island, is determined as follows:

$$p_{st}^{+} = \frac{c_{st}^{+}}{\sum_{t'} c_{st'}^{+}}$$

- where c_{st} is the number of positions in a training set of CG-islands at which the state s is followed by the state t.

Markov chains for CG-islands and non CG-islands

```
# Markov chain for CpG islands
# Number of states:
4
# State labels:
A C G T
# Transition matrix P+:
.1795 .2735 .4255 .1195
.1705 .3665 .2735 .1875
.1605 .3385 .3745 .1245
.0785 .3545 .3835 .1815
```

$\boldsymbol{model}^{+}$

```
# Markov chain for non-CpG islands
# Number of states:
4
# State labels:
A C G T
# Transition matrix P-:
.2995 .2045 .2845 .2095
.3215 .2975 .0775 .0775
.2475 .2455 .2975 .2075
.1765 .2385 .2915 .2915
```

$\boldsymbol{model}^{-}$

Solving Problem 1 – discrimination

- Given a short sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_L)$. Does it come from a CG-island ($model^+$)?

$$P(\,\boldsymbol{x} \mid model^+\,) = \prod_{i=1}^{L} a^+_{x_{i-1,i}x_i}$$

- Or does it not come from a non-CG-island ($model^-$)?

$$P(\,\boldsymbol{x} \mid model^-\,) = \prod_{i=1}^{L} a^-_{x_{i-1,i}x_i}$$

- We calculate the log-odds ratio

$$S(\boldsymbol{x}) = \log \frac{P(\boldsymbol{x} \mid model^+)}{P(\boldsymbol{x} \mid model^-)} = \sum_{i=1}^{L} \log\left(\frac{a^+_{x_{i-1},x_i}}{a^-_{x_{i-1},x_i}}\right) = \sum_{i=1}^{L} \beta_{x_{i-1},x_i}$$

with β_{XY} being the log likelihood ratios of corresponding transition probabilities. For the transition matrices above we calculate for example $\beta_{AA} = \log(0.18/0.3)$. Often the base $2 \log$ is used, in which case the unit is in bits.

Solving Problem 1 – discrimination cont

- If $model^{+}$ and $model^{-}$ differ substantially then a typical CG-island should have a higher probability within the $model^{+}$ than in the $model^{-}$. The log-odds ratio should become positive.
- Generally we could use a threshold value c^{*} and a test function to determine whether a sequence x comes from a CG-island:

$$\phi^{*}(x) := \begin{cases} 1 & \text{if } S(x) > c^{*} \\ 0 & \text{if } S(x) \leq c^{*} \end{cases}$$

  where $\phi^{*}(x) = 1$ indicates that x comes from a CG-island.
- Such a test is called Neyman-Pearson-Test.

Outline

1. CG-islands
2. **The “Fair Bet Casino”**
3. Hidden Markov Model
4. Decoding Algorithm
5. Forward-Backward Algorithm
6. HMM Parameter Estimation
7. Viterbi training
8. Baum-Welch algorithm
9. Applications of HMM in Biology

CG Islands and the "Fair Bet Casino"

- The problem of localisations of CG-islands can be modeled after a problem named ***"The Fair Bet Casino"***
- The game is to flip coins, which results in only two possible outcomes: **H**ead or **T**ail.
- The **F**air coin will give **H**eads and **T**ails with same probability $\frac{1}{2}$.
- The **B**iased coin will give **H**eads with prob. $\frac{3}{4}$.
- Thus, we define the probabilities:
  - $P(H|F) = P(T|F) = \frac{1}{2}$
  - $P(H|B) = \frac{3}{4}, \quad P(T|B) = \frac{1}{4}$
  - The crooked dealer changes between Fair and Biased coins with probability 10%

The Fair Bet Casino Problem

- **Input:** A sequence $\boldsymbol{x} = x_1x_2x_3 \ldots x_n$ of coin tosses made by two possible coins (**F** or **B**).
- **Output:** A sequence $\boldsymbol{\pi} = \pi_1\pi_2\pi_3 \ldots \pi_n$, with each π_i being either F or B indicating that x_i is the result of tossing the **F**air or **B**iased coin respectively.

Fair Bet Casino Problem

Any observed outcome of coin tosses could have been generated by any sequence of states!

$\Rightarrow$ ***Ill formulated problem!***

Need to incorporate a way to grade different sequences differently.

$\Downarrow$

Decoding Problem

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

- Suppose first that dealer never changes coins. Some definitions:
  - $P(\boldsymbol{x}|fair\ coin)$: probability of the dealer using the F coin and generating the outcome $\boldsymbol{x}$.
  - $P(\boldsymbol{x}|biased\ coin)$: prob. of the dealer using the B coin and generating outcome $\boldsymbol{x}$.

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

$$\mathrm{P}(x \mid \text{fair coin}) = \mathrm{P}(x_1 \cdots x_n \mid \text{fair coin}) = \prod_{i=1}^{n} p(x_i \mid \text{fair coin}) = \left(\frac{1}{2}\right)^n$$

$$\mathrm{P}(x \mid \text{biased coin}) = \mathrm{P}(x_1 \cdots x_n \mid \text{biased coin}) = \prod_{i=1}^{n} p(x_i \mid \text{biased coin}) = \left(\frac{3}{4}\right)^k \left(\frac{1}{4}\right)^{n-k}$$

k – the number of H eads in x.

$P(\boldsymbol{x} \mid fair\ coin)$ vs. $P(\boldsymbol{x} \mid biased\ coin)$

$$P(x \mid fair\ coin) = P(x \mid biased\ coin)$$
$$\left(\frac{1}{2}\right)^n = \frac{3^k}{4^n}$$
$$2^n = 3^k$$
$$n = k \log_2 3$$

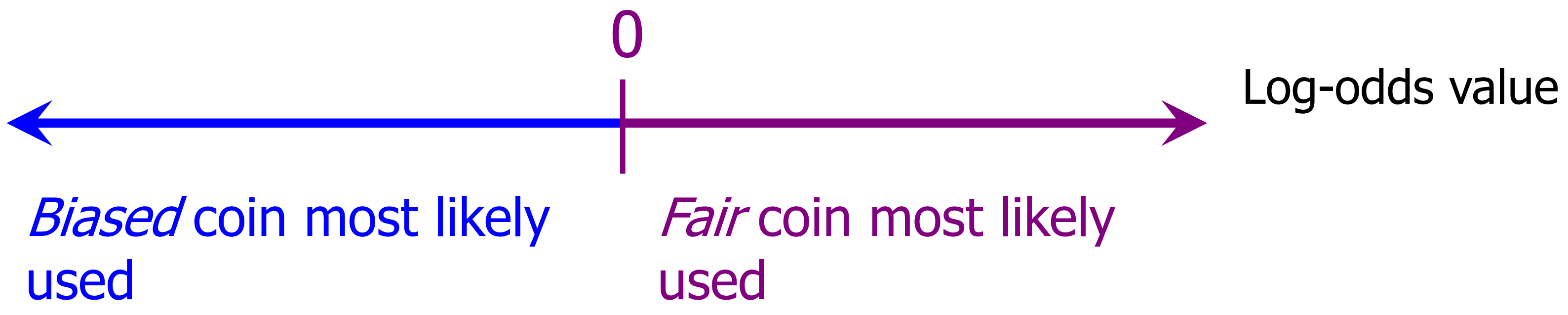
- when $k < n/\log_2 3$ $(k \sim 0.67n)$, the dealer most likely used the fair coin
- when $k > n/\log_2 3$, he most likely used the biased coin

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.

$$\log_2 \frac{P(window \mid fair\ coin)}{P(windows \mid biased\ coin)}$$



Disadvantages:
- the length of CG-island is not known in advance
- different windows may classify the same position differently

Outline

Hidden Markov Model (HMM)

- 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?
- Observer 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

  HHHTHTHHTTTTHTHTHTHHHTHTHTHT
  BBBFFFFFFFFFFFFFFFFFBBBFFFFF?

HMM Parameters $M(Q,\Sigma,A,E)$

Σ: a set of emission characters.

Ex.: $\Sigma = \{H, T\}$ for coin tossing

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

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

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

$A = (a_{kl})$: 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.

$$e_F(0) = ½ \quad e_F(1) = ½ \qquad 0 = Tail$$
$$e_B(0) = ¼ \quad e_B(1) = ¾ \qquad 1 = Head$$

HMM for Fair Bet Casino

- The *Fair Bet Casino* in *HMM* terms:
  $\Sigma = \{0, 1\}$ (0 for **T**ails and 1 **H**eads)
  $Q = \{F, B\}$ – F for Fair & B for Biased coin.

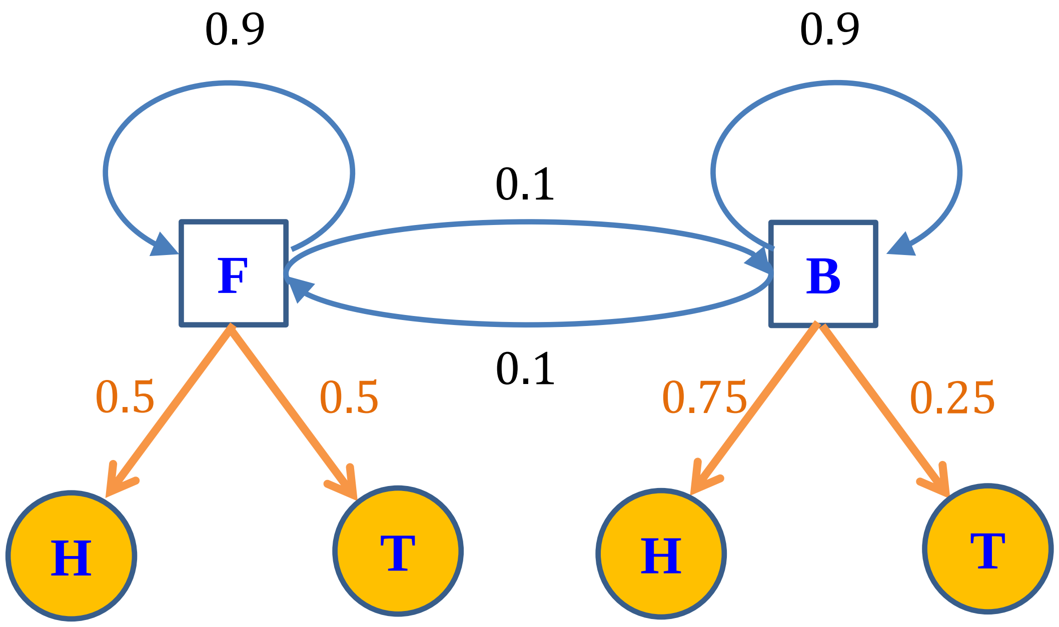
Transition Probabilities A

	Fair	Biased
Fair	$a_{FF} = 0.9$	$a_{FB} = 0.1$
Biased	$a_{BF} = 0.1$	$a_{BB} = 0.9$

Emission Probabilities E

	Tails(0)	Heads(1)
Fair	$e_F(0) = ½$	$e_F(1) = ½$
Biased	$e_B(0) = ¼$	$e_B(1) = ¾$

HMM for Fair Bet Casino
(cont'd)



HMM model for the *Fair Bet Casino* Problem

Hidden Paths

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

  sequence $\boldsymbol{x} = 01011101001$

Probability that x_i was emitted from state π_i

$$
\begin{array}{ll}
x & \left(\begin{array}{ccccccccccc} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
\pi \;\; = & F & F & F & B & B & B & B & B & F & F & F \\
P(x_i \mid \pi_i) & \tfrac{1}{2} & \tfrac{1}{2} & \tfrac{1}{2} & \tfrac{3}{4} & \tfrac{3}{4} & \tfrac{3}{4} & \tfrac{1}{4} & \tfrac{3}{4} & \tfrac{1}{2} & \tfrac{1}{2} & \tfrac{1}{2} \\
P(\pi_{i-1} \to \pi_i) & \tfrac{1}{2} & 9/10 & 9/10 & 1/10 & 9/10 & 9/10 & 9/10 & 9/10 & 1/10 & 9/10 & 9/10
\end{array}\right)
\end{array}
$$

Transition from the state $begin$

Transition probability from state π_{i-1} to state π_i

$P(\boldsymbol{x} \mid \boldsymbol{\pi})$ Calculation

- $P(\boldsymbol{x} \mid \pi)$: Probability that the sequence $\boldsymbol{x} = x_1 x_2 \ldots x_n$ was generated by the path $\boldsymbol{\pi} = \pi_1 \pi_2 \ldots \pi_n$:

$$P(\boldsymbol{x} \mid \boldsymbol{\pi}) = P(\pi_1)P(x_1 \mid \pi_1)P(\pi_1 \to \pi_2)P(x_2 \mid \pi_2)\cdots$$
$$P(x_{n-1} \mid \pi_{n-1})P(\pi_{n-1} \to \pi_n)P(x_n \mid \pi_n) =$$
$$= P(\pi_0 \to \pi_1)P(x_1 \mid \pi_1)P(\pi_1 \to \pi_2)P(x_2 \mid \pi_2)\cdots$$
$$P(x_{n-1} \mid \pi_{n-1})P(\pi_{n-1} \to \pi_n)P(x_n \mid \pi_n) =$$
$$= \prod_{i=1}^{n} P(\pi_{i-1} \to \pi_i) \cdot P(x_i \mid \pi_i)$$
$$= \prod_{i=1}^{n} a_{\pi_{i-1},\pi_i} \cdot e_{\pi_i}(x_i)$$

$$\pi_0 = begin$$
$$\pi_{n+1} = end$$

Outline

1. CG-islands
2. The “Fair Bet Casino”
3. Hidden Markov Model
4. **Decoding Algorithm**
5. Forward-Backward Algorithm
6. HMM Parameter Estimation
7. Viterbi training
8. Baum-Welch algorithm
9. Applications of HMM in Biology

Decoding Problem

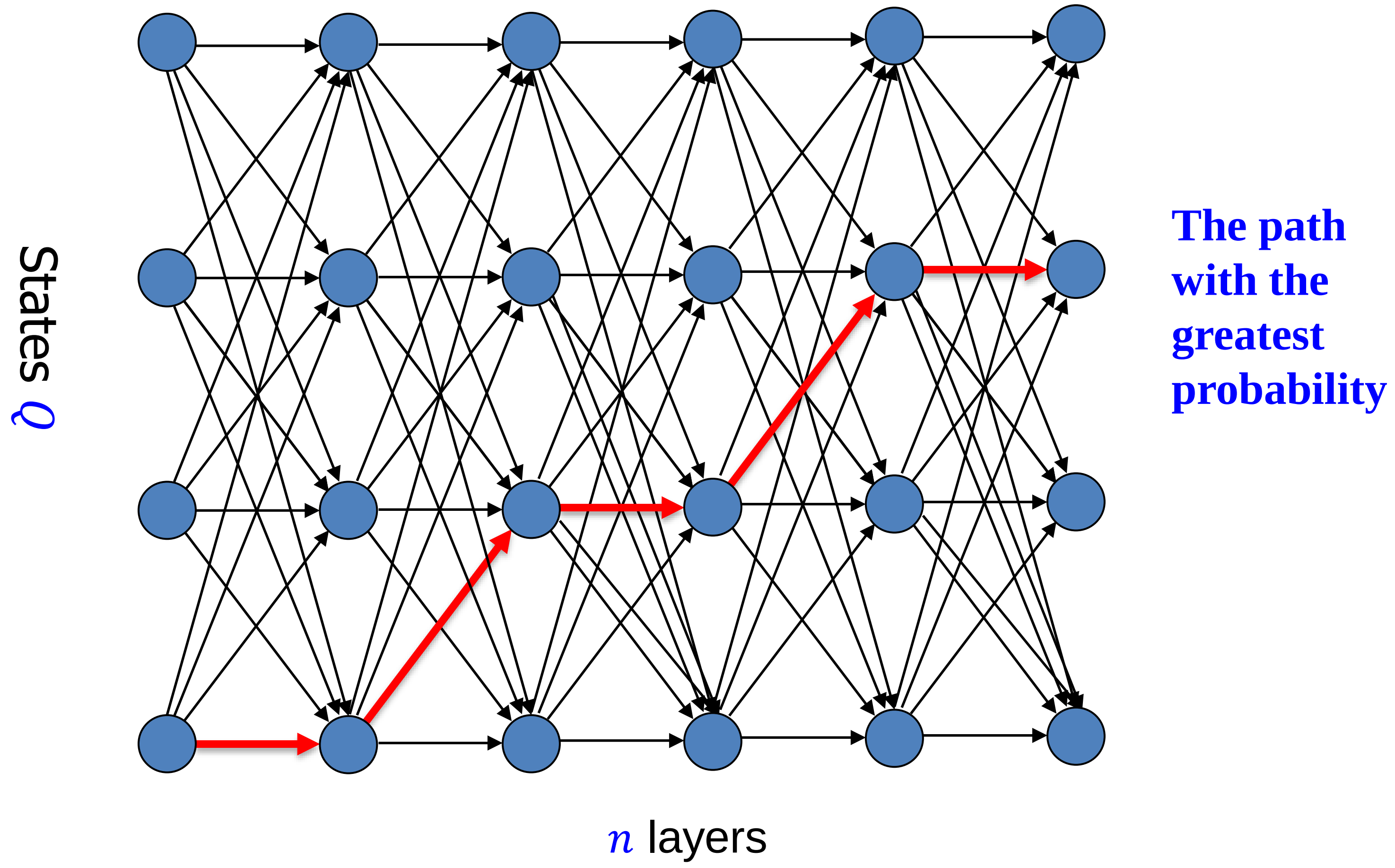
- **Goal:** Find an optimal hidden path of states given observations.
- **Input:** Sequence of observations $\boldsymbol{x} = x_1 \dots x_n$ generated by an HMM $M\ (\Sigma, Q, A, E)$
- **Output:** A path that maximizes $P(\boldsymbol{x} \mid \boldsymbol{\pi})$ over all possible paths $\boldsymbol{\pi}$.

$\Rightarrow$ *Solves Problem 2 - localization*

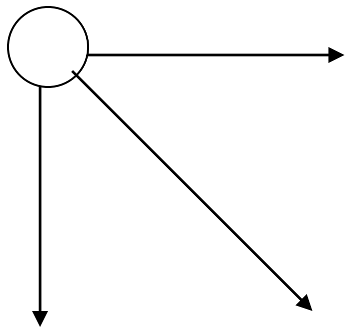
Building Manhattan for Decoding Problem

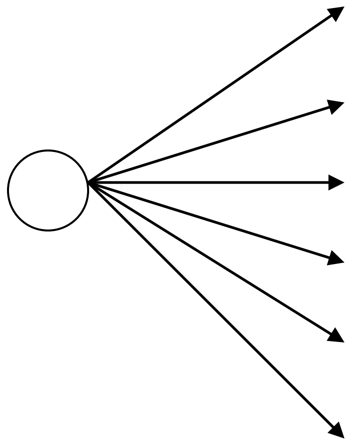
- Andrew Viterbi used the Manhattan grid model to solve the *Decoding Problem*.
- Every choice of $\boldsymbol{\pi} = \pi_1 \ldots \pi_n$ corresponds to a path in a graph.
- The only valid direction in the graph is *eastward.*
- This graph has $|Q|^2\,(n-1)$ edges.

Edit Graph for Decoding Problem



Decoding Problem vs. Alignment Problem





Valid directions in the *alignment problem.*

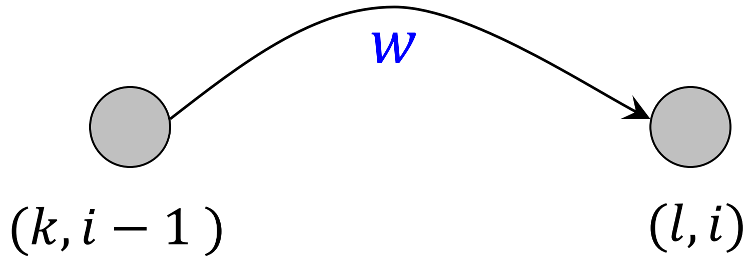
Valid directions in the *decoding problem.*

Decoding Problem as Finding a Longest Path in a DAG

- The *Decoding Problem* is reduced to finding a longest path in the *directed acyclic graph (DAG)* above.

- Notes: the length of the path is defined as the ***product*** of its edges' weights, not the ***sum.***
- Every path in the graph has the probability $P(\boldsymbol{x} \mid \boldsymbol{\pi})$.
- The Viterbi algorithm finds the path that maximizes $P(\boldsymbol{x} \mid \boldsymbol{\pi})$ among all possible paths.
- The Viterbi algorithm runs in $O(n|Q|^2)$ time.

Decoding Problem: weights of edges

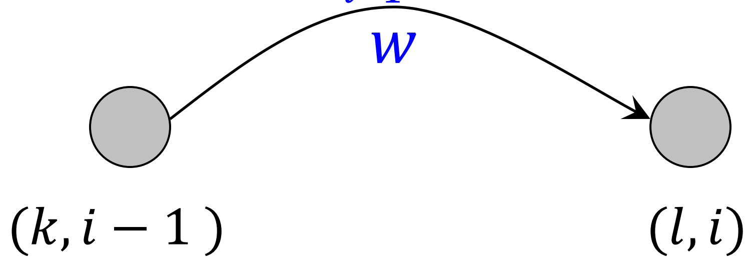


The weight w is given by:

$???$

Decoding Problem: weights of edges

$$P(\boldsymbol{x} \mid \boldsymbol{\pi}) = \prod_{i=1}^{n} a_{\pi_{i-1},\pi_i} \cdot e_{\pi_i}(x_i)$$

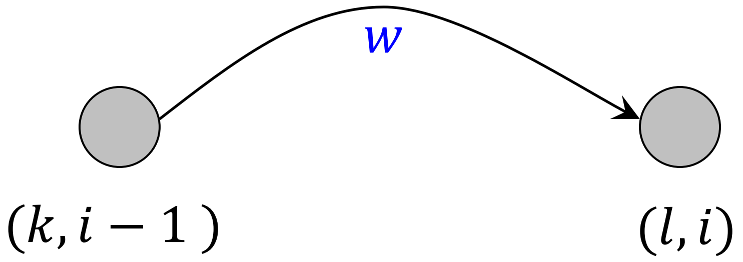


The weight w is given by:

$$??$$

Decoding Problem: weights of edges

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



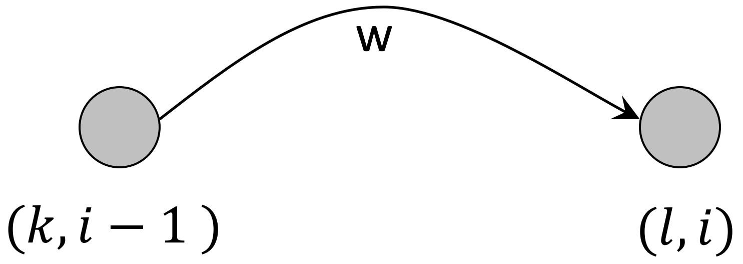
The weight w is given by:

?

Each weight is a factor in the product

Decoding Problem: weights of edges

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



The weight $w = e_l(x_i) \, . \, a_{kl}$

Decoding Problem and Dynamic Programming

Let s_{li} denote the probability of the most likely path generating the prefix $x_1, \dots, x_i$ and ending in state l

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

$$= \max_{k \in Q} \{s_{k,i-1} \cdot \quad a_{kl} \cdot e_l(x_i) \quad \}$$

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

Decoding Problem (cont'd)

- **Initialization**:
    - $s_{begin,0} = 1$
    - $s_{k,0} = 0$ for $k \neq begin$.

- Let $\boldsymbol{\pi}^*$ be the optimal path. Then,

$$P(\boldsymbol{x} \mid \boldsymbol{\pi}^*) = \max_{k \in Q} \{s_{k,n}\}$$

Viterbi Algorithm

- The value of the product can become extremely small, which leads to underflowing → use log value instead.
- **Goal:** Find an optimal hidden path of states given observations.
- **Input:** Sequence of observations $\boldsymbol{x} = x_1 \dots x_n$ generated by an HMM $M\ (\Sigma, Q, A, E)$
- **Output:** A path that maximizes $P(\boldsymbol{x} \mid \boldsymbol{\pi})$ over all possible paths $\boldsymbol{\pi}$.
- **Initialization**:

  $s_{begin,0} = \log 1 = 0$

  $s_{k,0} = \log 0 = -\infty$ for $k \neq begin$.
- **Iterate:**

```
For $i = 1$ to $n$
    For $l = 1$ to $|Q|$
        $s_{l,i} = \log e_l\,(x_i) + \max_{k \in Q} \{s_{k,i-1} + \log a_{kl}\}$  // note where the maximum was achieved
```

- **Output:** the sequence $\pi_1, \dots, \pi_n$ such that

$$s_{n,\pi_n} = \max_{k \in Q} s_{n,k} = \sum_{i=1,\dots,n} \left(\log e_{\pi_i} + \log a_{\pi_{i-1},\pi_i}\right)$$

Outline

1. CG-islands
2. The “Fair Bet Casino”
3. Hidden Markov Model
4. Decoding Algorithm
5. **Forward-Backward Algorithm**
6. HMM Parameter Estimation
7. Viterbi training
8. Baum-Welch algorithm
9. Applications of HMM in Biology

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

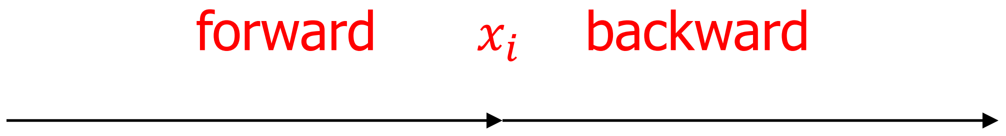
- Define $f_{k,i}$ (*forward probability*) as the probability of emitting the prefix $x_1 \dots x_i$ and reaching the state $\pi_i = k$.
- The recurrence for the forward algorithm:

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

Backward Algorithm

- However, *forward probability* is not the only factor affecting $P(\pi_i = k \mid \boldsymbol{x})$.

- The sequence of transitions and emissions that the HMM undergoes between π_{i+1} and π_n also affect $P(\pi_i = k \mid \boldsymbol{x})$.



Backward Algorithm (cont'd)

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

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

Backward-Forward Algorithm

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

$$P(\pi_i = k \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x}, \pi_i = k)}{P(\boldsymbol{x})} = \frac{f_{k,i} \cdot b_{k,i}}{P(\boldsymbol{x})}$$

$P(\boldsymbol{x})$ is the sum of $P(x, \pi_i = k)$ over all k

Outline

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.
- Given
  - HMM with states and alphabet (emission characters)
  - Independent training sequences $\boldsymbol{x}^{(1)}, \dots \boldsymbol{x}^{(m)}$

Sequences of different length

- Find HMM parameters Θ (that is, $a_{kl}, e_k(b)$) that maximize

$$P(\boldsymbol{x}^{(1)}, \dots \boldsymbol{x}^{(m)} \mid \Theta)$$

the joint probability of the training sequences.

Maximize the likelihood

$P(\boldsymbol{x}^{(1)}, \dots \boldsymbol{x}^{(m)} \mid \Theta)$ as a function of Θ is called the likelihood of the model.

The training sequences are assumed independent, therefore

$$P(\boldsymbol{x}^{(1)}, \dots \boldsymbol{x}^{(m)} \mid \Theta) = \prod_i P(\boldsymbol{x}^{(i)} \mid \Theta)$$

The parameter estimation problem seeks Θ that realizes

$$\max \prod_i P(\boldsymbol{x}^{(i)} \mid \Theta)$$

In practice the log likelihood is computed to avoid underflow errors

Two situations

Known paths for training sequences

- CpG islands marked on training sequences
- One evening the casino dealer allows us to see when he changes dice

Unknown paths

- CpG islands are not marked
- Do not see when the casino dealer changes dice

Known paths

A_{kl} = # of times each $k \rightarrow l$ is taken in the training sequences

$E_k(b)$ = # of times b is emitted from state k in the training sequences

Compute a_{kl} and $e_k(b)$ as maximum likelihood estimators:

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

A Parameter Estimations Approach

- If hidden states **were** known, we could use our training data to estimate parameters

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}, \qquad e_k\,(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

- However, usually the hidden state sequence π is not given, but only the observed output stream $\boldsymbol{x}$
- An alternative is to make an intelligent guess of π, use the equations above to estimate parameters, then run Viterbi to estimate the hidden state, then re-estimate the parameters and repeat until the state assignments or parameter values converge.
- Such iterative approaches are called Expectation Maximization (EM) methods of parameter estimation

Pseudocounts

- Some state k may not appear in any of the training sequences. This means $A_{kl} = 0$ for every state l and a_{kl} cannot be computed with the given equation.
- To avoid this overfitting use predetermined pseudocounts r_{kl} and $r_k(b)$.

$$A_{kl} = \text{\# of transitions } k \rightarrow l + r_{kl}$$

$$E_k(b) = \text{\# of emissions of } b \text{ from } k + r_k(b)$$

The pseudocounts can reflect our prior biases about the probability values.

Outline

1. CG-islands
2. The “Fair Bet Casino”
3. Hidden Markov Model
4. Decoding Algorithm
5. Forward-Backward Algorithm
6. HMM Parameter Estimation
7. **Viterbi training**
8. Baum-Welch algorithm
9. Applications of HMM in Biology

Unknown paths: Viterbi training

Idea: use Viterbi decoding to compute the most probable path for training sequence $\boldsymbol{x}$.

Start with some guess for initial parameters and compute $\boldsymbol{\pi}^*$ the most probable path for $\boldsymbol{x}$ using initial parameters.

Iterate until no change in $\boldsymbol{\pi}^*$:

- Determine A_{kl} and $E_k\,(b)$ as before
- Compute new parameters a_{kl} and $e_k\,(b)$ using the same formulas as before
- Compute new $\boldsymbol{\pi}^*$ for $\boldsymbol{x}$ and the current parameters

Viterbi training analysis

- The algorithm converges precisely.

  There are finitely many possible paths.

  New parameters are uniquely determined by the current $\boldsymbol{\pi}^*$.

  There may be several paths for $\boldsymbol{x}$ with the same probability, hence must compare the new $\boldsymbol{\pi}^*$ with all previous paths having highest probability.

- Does not maximize the likelihood $\prod_{\boldsymbol{x}} P(\boldsymbol{x} \mid \Theta)$ but the contribution to the likelihood of the most probable path $\prod_{\boldsymbol{x}} P(\boldsymbol{x} \mid \Theta, \boldsymbol{\pi}^*)$
- In general performs less well than Baum-Welch

Outline

Unknown paths: Baum-Welch

Idea:

1. Guess initial values for parameters.

   art and experience, not science

2. Estimate new (better) values for parameters.

   how?

3. Repeat until stopping criteria is met.

   what criteria?

Better values for parameters

- Would need the A_{kl} and $E_k(b)$ values but cannot count (the path is unknown) and do not want to use the most probable path.
- For all states k, l, symbol b and training sequence $\boldsymbol{x}$

Compute A_{kl} **and** $E_k(b)$ as expected values, given the current parameters

Notation

- For any sequence of characters $\boldsymbol{x}$ emitted along some unknown path $\boldsymbol{\pi}$, denote by $\pi_i = k$ the assumption that the state at position i (in which x_i is emitted) is k.

Probabilistic setting for A_{kl}

- Given $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}$ consider a discrete probability space with elementary events

$$\varepsilon_{k,l} = \text{“}k \to l \text{ is taken in } \boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\text{”}$$

- For each $\boldsymbol{x}$ in $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ and each position i in $\boldsymbol{x}$ let $Y_{x,i}$ be a random variable defined by

$$Y_{x,i}(\varepsilon_{k,l}) = \begin{cases} 1 & \text{if } \pi_i = k \text{ and } \pi_{i+1} = l \\ 0 & \text{otherwise} \end{cases}$$

- Define $Y = \sum_x \sum_i Y_{x,i}$ random variable that counts # of times the event $\varepsilon_{k,l}$ happens in $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}$.

The meaning of A_{kl}

Let A_{kl} be the expectation of Y

$$E(Y) = \sum_x \sum_i E(Y_{x,i}) = \sum_x \sum_i P(Y_{x,i} = 1) =$$

$$= \sum_x \sum_i P(\{\, \varepsilon_{k,l} \mid \pi_i = k \; and \; \pi_{i+1} = l \,\}) =$$

$$= \sum_x \sum_i P(\pi_i = k \; and \; \pi_{i+1} = l \mid \boldsymbol{x})$$

Need to compute $P(\pi_i = k \; and \; \pi_{i+1} = l \mid \boldsymbol{x})$

Probabilistic setting for $E_k(b)$

Given $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}$ consider a discrete probability space with elementary events

$$\varepsilon_{k,b} = \text{"}b \text{ is emitted in state } k \text{ in } \boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\text{"}$$

For each $\boldsymbol{x}$ in $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ and each position i in $\boldsymbol{x}$ let $Y_{x,i}$ be a random variable defined by

$$Y_{x,i}(\varepsilon_{k,b}) = \begin{cases} 1 & \text{if } x_i = b \text{ and } \pi_i = k \\ 0 & \text{otherwise} \end{cases}$$

Define $Y = \sum_x \sum_i Y_{x,i}$ random variable that counts # of times the event $\varepsilon_{k,b}$ happens in $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}$.

The meaning of $E_k(b)$

Let $E_k(b)$ be the expectation of Y

$$E(Y) = \sum_x \sum_i E(Y_{x,i}) = \sum_x \sum_i P\left(Y_{x,i} = 1\right) =$$

$$= \sum_x \sum_i P(\{\varepsilon_{k,b} \mid x_i = b \; and \; \pi_i = k \}) =$$

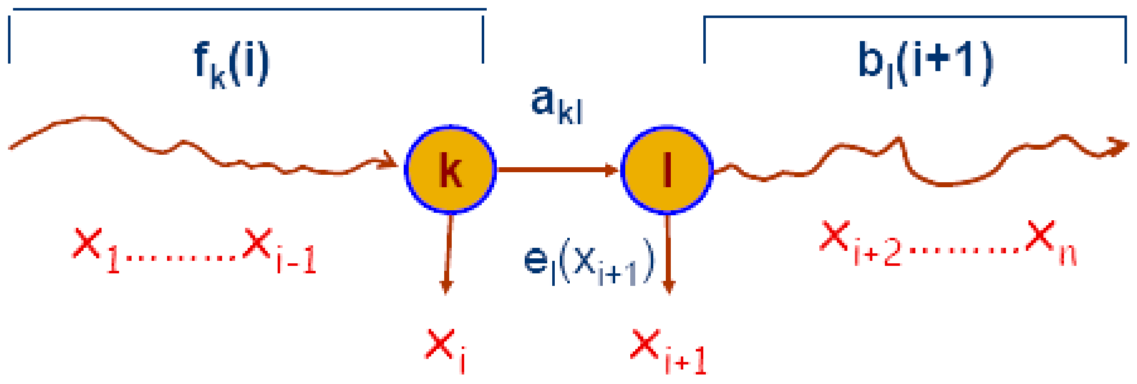
$$= \sum_x \sum_{\{i|x_i = b\}} P(\{\varepsilon_{k,b} \mid x_i = b, \pi_i = k\}) = \sum_x \sum_{\{i|x_i = b\}} P(\{\pi_i = k \mid \boldsymbol{x}\})$$

Need to compute $P(\pi_i = k \mid \boldsymbol{x})$

Computing new parameters

Consider $\boldsymbol{x} = x_1 \dots x_n$ training sequence

Concentrate on positions i and $i+1$



Use the forward-backward values:

$$f_{ki} = P(x_1 \dots x_i, \pi_i = k)$$

$$b_{ki} = P(x_{i+1} \dots x_n \mid \pi_i = k)$$

Compute A_{kl} (1)

- Prob $k \rightarrow l$ is taken at position i of $\boldsymbol{x}$

$$P(\pi_i = k, \pi_{i+1} = l \mid x_1 \ldots x_n) = P(\boldsymbol{x}, \pi_i = k, \pi_{i+1} = l)/P(\boldsymbol{x})$$

- Compute $P(\boldsymbol{x})$ using either forward or backward values

- We'll show that

$$P(\boldsymbol{x}, \pi_i = k, \pi_{i+1} = l) = b_{l,i+1} \cdot e_l\,(x_{i+1}) \cdot a_{kl} \cdot f_{ki}$$

- Expected # times $k \rightarrow l$ is used in training sequences

$$A_{kl} = \sum_x \sum_i \left(b_{i,i+1} \cdot e_l(x_{i+1}) \cdot a_{kl} \cdot f_{ki}\right)/P(\boldsymbol{x})$$

Compute A_{kl} (2)

$P(\boldsymbol{x},\pi_i = k,\pi_{i+1} = l) =$

$P(x_1 \ldots x_i, \pi_i = k, \pi_{i+1} = l, x_{i+1} \ldots x_n) =$

$P(\pi_{i+1} = l, x_{i+1} \ldots x_n \mid x_1 \ldots x_i, \pi_i = k) \cdot P(x_1 \ldots x_i, \pi_i = k) =$

$P(\pi_{i+1} = l, x_{i+1} \ldots x_n \mid \pi_i = k) \cdot f_{ki} =$

$P(x_{i+1} \ldots x_n \mid \pi_i = k, \pi_{i+1} = l) \cdot P(\pi_{i+1} = l \mid \pi_i = k) \cdot f_{ki} =$

$P(x_{i+1} \ldots x_n \mid \pi_{i+1} = l) \cdot a_{kl} \cdot f_{ki} =$

$P(x_{i+2} \ldots x_n \mid x_{i+1}, \pi_{i+1} = l) \cdot P(x_{i+1} \mid \pi_{i+1} = l) \cdot a_{kl} \cdot f_{ki} =$

$P(x_{i+2} \ldots x_n \mid \pi_{i+1} = l) \cdot e_l(x_{i+1}) \cdot a_{kl} \cdot f_{ki} =$

$b_{l,i+1} \cdot e_l(x_{i+1}) \cdot a_{kl} \cdot f_{ki}$

Compute $E_k(b)$

Probability x_i of x is emitted in state k

$$P(\pi_i = k \mid x_1 \ldots x_n) = P(\pi_i = k, x_1 \ldots x_n)/P(\boldsymbol{x})$$

$$P(\pi_i = k, x_1 \ldots x_n) = P(x_1 \ldots x_i, \pi_i = k, x_{i+1} \ldots x_n) =$$

$$P(x_{i+1} \ldots x_n \mid x_1 \ldots x_i, \pi_i = k) \cdot P(x_1 \ldots x_i, \pi_i = k) =$$

$$P(x_{i+1} \ldots x_n | \pi_i = k) \cdot f_{ki} = b_{ki} \cdot f_{ki}$$

Expected # times b is emitted in state k

$$E_k(b) = \sum_x \frac{\sum_{i:x_i=b}(f_{ki} \cdot b_{ki})}{P(\boldsymbol{x})} = \sum_x \sum_{i:x_i=b} \frac{f_{ki} \cdot b_{ki}}{P(\boldsymbol{x})}$$

Finally, new parameters

Can add pseudocounts as before.

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

Stopping criteria

Cannot actually reach maximum (optimization of continuous functions)
Therefore need stopping criteria.

- Compute the $\log$ likelihood of the model for current Θ

$$\sum_{\boldsymbol{x}} \log P(\boldsymbol{x} \mid \Theta)$$

- Compare with previous log likelihood.
- Stop if small difference.
- Stop after a certain number of iterations.

The Baum-Welch algorithm

Initialization:

Pick the best-guess for model parameters (or arbitrary)

Iteration:

1. Forward for each $\boldsymbol{x}$
2. Backward for each $\boldsymbol{x}$
3. Calculate $A_{kl}, E_k(b)$
4. Calculate new $a_{kl}, e_k(b)$
5. Calculate new log-likelihood

Until log-likelihood does not change much

Baum-Welch analysis

- Log-likelihood is increased by iterations

  Baum-Welch is a particular case of the EM (expectation maximization) algorithm
- Convergence to local maximum. Choice of initial parameters determines local maximum to which the algorithm converges

Implementation Issue 1: Scaling

- To compute $f_k(i)$ and $b_k(i)$, multiplication of a large number of terms (probability), value heads to 0 quickly, which exceed the precision range of any machine.
- The basic procedure is to multiply them by a scaling coefficient that is independent of i (i.e. it depends only on k). Logarithm cannot be used because of summation. But we can use

$$c_t = \frac{1}{\sum_{k=1}^{n} f_k(i)}$$

- c_t will be stored for the time points when the scaling is performed. c_t is used for both $f_k(i)$ and $b_k(i)$. The scaling factor will be canceled out for parameter estimation.
- For Viterbi algorithm, the use of logarithm is O.K.

Implementation Issue 2: Multiple Observation Sequence

- Denote a set of m observation sequences as $\boldsymbol{X} = [\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}]$. Assume the observed sequences are independent.
- The re-estimation of formulas for multiple sequences are modified by adding together the individual frequencies for each sequence.

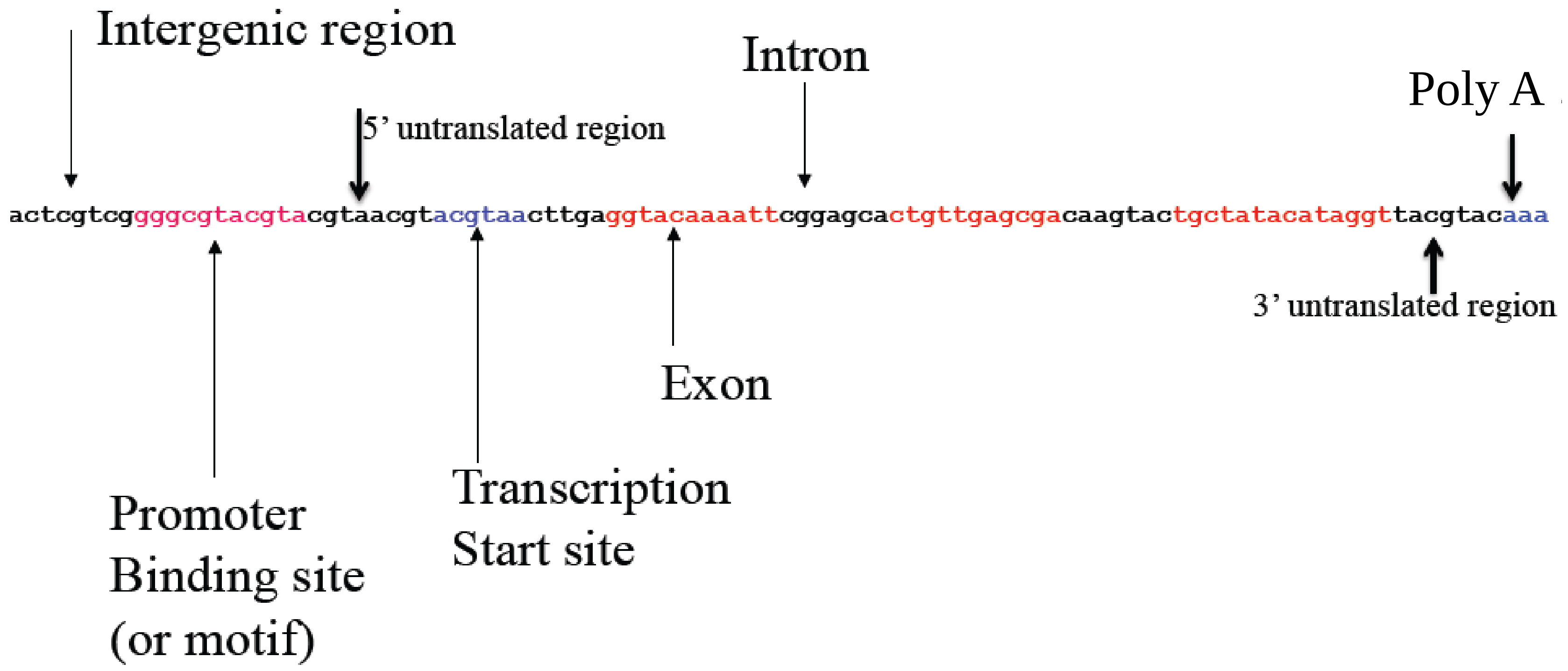
Outline

1. CG-islands
2. The “Fair Bet Casino”
3. Hidden Markov Model
4. Decoding Algorithm
5. Forward-Backward Algorithm
6. HMM Parameter Estimation
7. Viterbi training
8. Baum-Welch algorithm
9. **Applications of HMM in Biology**
   A. Finding genes
   B. Profile HMM
   C. Pairwise Alignment via HMM

Application of HMM in Biological Sequence Analysis

- Gene prediction
- Protein sequence modeling (learning, profile)
- Protein sequence alignment (decoding)
- Protein database search (scoring, e.g. fold recognition)
- Protein structure prediction
- …

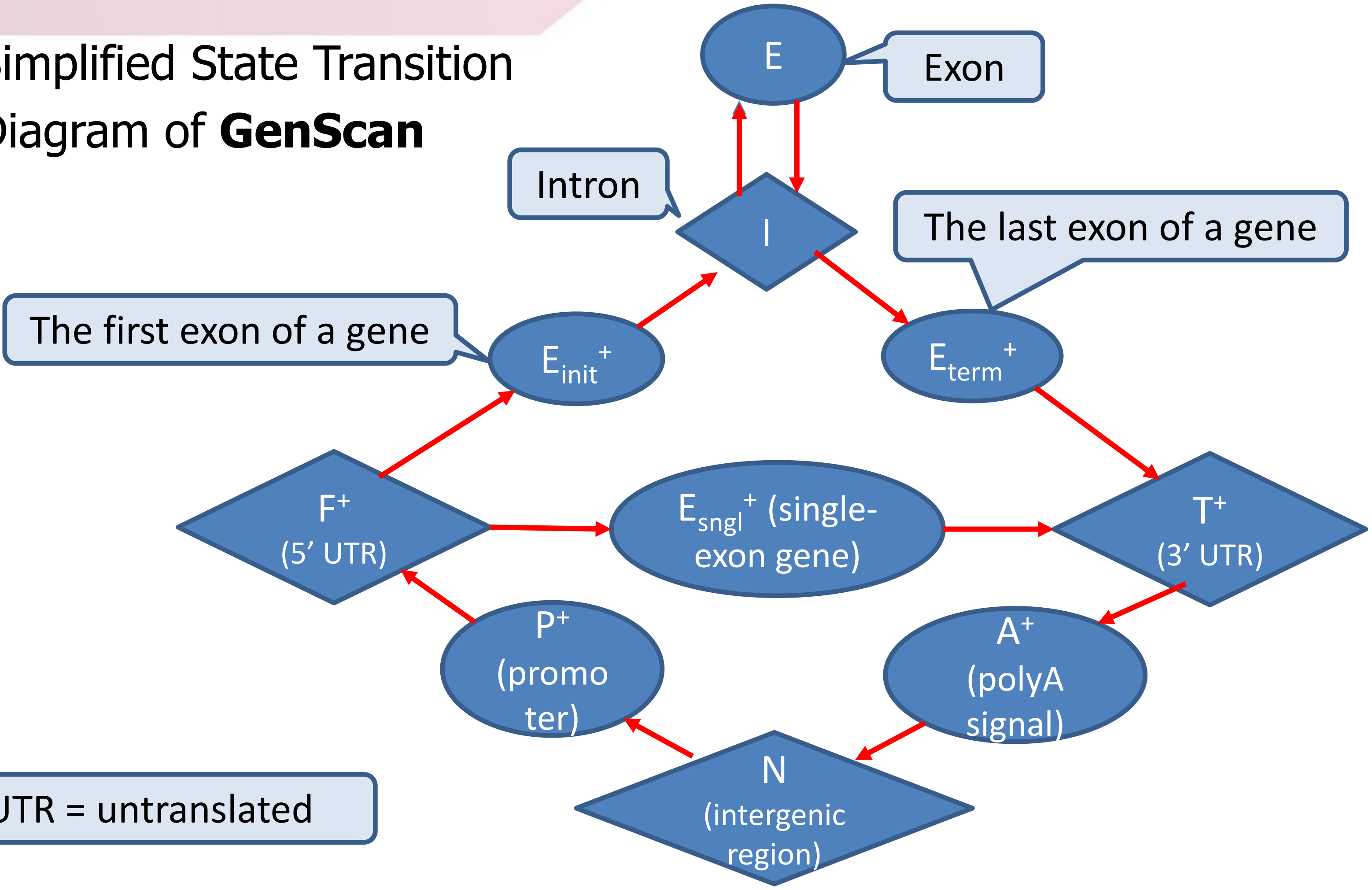
Motif and Gene Structure


Intergenic region
5' untranslated region
Intron
Poly A
actcgtcgggcgtacgtacgtaacgtacgtaacttgaggtacaaaattcggagcactgttgagcgacaagtactgctatacataggttacgtacaaa
3' untranslated region
Exon
Promoter Binding site (or motif)
Transcription Start site

- HMM has been used for modeling binding site and gene structure prediction.

GENSCAN (genes.mit.edu/GENSCAN.html)

- Simplified State Transition Diagram of **GenScan**



UTR = untranslated

Outline

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.
    - However multiple alignment is computationally expensive
- **A solution:** A family of related proteins can be represented by their multiple alignment and the corresponding profile.

Profile Representation of Protein Families

- Aligned DNA sequences (without gaps) can be represented by a $4 \times n$ profile matrix reflecting the frequencies of nucleotides in every aligned position.

$$\begin{array}{c|cccccccc}
\mathbf{A} & .72 & .14 & 0 & 0 & .72 & .72 & 0 & 0 \\
\mathbf{T} & .14 & .72 & 0 & 0 & 0 & .14 & .14 & .86 \\
\mathbf{G} & .14 & .14 & .86 & .44 & 0 & .14 & 0 & 0 \\
\mathbf{C} & 0 & 0 & .14 & .56 & .28 & 0 & .86 & .14
\end{array}$$

- Protein family can be represented by a $20 \times n$ profile representing frequencies of amino acids, but
  - an alignment can contain gaps and insertions
  - a profile does not preserve information about consecutive bases

Profiles and HMMs

- HMMs can also be used for aligning a sequence against a profile representing protein family.
- A $20 \times n$ profile P corresponds to n sequentially linked *match* states $M_1, \ldots, M_n$ in the profile HMM of P.
- Multiple alignment of a protein family shows variations in conservation along the length of a protein
- Example: after aligning many globin proteins, the biologists recognized that the helices region in globins are more conserved than others.

What are Profile HMMs?

- A Profile HMM is a probabilistic representation of a multiple alignment.
- A given multiple alignment (of a protein family) is used to build a profile HMM.
- This model then may be used to find and score less obvious potential matches of new protein sequences.

Multiple Sequence Alignment

- Based on a score matrix, sequences are aligned with gaps

	A	B	C	D	…	–
A	5	1	-2	-1	…	-5
B	1	6	4	-3	…	-5
C	-2	4	5	-4	…	-5
D	-1	-3	-4	4	…	-5

gap

- Unaligned sequences

```
AABNFCAQCDTYBNNBBTYANGC
AACFCBNFQADNNBCDTYBNANBAGC
```

- Alignment with gaps (indels) and mismatches

```
AABNFCA--QCDTYBNNBB-TY--AN--GC
AAC-FCBNFQAD---NNBCDTYBNANBAGC
```

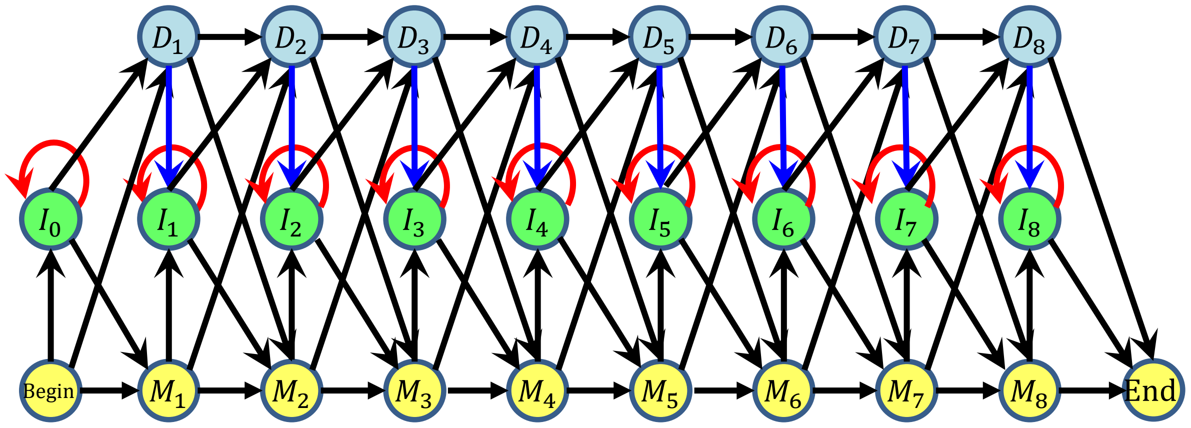
Multiple Sequence Alignment

- Dynamic programming too slow, use a heuristics
  1. Compute all pair alignments
  2. Compute maximum spanning tree
  3. Incrementally add sequences with the highest score according to the spanning tree

```
AA-BFFCA--QCDTYBNNBB-TY--ANGC
AAC-FFCANFQCD-Y-NNB-CTYBNANGC
CA-BFFCA--QCDTYBNNBB-TYBNAN-C
CAC-FCBANFQCD--BNNB-CTYBNANGC
CDBB-FBANFAC-QCDTYBCNTY--ANGC
CD-NC-BANFQCDQCDNNBCDTYBNANG-
-ABNCFCA--QCDTCBNNCCDTY--ANGC
AAC-CCB-NFQ-DDCDNNCCDTYBNANGC
```

Profile HMM

- Assign each column to a *Match* state in HMM. Add *Insertion* and *Deletion* state.
- Estimate the emission probabilities according to amino acid counts in column. Different positions in the protein will have different emission probabilities



D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8
I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8
Begin M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8 End

A profile HMM

Profile HMM from Multiple Sequence Alignment

- Less than half gaps in columns 1, 2, 6
  - Columns 1,2,6 are match states M_1, M_2, M_3
- Columns 3,4,5 more than half gaps
  - Create a single insert state I_2
- Emission probabilities
  - $e_{M_1}(B) = \frac{3}{3}$ $e_{M_1}(A) = \frac{0}{3}$ $e_{M_1}(T) = \frac{0}{3}$ …
- Zero probabilities cause problems (overfitting) – use Laplace correction (add 1; pseudocounts)
  - $e_{M_1}(B) = \frac{3+1}{3+20}$ $e_{M_1}(A) = \frac{0+1}{3+20}$ $e_{M_1}(T) = \frac{0+1}{3+20}$ …
- What are the emission probabilities $e_{M_2}(.)$, $e_{M_3}(.)$?

M_1	M_2	I_2			M_3
B	A	–	–	–	Q
B	A	G	–	C	Q
-	T	G	–	–	Q
B	T	–	F	–	Q
-	A	–	–	C	–
1	2	3	4	5	6

Profile HMM from Multiple Sequence Alignment

- Emission probabilities
  - $e_{I_2}(G) = \frac{2}{5}$ $e_{I_2}(A) = \frac{0}{5}$ $e_{I_2}(F) = \frac{1}{5}$ …
- zero probabilities cause problems (overfitting) – use Laplace correction (add 1; pseudocounts)
  - $e_{I_2}(G) = \frac{2+1}{5+20}$ $e_{I_2}(A) = \frac{0+1}{5+20}$ $e_{I_2}(F) = \frac{1+1}{5+20}$ …
- Transition probabilities
  - $a_{Begin,M_1} = \frac{3}{5}$ $a_{Begin,D_1} = \frac{2}{5}$ $a_{Begin,I_0} = \frac{0}{5}$ …
- zero probabilities cause problems (overfitting) – use Laplace correction (add 1; pseudocounts)
  - $a_{Begin,M_1} = \frac{3+1}{5+3}$ $a_{Begin,D_1} = \frac{2+1}{5+3}$

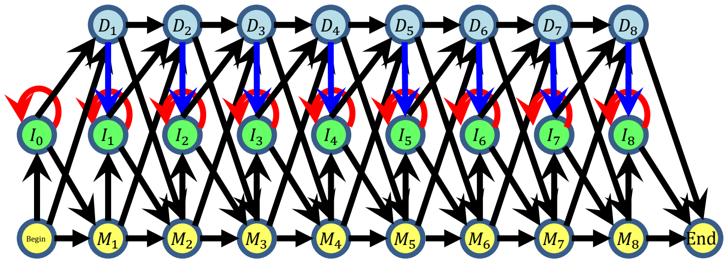
    $a_{Begin,I_0} = \frac{0+1}{5+3}$ …

M_1	M_2	I_2			M_3
B	A	–	–	–	Q
B	A	G	–	C	Q
-	T	G	–	–	Q
B	T	–	F	–	Q
-	A	–	–	C	–
1	2	3	4	5	6

Profile HMM from Multiple Sequence Alignment

- When there is no information in the alignment – set the probabilities to uniform
- I_1 does not appear in the alignment
  - $a_{I_1,M_2} = a_{I_1,I_1} = a_{I_1,D_2} = \frac{1}{3}$
- Transition from the delete state D_1 only into M_2
  - $a_{D_1,M_2} = \frac{5}{5}$ $a_{D_1,I_1} = \frac{0}{5}$ $a_{D_1,D_2} = \frac{0}{5}$
- Again add 1 to the counts
  - $a_{D_1,M_2} = \frac{5+1}{5+3}$ $a_{D_1,I_1} = \frac{0+1}{5+3}$ $a_{D_1,D_2} = \frac{0+1}{5+3}$
- What are the emission probabilities
  - $e_{D_1}(A) = ?$ $e_{D_1}(B) = ?$ …

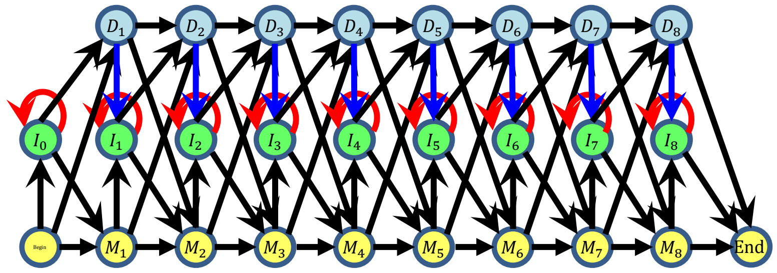
M_1	M_2	I_2			M_3
B	A	–	–	–	Q
B	A	G	–	C	Q
-	T	G	–	–	Q
B	T	–	F	–	Q
-	A	–	–	C	–
1	2	3	4	5	6



D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8
I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8
Begin M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8 End

Building a profile HMM

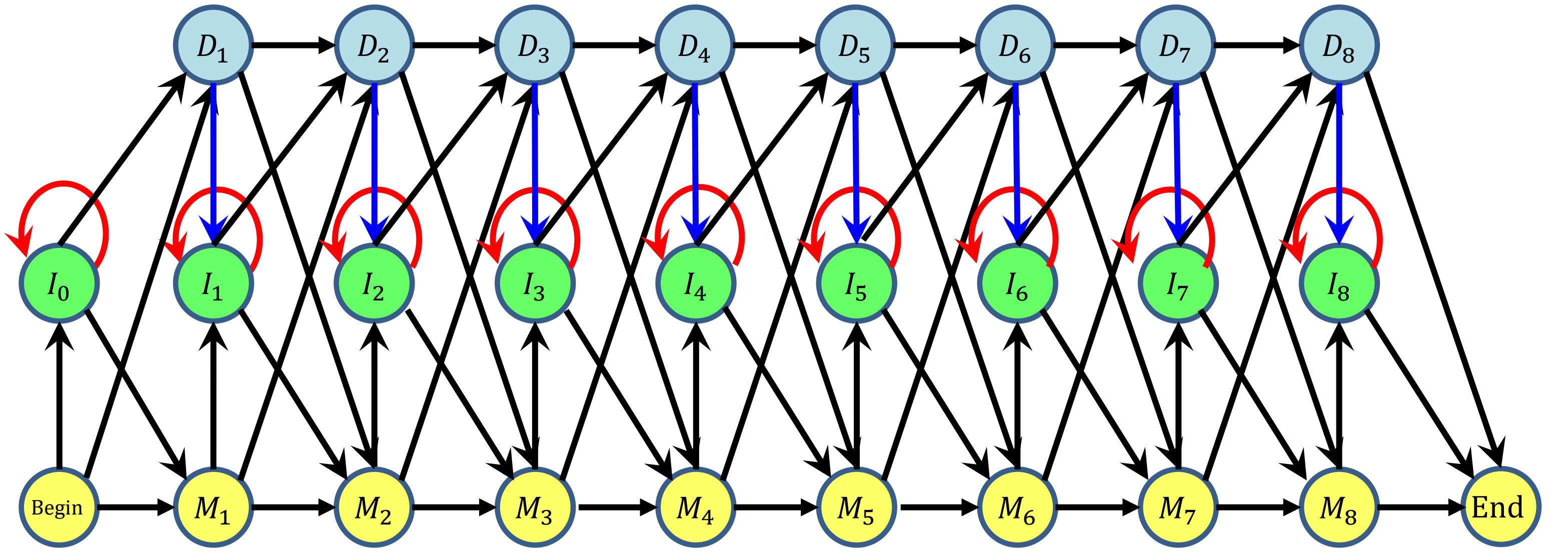
- Multiple alignment is used to construct the HMM model.
- Assign each column to a *Match* state in HMM. Add I*nsertion* and *Deletion* state.
- Estimate the emission probabilities according to amino acid counts in column. Different positions in the protein will have different emission probabilities.
- Estimate the transition probabilities between *Match, Deletion* and *Insertion* states
- The HMM model gets trained to derive the optimal parameters.



D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8
I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8
Begin M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8 End

States of Profile HMM

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



Probabilities in Profile HMM

- Transition probabilities:
  - $\log(a_{MI}) + \log(a_{DI})$ = gap open penalty
  - $\log(a_{II})$ = gap extension penalty

- Emission probabilities:
  - Probability of emitting a symbol a at an insertion state I_j:

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

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

Profile HMM Alignment

- Define $v_j^M(i)$ as the log-odds score of the best path for matching $x_1 \dots x_i$ to profile HMM ending with x_i emitted by the state M_j.
- $v_j^I(i)$ is the log-odds score of the best path ending in x_i being emitted by I_j and
- $v_j^D(i)$ is the log-odds score of the best path ending in state D_j.

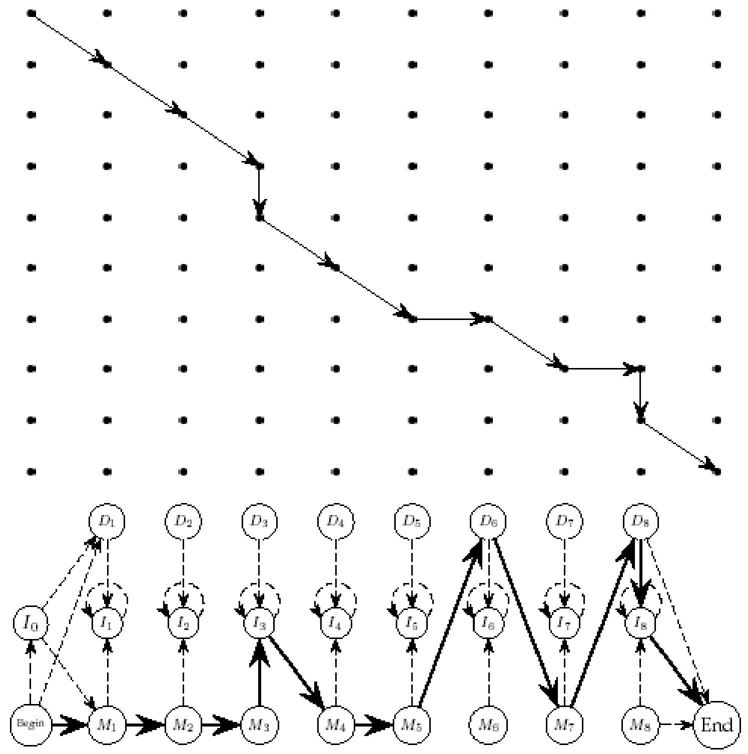
Profile HMM Alignment: Dynamic Programming

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

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

$$v_j^D(i) = \max\begin{cases} v_{j-1}^M(i-1) + \log\left(a_{M_{j-1},D_j}\right) \\ v_{j-1}^I(i-1) + \log\left(a_{I_{j-1},D_j}\right) \\ v_{j-1}^D(i-1) + \log\left(a_{D_{j-1},D_j}\right) \end{cases}$$

Paths in Edit Graph and Profile HMM



D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8
I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8
Begin M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8 End

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

Making a Collection of HMM for Protein Families

- Use Blast to separate a protein database into families of related proteins.
- Construct a multiple alignment for each protein family.
- Construct a profile HMM model and optimize the parameters of the model (transition and emission probabilities).
- Align the target sequence against each HMM to find the best fit between a target sequence and an HMM.

Application of Profile HMM to Modeling Globin Proteins

- Globins represent a large collection of protein sequences
- 400 globin sequences were randomly selected from all globins and used to construct a multiple alignment.
- Multiple alignment was used to assign an initial HMM
- This model then get trained repeatedly with model lengths chosen randomly between 145 to 170, to get an HMM model optimized probabilities.

How Good is the Globin HMM?

- 625 remaining globin sequences in the database were aligned to the constructed HMM resulting in a multiple alignment. This multiple alignment agrees extremely well with the structurally derived alignment.
- 25 044 proteins were randomly chosen from the database and compared against the globin HMM.
- This experiment resulted in an excellent separation between globin and non-globin families.

PFAM

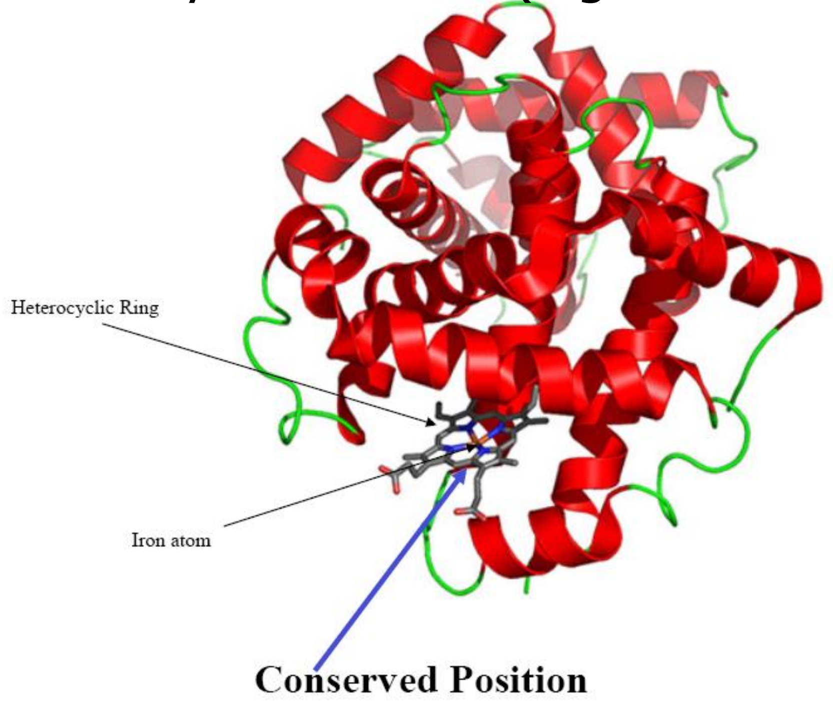
- Pfam http://pfam.xfam.org/ describes ***protein domains***
- Each protein domain family in Pfam has:
  - Seed alignment: manually verified multiple alignment of a representative set of sequences.
  - HMM built from the seed alignment for further database searches.
  - Full alignment generated automatically from the HMM
- The distinction between seed and full alignments facilitates Pfam updates.
  - Seed alignments are stable resources.
  - HMM profiles and full alignments can be updated with newly found amino acid sequences.

PFAM Uses

- Pfam HMMs span entire domains that include both well-conserved motifs and less-conserved regions with insertions and deletions.
- It results in modeling complete domains that facilitates better sequence annotation and leads to a more sensitive detection.

Model Protein Family (Profile HMM)

- Create a statistical model (HMM) for a group of related protein sequences (e.g., protein family)
- Identify core (conserved) elements of homologous sequences
- Positional evolutionary information (e.g. insertion and deletion)



Why do We Build a Profile (Model)?

- Understand the conservation (core function and structure elements) and variation
- Sequence generation
- Multiple sequence alignments
- Profile-sequence alignment (more sensitive than sequence-sequence alignment)
- Family/fold recognition
- Profile-profile alignment

Protein Family

```
seq1 VRRNNMGMPLIESSSYHDALFTLGYAGDRISQMLGMRLLAQGRLSEMAGADALDV
seq2 NIYIDSNGIAHIYANNLHDLFLAEGYYEASQRLFEIELFGLAMGNLSSWVGAKALSS
seq3 SAETYRDAWGIPHLRADTPHELARAQGTARDRAWQLEVERHRAQGTSASFLGPEALSW
seq4 DRLGVVTIDAANQLDAMRALGYAQERYFEMDLMRRAPAGELSELFGAKAVDL
```

```
seq1 ---VRRNNMGMPLIESSSYHDALFTLGY--AGDRISQMLGMRLLAQGRLSEMAGADALDV
seq2 --NIYIDSNGIAHIYANNLHDLFLAEGYYEASQRLFEIELFG-LAMGNLSSWVGAKALSS
seq3 SAETYRDAWGIPHLRADTPHELARAQGT--ARDRAWQLEVERHRAQGTSASFLGPEALSW
seq4 ------DRLGVVTIDAANQLDAMRALGY—AQERYFEMDLMRRAPAGELSELFGAKAVDL
```

- Imagine these sequences evolve from a single ancestral sequence and undergo evolutionary mutations. How to use a HMM to model?

Key to Build a HMM is to Set Up States

- Think about the positions of the ancestral sequence is undergoing mutation events to generate new sequences in difference species. A position can be modeled by a **dice.**

1. **Match** (match or mutate): the position is kept with or without variations/mutations.
2. **Delete**: the position is deleted
3. **Insert**: amino acids are inserted between two positions.

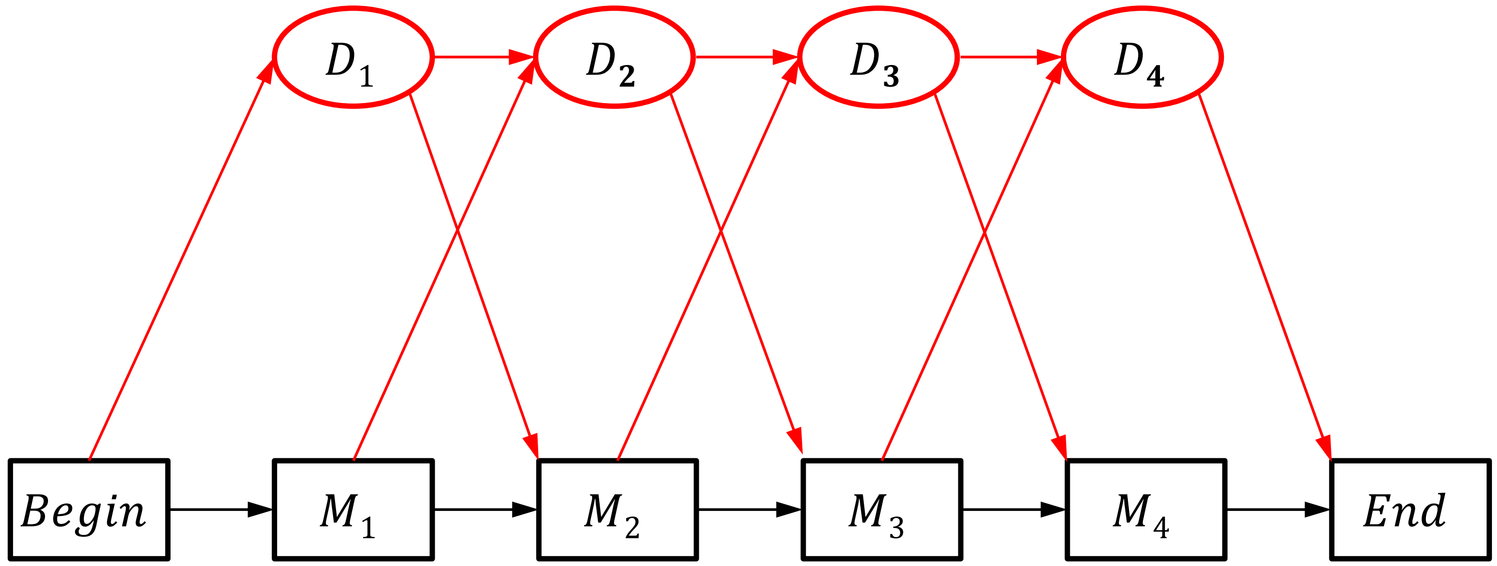
Hidden Markov Model

- Each match state has an emission distribution of 20 amino acids; one match state for a position.

$$\begin{tikzcd}
\boxed{Begin} \arrow[r] & \boxed{M_1} \arrow[r] & \boxed{M_2} \arrow[r] & \boxed{M_3} \arrow[r] & \boxed{M_4} \arrow[r] & \boxed{End}
\end{tikzcd}$$

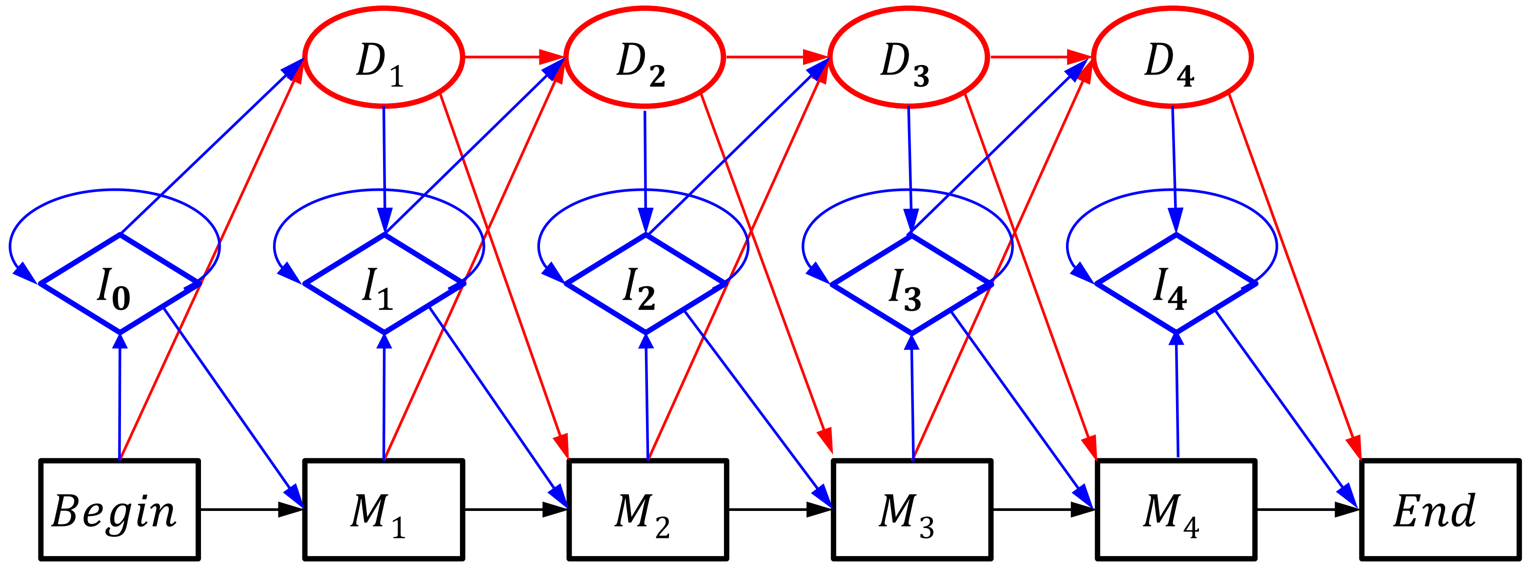
Hidden Markov Model

- Each match state has an emission distribution of 20 amino acids; one match state for a position.
- Deletion state is a mute state (emitting a dummy)



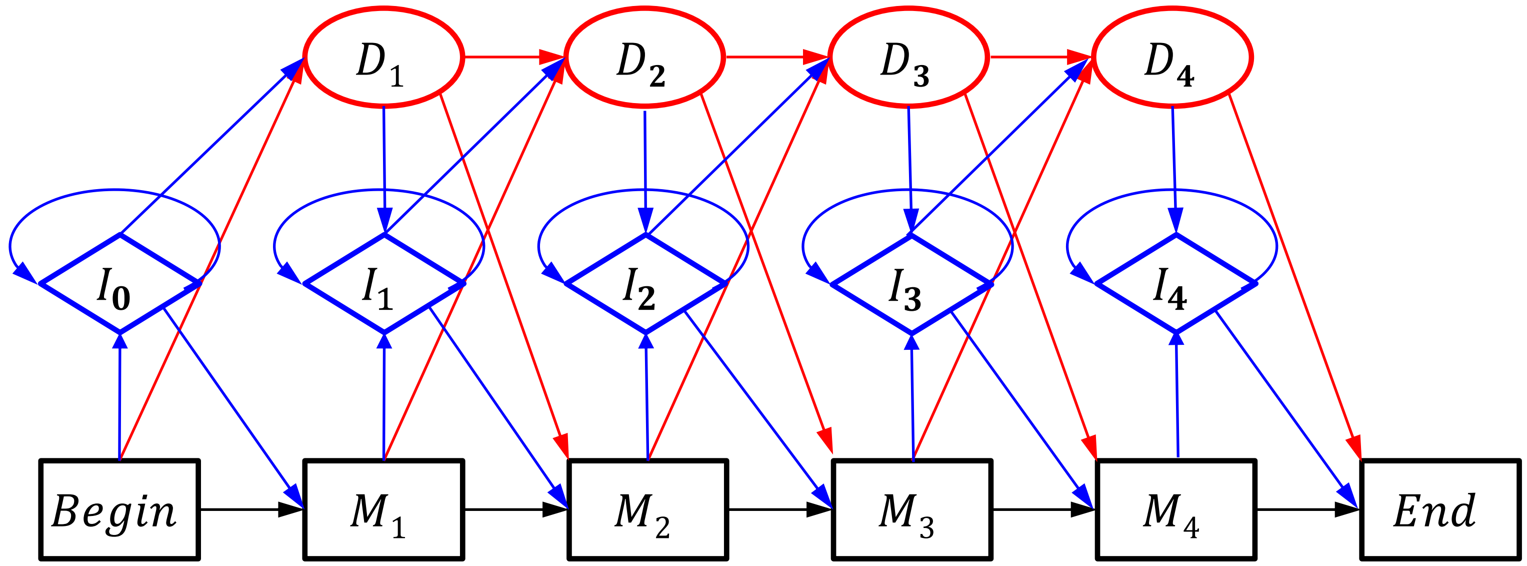
Hidden Markov Model

- Each match state has an emission distribution of 20 amino acids; one match state for a position.
- Deletion state is a mute state (emitting a dummy)
- Each insertion state has an emission distribution of 20 amino acids.



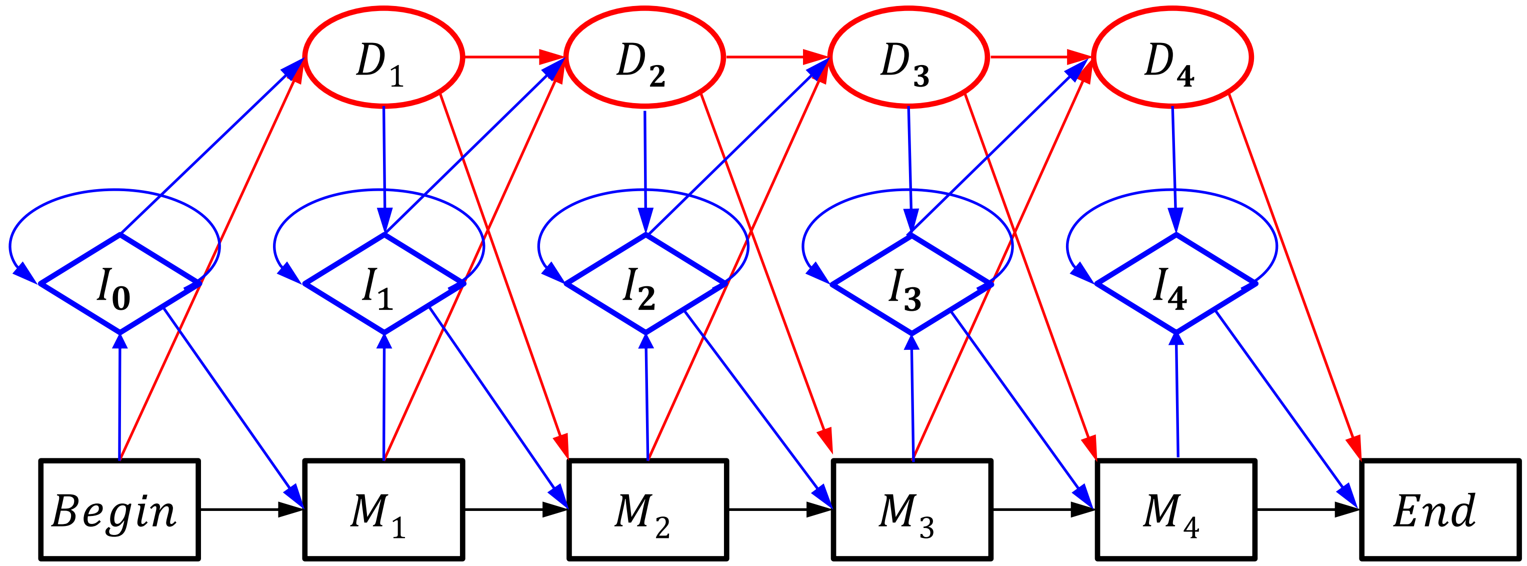
Hidden Markov Model

- How many states? (M positions: length of model)
  M(match)$+M$(deletion)$+(M+1)$(insertion)$+2=3M+3$



Hidden Markov Model

- How many transitions? (M positions = length of the model)
- Deletion: $3M-1$, Match: $3M-1$, Insertion: $3(M+1)-1$, B/E: 3
- Total $=9M+3$.



Initialization of HMM

- **How to decide model length (the number of match states)?**
  - **Learn:** Use a range of model lengths (centered at the average sequence length). If transition probability from a match (M_i) state to a delete state (D_{i+1}) > 0.5, remove the M_{i+1}. If transition probability from a match (M_i) state to an insertion state $(I_{i+1}) > 0.5$, add a match state.
  - **Get from multiple alignment:** assign a match state to any column with $< 50\%$ gaps.
- How to initialize transition probabilities?
- How to initialize emission probabilities?

Initialization of HMM

- How to decide model length (the number of match states)?
- **How to initialize transition probabilities?**
  - Uniform initialization of transition probabilities is O.K. in most cases.
- **How to initialize emission probabilities?**
  - Uniform initialization of emission probability of insert state is O.K. in many cases.
  - Uniform initialization of emission probability of match state is bad. (leads to **bad** local minima)
  - Using amino acid distribution to initialize the emission probabilities is better. (need regularization / smoothing to avoid zero)

Initialize from Multiple Alignments

```
seq1 ---VRRNNMGMPLIESSSYHDALFTLGY--AGDRISQMLGMRLLAQGRLSEMAGADALDV
seq2 --NIYIDSNGIAHIYANNLHDLFLAEGYYEASQRLFEIELFG-LAMGNLSSWVGAKALSS
seq3 SAETYRDAWGIPHLRADTPHELARAQGT--ARDRAWQLEVERHRAQGTSASFLGPEALSW
seq4 ------DRLGVVTIDAANQLDAMRALGY-AQERYFEMDLMRRAPAGELSELFGAKAVDL
```

- First, assign match/main states, delete states, insert states from multiple sequence alignment
- Get the path of each sequence
- Count the amino acid frequencies emitted from match or insert states, which are converted into probabilities for each state (need smoothing/regularization/pseudo-count).
- Count the number of state transitions and use them to initialize transition probabilities.

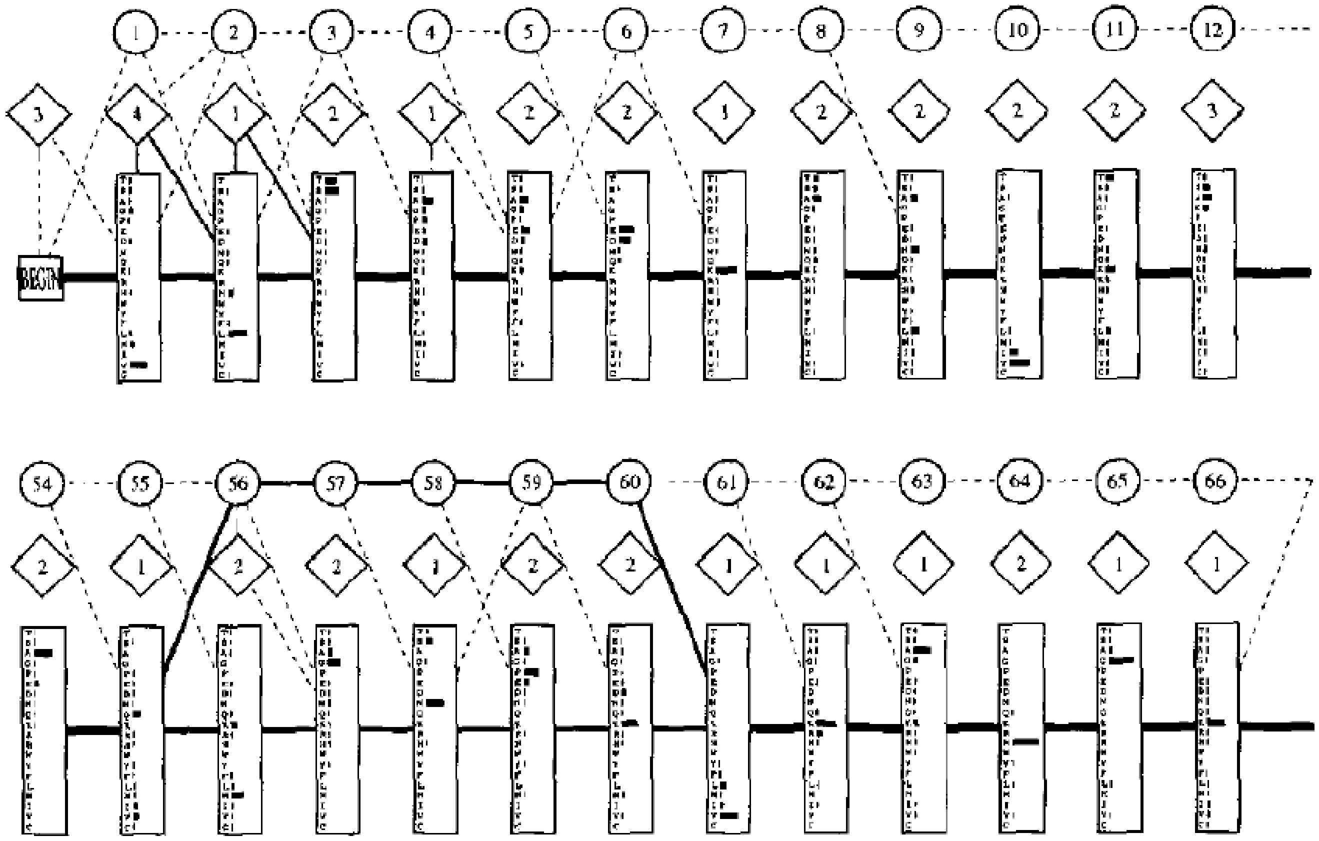
Estimate Parameters (Learning)

- We want to find a set of parameters to maximize the probability of the observed sequences in the family:
  - maximum likelihood:

$$P(\boldsymbol{x} \mid \Theta) = P\left(\boldsymbol{x}^{(1)} \mid \Theta\right) \cdot \ldots \cdot P(\boldsymbol{x}^{(m)} \mid \Theta).$$

- Baum-Welch's algorithm (or EM algorithm) (see above slides)

Visualization of Features and Structure in HMM



- Myoglobin protein family. How to interpret it? [Krogh et al. 94]

Protein Family Profile HMM Databases

- Pfam database (http://xpfam.pfam.org)
- PROSITE profiles database (http://prosite.expasy.org/) – protein domains, families and functional sites as well as associated patterns and profiles to identify them

- **What Can We Do With the HMM?**
  - **Recognition and classification:**
    - Widely used for database search: does a new sequence belong to the family? (database search)
    - **Idea:** The sequences belonging to the family (or generated from HMM) should receive higher probability than the sequence not belong to the family (unrelated sequences).

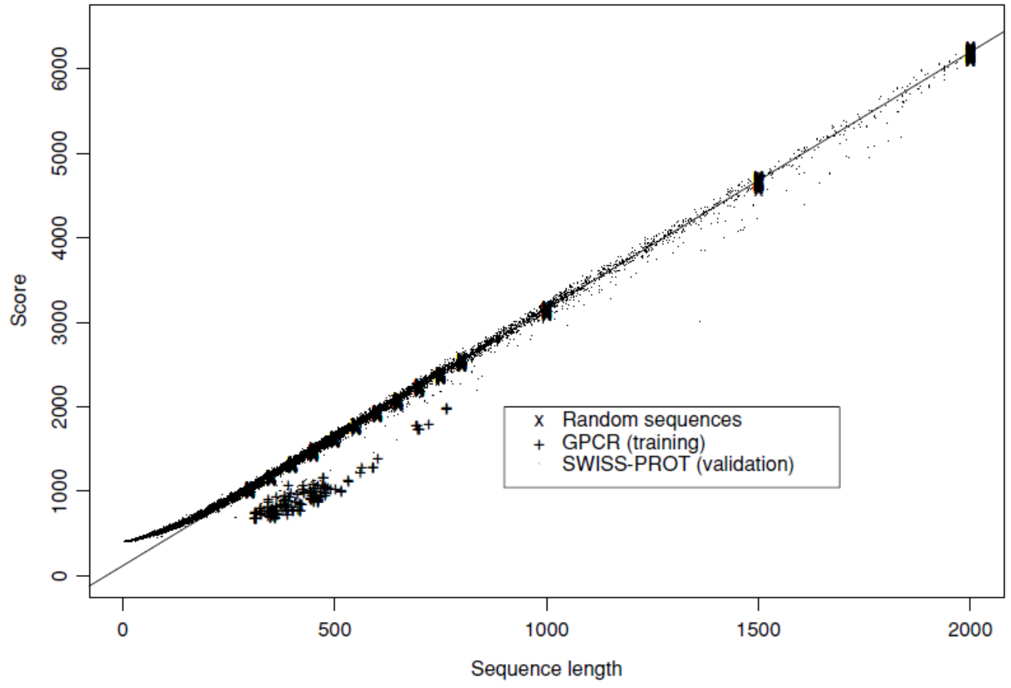
Two Ways to Search

1. Build a HMM for each family in the database. Search a query sequence against the database of HMMs. (Pfam)
2. Build a HMM for a query family, and search HMM against of a database of sequences

Compute $P(Sequence \mid HMM)$

- Forward algorithm to compute $P(\boldsymbol{x} \mid \Theta)$
- We work on: $-\log(P(\boldsymbol{x} \mid \Theta))$: distance from the sequence to the model. (**N**egative **L**og **L**ikelihood score – NLL)
- Unfortunately, $-\log(P(\boldsymbol{x} \mid \Theta))$ is length dependent. So what can we do?
    - Normalize the Score into Z-score
        - Search the profile against a large database such as Swiss-Prot
        - Plot $-\log(P(\mathbf{x} \mid \Theta))$, NLL scores, against sequence length.

Normalize the Score into Z-score

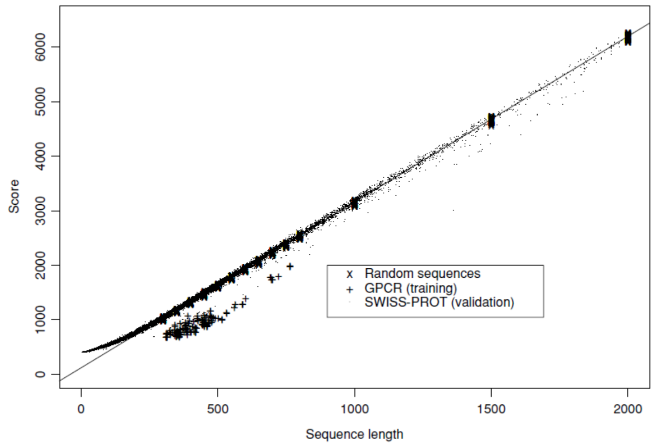


Example: **G-Protein-Coupled Receptors**

- Transmembrane proteins for signaling between environment and a cell

- NLL score is linear to sequence length.
- NLL scores of the same family is lower than un-related sequences
- We need normalization.

Normalize the Score into Z-score



- NULL model of unrelated sequences:

Length	Mean NLL (μ)	Std (σ)
100	500	5
101	550	6
…		
…		

- Compute Z-score: $\frac{|s-\mu|}{\sigma}$
- $Z > 4$: the sequence is very different from unrelated sequence (for non-database search, a randomization can work)

Pairwise Alignment via HMM

Seq 1: A T G R K E

Path : $M_1\ I_1\ I_1\ M_2\ D_3\ M_4\ I_4$

Seq 2: V C K E R P

Path : $M_1\ I_1\ M_2\ M_3\ M_4\ I_4$

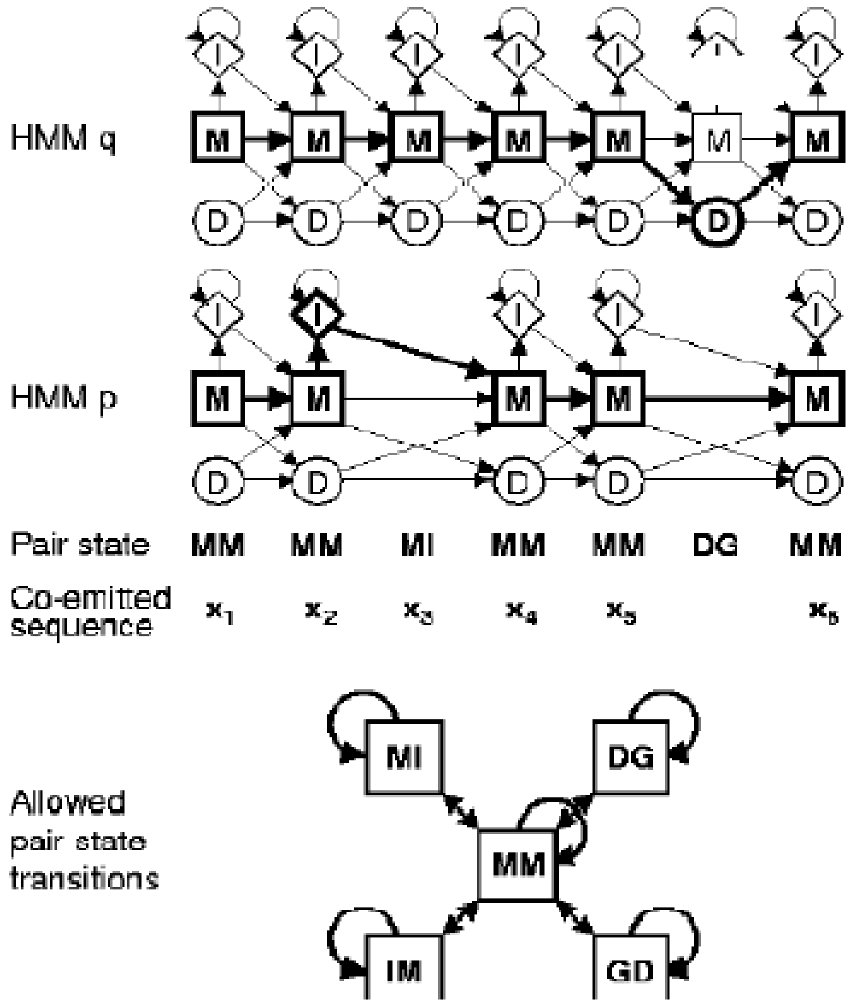
$$\downarrow$$

Path:	M_1			M_2	M_3	M_4	
Seq 1:	A	T	G	R	-	K	E
Seq 2:	V	C	-	K	E	R	P

HMM for Multiple Sequence Alignment

- Build a HMM for a group of sequences
- Align each sequence against HMM using Viterbi algorithm to find the most likely path. (dynamic programming)
- Match the main/match states of these paths together.
- Add gaps for delete states
- For insertion between two positions, use the longest insertion of a sequence as template. Add gaps to other sequence if necessary.

Similarity between HMMs



Generalized Sequence (of States) 1

Generalized Sequence (of States) 2

Goal: find a sequence of paired states with Maximum sum of log odds scores

J. Soding, Bioinformatics, 2005

COACH Approach

- COACH stands for Comparison Of Alignments by Constructing HMMs
- Given two families of sequences, build a multiple alignment (MSA) for each one of them.
- Build HMM from one MSA
- Align another MSA against the HMM (match each column of amino acids against states in the HMM)
- **How to do Local Alignment:**
  - With respect to sequence: add an insertion state right after the start state and right before the end state.
  - With respect to HMM: start state can jump to any match state and any match state can jump to end state.

HMM Software and Code

- HMMER: http://hmmer.org – biosequence analysis using profile hidden Markov models
- The MPI Bioinformatics Toolkit http://toolkit.tuebingen.mpg.de many tools **HH**xxxx (based on **H**MM-**H**MM comparison)
- PRC-HMM – the profile comparer: http://supfam.mrc-lmb.cam.ac.uk/PRC/
- COACH: profile-profile alignment of protein families using hidden Markov models : http://www.drive5.com/lobster/
- HMMCOMP – HMM-HMM comparison: http://users-cs.au.dk/cstorm/hmmcomp/
- MUSCLE – multiple alignment software: http://www.drive5.com/muscle/