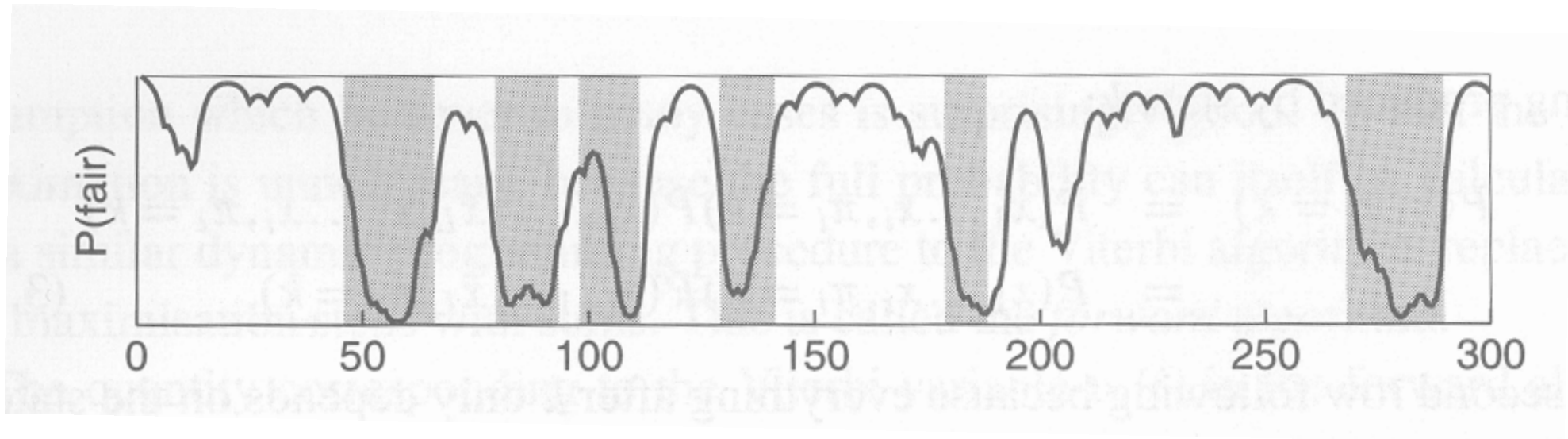


Chapter 2

Hidden Markov Models

General Introduction



Caveats of optimal alignment algorithms

- **All** pairs of sequences have an optimal alignment, **whether the sequences are related or not.**
- Optimal alignment is **not necessarily unique.** Many possible alignments may be statistically indistinguishable.
- There is, however, **only one biologically relevant alignment,** the one that traces the evolutionary descent of the two sequences.
- ...a more probabilistic treatment is desirable...

Hidden Markov Models

- Hidden Markov Models are very general **probabilistic models for sequences.**
- Typical questions for a given sequence:
 - Does a given (protein) sequence belong to a particular family (of proteins) ?
 - Assuming the protein does come from some family, what can we say about its internal (secondary) structure ?
- HMMs are powerful tools for **probabilistic sequence alignment**, overcoming most of the shortcomings of classical algorithms.

Markov Chains

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 ;

$P = P(x_1)$ is the **initial state probabilities**.

A is the **state transition probabilities**,

denoted by a_{st} for each $s, t \in Q$: $a_{st} := P(x_i = t | x_{i-1} = s)$.

Central property of Markov chains: Probability of symbol x_i depends only on the value of the preceding symbol x_{i-1} :

$$P(x_i | x_1, \dots, x_{i-1}) = P(x_i | x_{i-1}) = a_{x_{i-1}x_i}$$

M-chains are random processes with memory length 1.

Markov Chains

- Joint probability of two RVs A and B factorizes as $P(A, B) = P(B|A)P(A)$.
- **Total probability of sequence** $X = (x_1, \dots, x_L)$:

$$\begin{aligned} P(X) &= P(x_L|x_{L-1}, \dots, x_1) \cdot P(x_{L-1}|x_{L-2}, \dots, x_1) \cdots P(x_1) \\ &= P(x_L|x_{L-1}) \cdot P(x_{L-1}|x_{L-2}) \cdots P(x_1) \\ &= P(x_1) \prod_{i=2}^L a_{x_{i-1}x_i} \end{aligned}$$

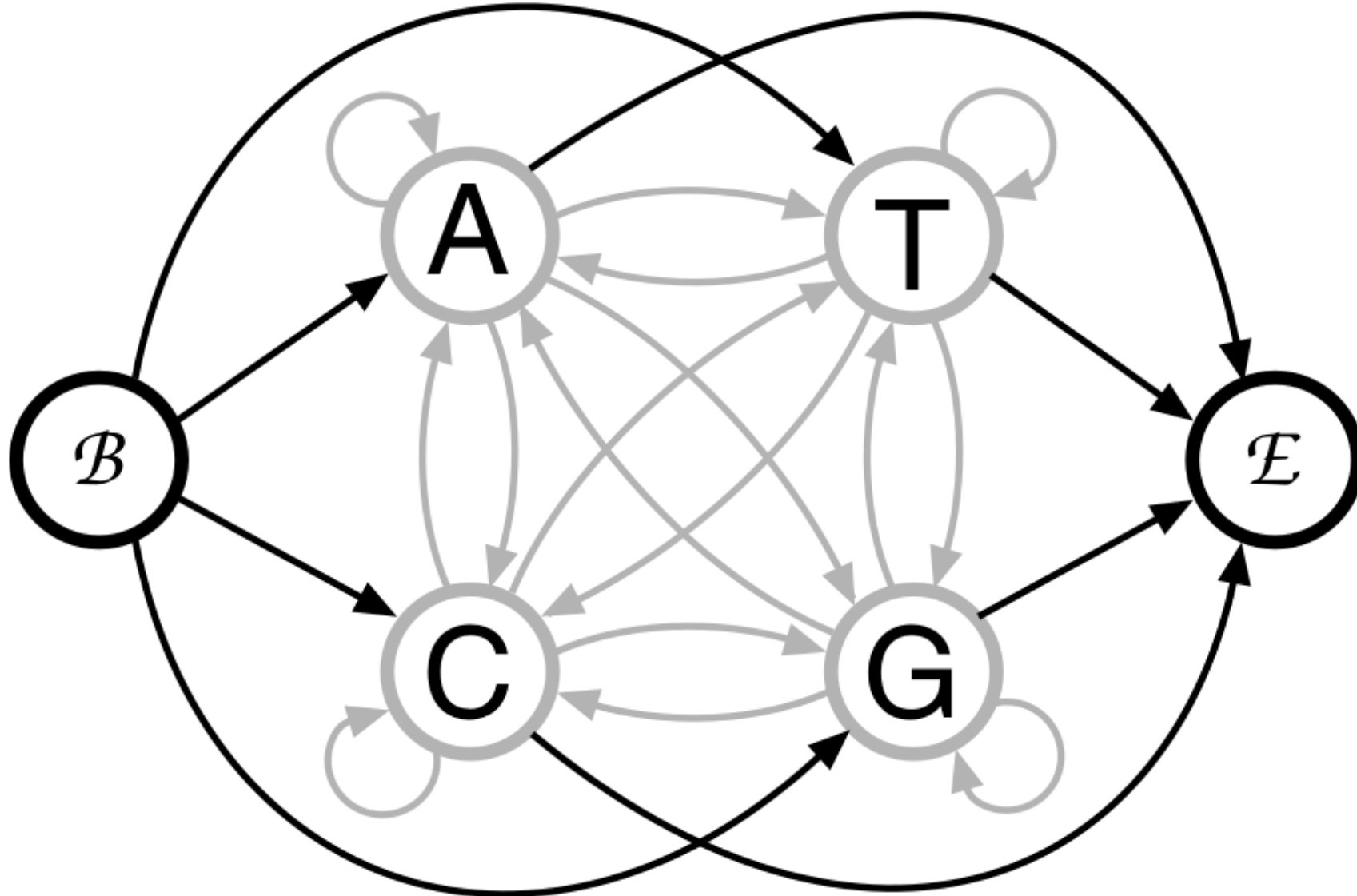
- **Silent states:**

Begin state: $x_0 := \mathcal{B} \Rightarrow P(x_1 = s) = a_{\mathcal{B}s}$

End state: $x_{L+1} = \mathcal{E} \Rightarrow P(\mathcal{E}|x_L = t) = a_{t\mathcal{E}}$

$$P(X) = \prod_{i=1}^{L+1} a_{x_{i-1}x_i}$$

A Markov chain for DNA sequences



Durbin et al., Cambridge University Press. <https://doi.org/10.1017/CBO9780511790492.004>

Example: CpG islands

- CpG denotes the **dinucleotide** CG:
...CATTCATCGCATTCTTTGGCAGGCGGAGGGGAAGCCT...
- For chemical reasons, CpG is relatively rare in most DNA sequences (as compared to the product of independent probabilities $P(C)$ and $P(G)$).
- However, in particular short subsequences, called **CpG islands**, the couple CG is more frequent.
- **CpG islands** are interesting, since they are known to appear in **more significant regions** of the genome, such as around the **start regions of genes**.

Example: CpG islands

CATTCCGCTTCTCTCCCGAGGTGGCGCGTGGGA
GGTGTGTTTGGCTCGGGTTCTGTAAGAATAGGCCAGG
CAGCTTCCCGCGGGATGCGCTCATCCCCTCTCGG
GGTTCGCTCCACCGCGCGCGTTGCGCGGT
CGCCTGCGAGATGTTTTCCAGCGACAATGATTC
CACTCTCGGCGCCTCCATGTTGATCCAGCTCCT
CTGCGGGCGTCAGGACCCCTGGGCCCGCCC
CTCCACTCAGTCAATCTTTGTCCCGTATAAGGCG
GATTATCGGGGTGGCTGGGGGCGGCTGATTCGGA
CGAATGCCCTTGGGGGTCACC CGGAGGGAACTC
CGGGCTCCGGCTTTGGCCAGCCCGCACCCCTGGT
TGAGCCGGGCCCGAGGGCCACCAGGGGGCGCTCG
ATGTTCTGCAGCCCCCGCAGCAGCCCCACTCC
CGCGCTCACCCCTACGATTGGCTGGCCGCCC
CTCTGTGCTGTGATTGGTCACAGCCCGTGTCCCGTC
GCGGGCGCGGGGCGGATA CGAGGTGACCGCGCA
GAGGCCAGCTCGGGCGGTGTCCCGCGCGCGC
GACTGCGGGCGGAGTTTCCGCGAGGGCCGAAGCG
GGGCAGTGTGACCGGCAGCGGTCCTGGGAGGCGC
CGCGCGCGCTCGGAGCAGCTCCC CGTCCTCCGCA
GC CGTCAACCGCCCGCGTCCCGCCCTGGCC
TCCCGCACTCGCGCACTCCTGTCCCGCCACC
GCCACCTCCCACCTCGATGCGGTGCCCGGGCTGC
TGCGTGATGGGGCTGCGGAGCGGCGCCCTGCGG
CTCGCGGCGGC CGCTGCTCGCGCTGAGGTGCGT
CGGTGCCCGGCCCCCGCGCCCCCGCGCGCGCG
GCTCCTGTTGACCCCGGTC CGCCCGTCCGTCTGC
AGCGCGGCTGAGGTAAGGCGCGGGGCTGGCGG
CGGTTGGCGCGCGGTCCCGGGGTTGGGGAGGG
GGCCGCTTC CGCGGGGAGGAGCGGCCGGCCGG
GGTCCGGGCGGGGTCTGAGGGGA

CTCTTAGTTTTGGGTGCATTTGTCTGGTCTTCCAAA
CTAGATTGAAAGCTCTGAAAAAAAAAACTATCTTGT
GTTTCTATCTGTTGAGCTCATAGTAGGTATCCAGGA
AGTAGTAGGTTGACTGCATTGATTTGGGACTACAC
TGGGAGTTTTCTTCCCATCTCCCTTTAGTTTTCT
TTTTTCTTTCTTTCTTTCTTTTTTTTTTTTTTTTT
TTGAGATGTCTTGTCTCAGTCCCCCAGGCTGGA
GTGCAGTGGTGCATCTTGGCTCACTGTAGCCTCC
ACCTCCCAGGTTCAAGCAATTCTACTGCCTTAGCCT
CCCGAGTAGCTGGGATTACAAGCACCCCGCCACCAT
TCCTGGCTAATTTTTTTTTTTTGTATTTTAGTTGAGA
CAGGGTTTACCAGTGGTGTGATGCTGGTCTCAGA
CTCCTGGGGCCTAGCGATCCCCCTGCCTCAGCCT
CCCAGAGTGTAGGATTACAGGCATGAGCCACTGT
ACC CGCCTCTCTCCAGTTTCCAGTTGGAATCCAA
GGGAAGTAAGTTTAAAGATAAAGTTACGATTTTGAAT
CTTTGGATTGAGAAGAATTTGTCACCTTTAACACCT
AGAGTTGAACTTCATACCTGGAGAGCCTTAACATT
AAGCCCTAGCCAGCCTCCAGCAAGTGGACATTGGT
CAGGTTTGGCAGGATTCTCCCTGAAGTGGACT
GAGAGCCACACCCTGGCCTGTCACCATACCCATCC
CCTATCCTTAGTGAAGCAAACTCCTTTGTTCCCTT
CTCCTTCTCCTAGTGACAGGAAATATTGTGATCCTA
AAGAATGAAAATAGCTTGTACCTCGTGGCCTCAG
GCCTCTTGACTTCAGGCGTTCTGTTAATCAAGT
GACATCTTCCCGAGGCTCCCTGAATGTGGCAGATG
AAAGAGACTAGTTCAACCCTGACCTGAGGGGAAAG
CCTTTGTGAAGGGTCAGGAG

Left: CpG sites at 1/10 nucleotides, constituting a CpG island. The sample is of a gene-promoter, the highlighted ATG constitutes the start codon.

Right: CpG sites present at every 1/100 nucleotides, constituting a more normal example of the genome, or a region of the genome that is commonly methylated.

A Markov model for CpG islands

Problem: Identifying CpG islands.

INPUT: A short DNA sequence

$$X = (x_1, \dots, x_L), x_i \in \{A, C, G, T\}.$$

QUESTION: Decide whether X is a CpG island.

- Use **two different Markov chains**: one for the CpG islands (the “+” model), one for non CpG islands (the “-” model).
- Denote by $a_{st}^{+/-}$ the **transition probabilities** inside/outside CpG islands.

$$\bullet \text{ score}(X) = \log \frac{P(X|\text{CpG island})}{P(X|\text{non CpG island})} = \sum_{i=1}^L \log \frac{a_{x_{i-1}x_i}^+}{a_{x_{i-1}x_i}^-}$$

High score \rightsquigarrow it is more likely that X is a CpG island.

Estimating the probabilities

Assume we are given a **training sample** of known sequences with regions **labeled as CpG/non-CpG islands**.

Estimate transition probabilities by counting the number $c_{st}^{+/-}$ of times letter t followed letter s in $+/-$ regions \rightsquigarrow **ML estimate:**

$$a_{st}^{+/-} = \frac{c_{st}^{+/-}}{\sum_{t'} c_{st'}^{+/-}}$$

Result are two tables:

+	A	C	G	T	-	A	C	G	T
A	a_{AA}^+	a_{AC}^+	a_{AG}^+	a_{AT}^+	a_{AA}^-	a_{AC}^-	a_{AG}^-	a_{AT}^-	
C	a_{CA}^+	...			a_{CA}^-	...			
G	a_{GA}^+				a_{GA}^-				
T	a_{TA}^+				a_{TA}^-				

Identifying CpG islands: Experimental results

In a set of human DNA sequences (in total $\approx 60,000$ nucleotides), 48 putative *CpG* islands have been identified

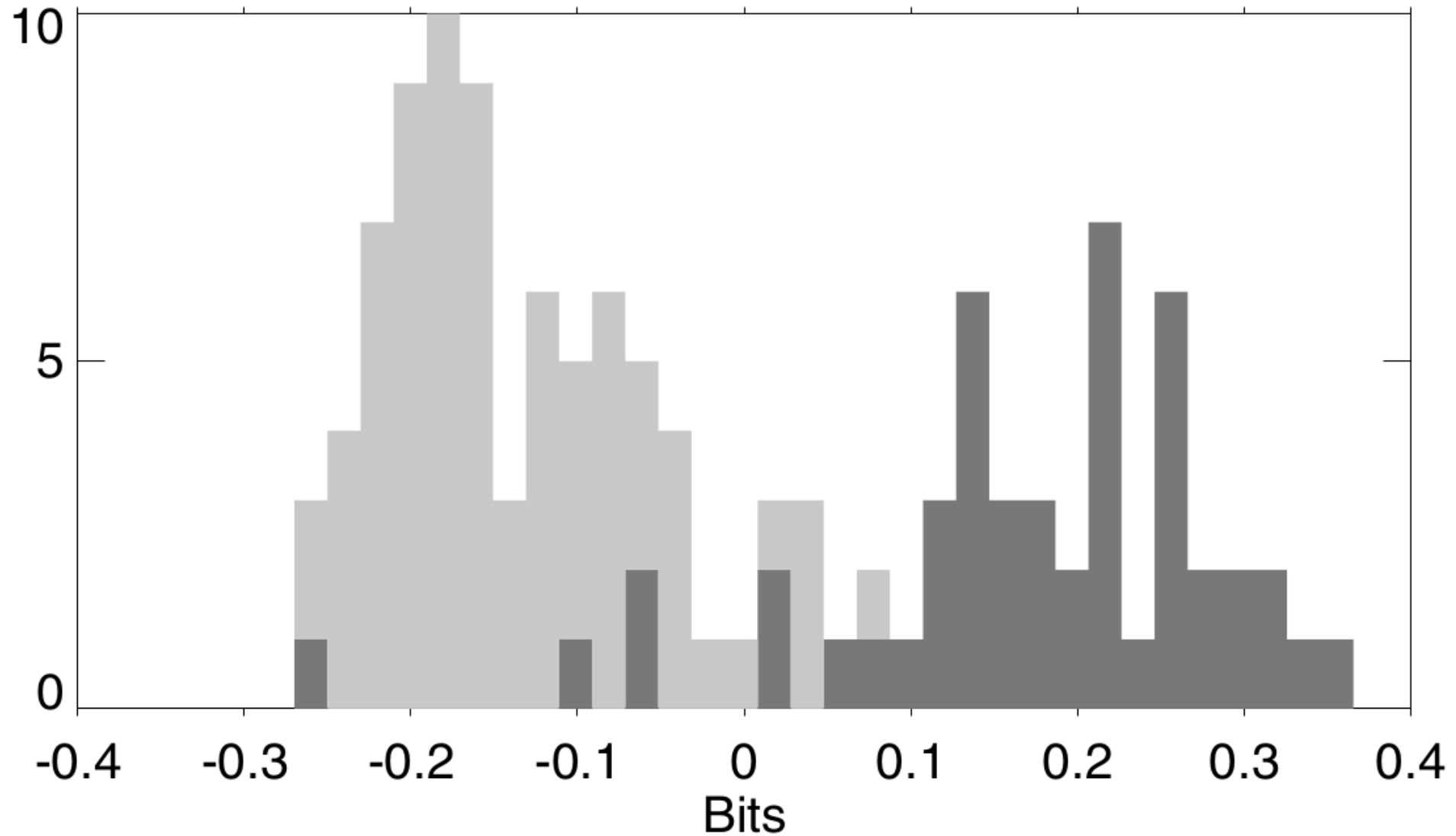
+	A	C	G	T	-	A	C	G	T
A	0.180	0.274	0.426	0.120	A	0.300	0.205	0.285	0.210
C	0.171	0.368	0.274	0.188	C	0.322	0.298	0.078	0.302
G	0.161	0.339	0.375	0.125	G	0.248	0.246	0.298	0.208
T	0.079	0.355	0.384	0.182	T	0.177	0.239	0.292	0.292

↪ Two Markov-chains have been trained

↪ Length-normalized scores reported (in bits) according to

$$\overline{score}(X) = \frac{1}{L} \log \frac{P(X|\text{CpG island})}{P(X|\text{non CpG island})} = \frac{1}{L} \sum_{i=1}^L \log \frac{a_{x_{i-1}x_i}^+}{a_{x_{i-1}x_i}^-}$$

Identifying CpG islands: Experimental results



Histogram of length-normalized scores. CpG islands are shown with dark grey, non-CpG islands shown with light grey.

Durbin et al., Cambridge University Press. <https://doi.org/10.1017/CBO9780511790492.004>

Locating CpG islands

Problem: Locating CpG islands in a DNA sequence.

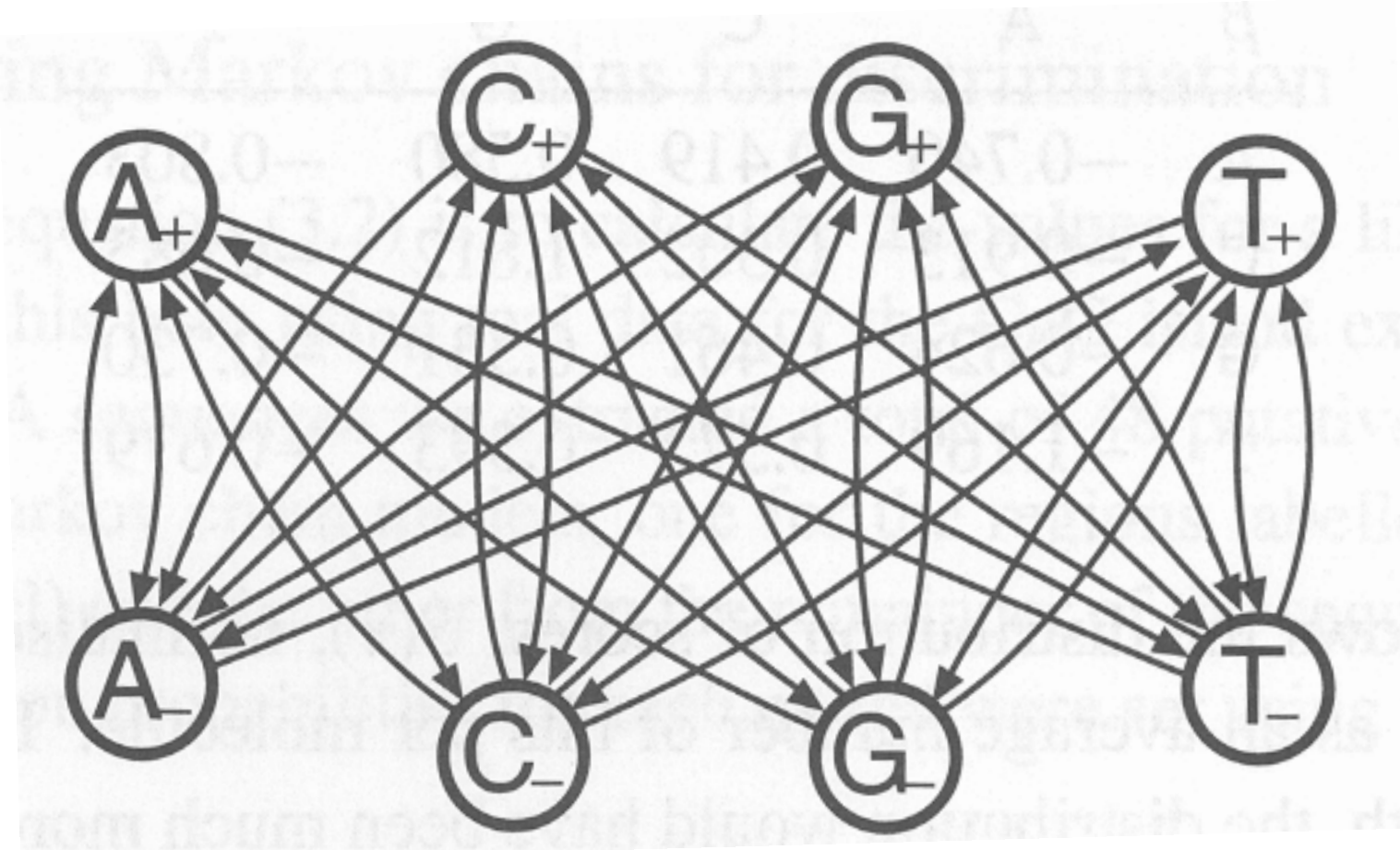
INPUT: A long DNA sequence $X = (x_1, \dots, x_L)$, $x_i \in \{A, C, G, T\}$.

QUESTION: Locate the CpG islands along X .

- **Naive solution:** extract sliding window of length $l \ll L$ from the sequence: $X^k = (x_{k+1}, \dots, x_{k+l})$ where $1 \leq k \leq L - l$.
Calculate $score(X^k)$ for each resulting subsequence.
Subsequences with positive score are potential CpG islands.
- **Problem: No information about the length of the islands,** but algorithm assumes that islands are **at least l nucleotides long.**
 - l too large \rightsquigarrow islands are short substrings, and the scores may not be high enough for a clear distinction.
 - l too small \rightsquigarrow small windows do not provide enough information to distinguish between islands / non-islands.

A unified model for locating CpG islands

Idea: combine the two Markov chains into a unified model, with a small probability of switching from one chain to the other.



(Transitions within each set are not shown) Durbin et al., Cambridge University Press.

Hidden Markov Models

Definition: A Hidden Markov Model (HMM) is a triplet

$\mathcal{M} = (\Sigma, Q, \Theta)$, where

Σ is an alphabet of symbols;

Q is a finite set of states, capable of emitting symbols from Σ ;

Θ is a set of probabilities, comprised of

state transition probabilities a_{kl} for each $k, l \in Q$:

$$a_{kl} = P(\pi_i = l | \pi_{i-1} = k).$$

emission probabilities $e_k(b)$ for each $k \in Q$ and $b \in \Sigma$:

$$e_k(b) = P(x_i = b | \pi_i = k).$$

A path $\Pi = (\pi_1, \dots, \pi_n)$ is a sequence of states.

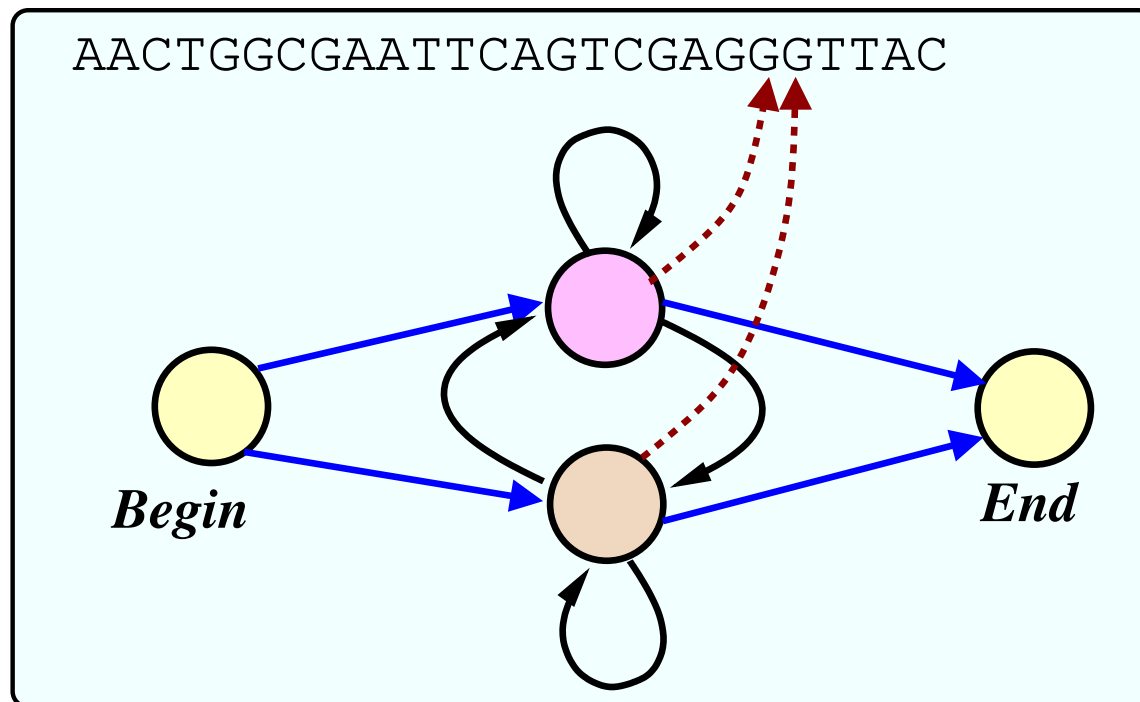
A path follows a simple **Markov chain** \rightsquigarrow probability of moving to a given state depends only on previous state.

Hidden Markov Models

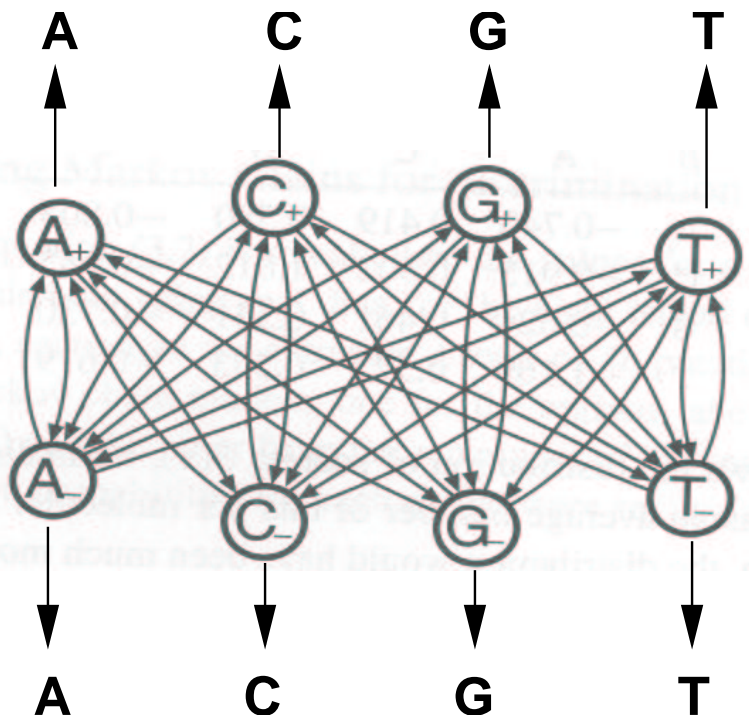
Probability that the sequence X was generated by model \mathcal{M} using the path Π :

$$P(X, \Pi) = a_{\pi_0\pi_1} \cdot \prod_{i=1}^L e_{\pi_i}(x_i) a_{\pi_i\pi_{i+1}},$$

with $\pi_0 \equiv \textit{Begin}$ and $\pi_{n+1} \equiv \textit{End}$.



Example 1: An HMM for detecting CpG islands



State							
A^+	C^+	G^+	T^+	A^-	C^-	G^-	T^-
↓	↓	↓	↓	↓	↓	↓	↓
A	C	G	T	A	C	G	T

Emitted Symbol

Degenerate emission probabilities:

$$P(x_i = A | \pi_i = A^{+/-}) = 1,$$

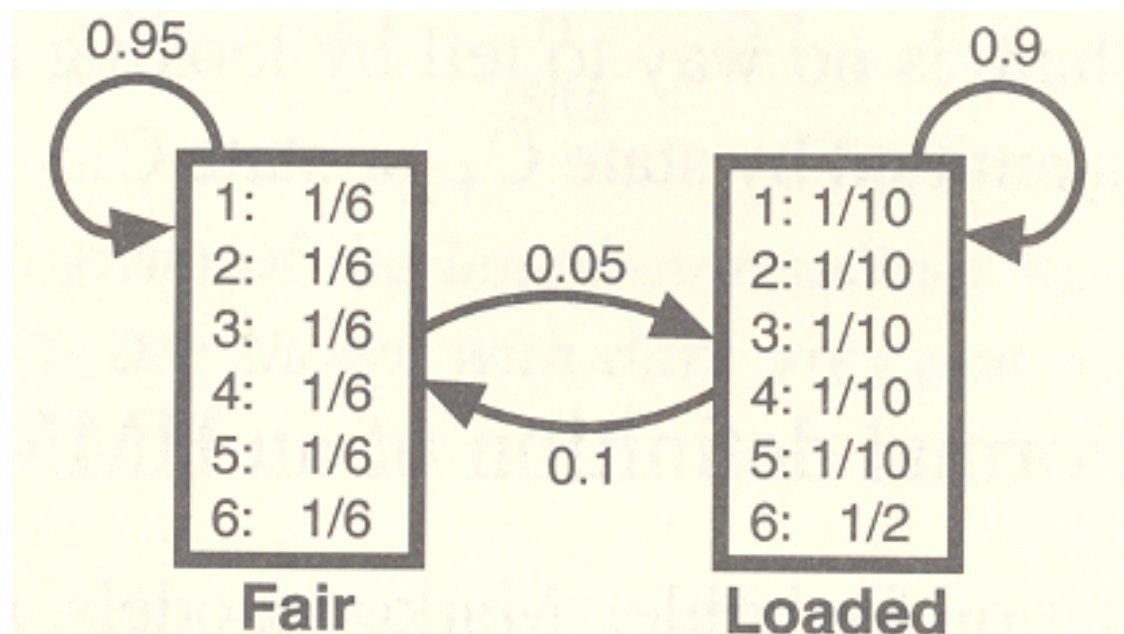
$$P(x_i = A | \pi_i = \{C, G, T\}^{+/-}) = 0$$

analogous for other symbols.

Contrary to Markov chains: No longer a one-to-one correspondence between states and the symbols. The path is **hidden**.

Example 2: Modeling a dishonest casino

- Setting: dealer in casino rolls a die.
- He uses a fair die most of the time, but occasionally he switches to a loaded die: $P(6) = 0.5$, $P(\{1, \dots, 5\}) = 0.1$.
- He switches from state “fair” (\mathcal{F}) to state “loaded” (\mathcal{L}) with probability $a_{\mathcal{F}\mathcal{L}} = 0.05$. He switches back with probability $a_{\mathcal{L}\mathcal{F}} = 0.1$.



The decoding problem

- We observe a sequence of emitted symbols $X = (x_1, \dots, x_L)$.
- But we do not know the sequence of states $\Pi = (\pi_1, \dots, \pi_L)$ that emitted $X \rightsquigarrow$ **the path Π is hidden.**
- CpG islands:
knowing the path would allow us to locate the islands:
all parts that pass through the “+” states are CpG islands.

Problem: The decoding problem.

INPUT: A hidden Markov model $\mathcal{M} = (\Sigma, Q, \Theta)$ and a sequence X over alphabet Σ , for which the generating path $\Pi = (\pi_1, \dots, \pi_L)$ is unknown.

QUESTION: Find the most probable generating path Π^* for X , i.e. the path $\Pi^* = \arg \max_{\Pi} P(X, \Pi)$.

The Viterbi Algorithm

Notation:

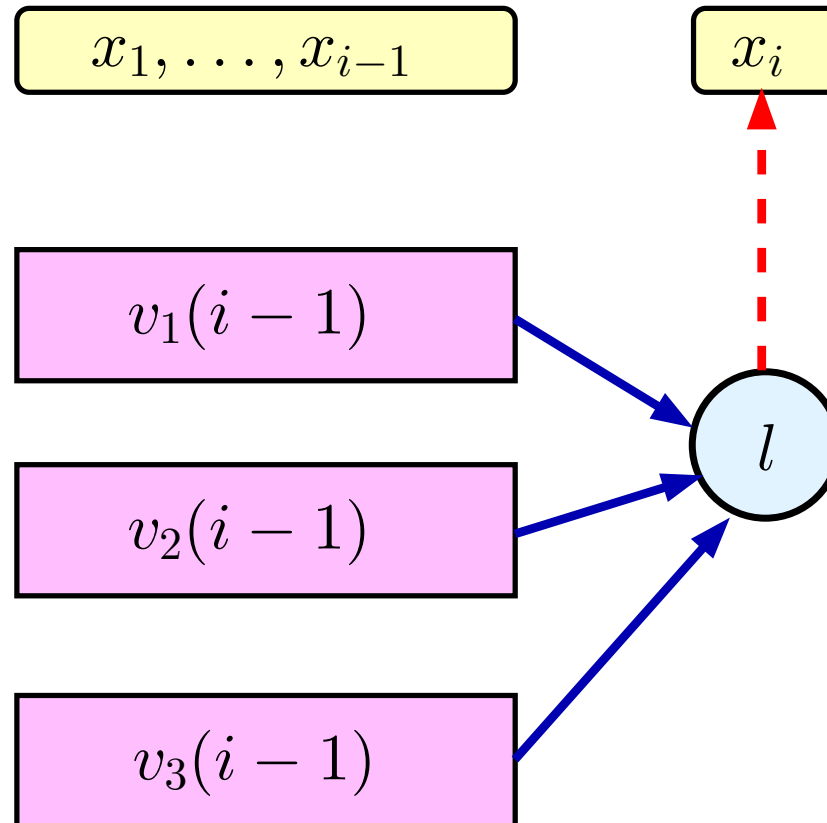
For $k \in Q$ and $0 \leq i \leq L$, denote by $v_k(i)$ the probability of the most probable path for the prefix (x_1, \dots, x_i) that ends in state k :

$$v_k(i) = \max_{\{\Pi | \pi_i = k\}} P(x_1, \dots, x_i, \Pi).$$

Arrange these values in a matrix V with $V_{ki} = v_k(i)$.

The Viterbi Algorithm: Main idea

- Assume that we know (for all k) $v_k(i - 1) =$ probability of most likely path ending in state k with observation x_{i-1} .
- Then $v_l(i) = e_l(x_i) \cdot \max_{k \in Q} (v_k(i - 1) a_{kl})$



The Viterbi Algorithm: Recursion

1. Initialization ($i = 0$):

$$v_{\mathcal{B}}(0) = 1$$

$$\forall_{k \neq \mathcal{B}} : v_k(0) = 0$$

2. Recursion ($i = 1, \dots, L$): for all $l \in Q$ calculate

$$v_l(i) = e_l(x_i) \cdot \max_k (v_k(i-1) a_{kl})$$

$$\text{ptr}_i(l) = \arg \max_k (v_k(i-1) a_{kl})$$

3. Termination:

$$P(X, \Pi^*) = \max_k (v_k(L) a_{k\mathcal{E}})$$

$$\pi_L^* = \arg \max_k (v_k(L) a_{k\mathcal{E}})$$

4. Traceback ($i = L, \dots, 1$): $\pi_{i-1}^* = \text{ptr}_i(\pi_i^*)$

The Viterbi Algorithm (cont'd)

- **Problem:** extensive multiplication of probabilities might result in an underflow.
- **Solution:** logarithmic scores \rightsquigarrow products become sums:

$$v_l(i + 1) = \log(e_l(x_{i+1})) + \max_k \{v_k(i) + \log(a_{kl})\}$$

- **Complexity:**

- **Time:** calculate the values of $O(|Q| \cdot L)$ cells of the matrix V , spending $O(|Q|)$ operations per cell
 \rightsquigarrow overall time complexity is $O(|Q|^2 \cdot L)$.
- **Space:** we have to store matrices of size $(|Q| \times L)$
 \rightsquigarrow space complexity is $O(|Q| \cdot L)$.

A dishonest casino (cont'd)

Experiment: 300 random rolls were generated from the dishonest casino model. Most probable path was computed by way of the Viterbi algorithm.

```
Rolls      315116246446644245311321631164152133625144543631656626566666
Die        FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Viterbi    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls      65116645313265124563666463163666316232645523626666625151631
Die        LLLLLLFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
Viterbi    LLLLLLFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL

Rolls      222555441666566563564324364131513465146353411126414626253356
Die        FFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
Viterbi    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls      366163666466232534413661661163252562462255265252266435353336
Die        LLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi    LLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls      233121625364414432335163243633665562466662632666612355245242
Die        FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Viterbi    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
```

The total probability of a sequence

- For discriminating between CpG/non-CpG islands with Markov chains, for each of both chains we calculated

$$P(X) = \prod_{i=1}^{L+1} a_{x_{i-1}x_i}.$$

For a HMM, calculating $P(X, \Pi)$ is easy, but **what about $P(X)$?**

- Must add the probabilities for **all possible paths** producing X :

$$P(X) = \sum_{\Pi} P(X, \Pi)$$

INPUT: A hidden Markov model $\mathcal{M} = (\Sigma, Q, \Theta)$ and a sequence X over alphabet Σ , for which the generating path $\Pi = (\pi_1, \dots, \pi_L)$ is unknown.

QUESTION: Calculate the total probability $P(X)$.

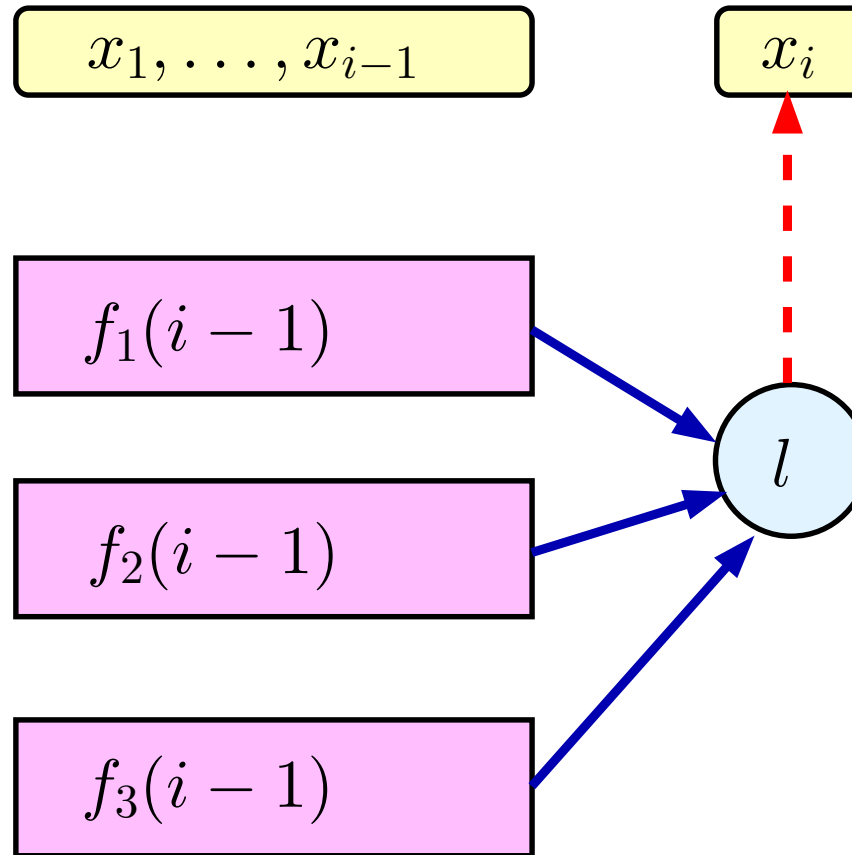
The total probability of a sequence (cont'd)

- **Problem:** number of possible paths **increases exponentially** with the length of X .
- **Possible approximation:** first compute most probable path Π^* by using the Viterbi algorithm, then evaluate

$$P(X, \Pi^*) = a_{\pi_0^* \pi_1^*} \cdot \prod_{i=1}^L e_{\pi_i^*}(x_i) a_{\pi_i^* \pi_{i+1}^*}$$

- **Assumption:** Π^* is only path with significant probability.
- **Approximation unnecessary**, $P(X)$ can be calculated by dynamical programming: replace maximization with summation
 \rightsquigarrow **forward algorithm.**
- Interesting quantity: $P(\Pi^* | X) = \frac{P(X, \Pi^*)}{P(X)}$ is a measure of **“correctness” of the most probable path Π^* .**

The forward algorithm



$f_l(i)$: Prob. of **emitting prefix** (x_1, \dots, x_i) **and reaching state** $\pi_i = l$

$$f_l(i) = e_l(x_i) \cdot \sum_{k \in Q} f_k(i-1) a_{kl}$$

The forward algorithm

$f_k(i)$: Joint probability of emitting the prefix (x_1, \dots, x_i) and reaching the state $\pi_i = k$: $f_k(i) = P(x_1, \dots, x_i, \pi_i = k)$.

1. Initialization ($i = 0$), i.e. empty prefix:

$$f_{\mathcal{B}}(0) = 1$$

$$\forall_{k \neq \mathcal{B}} : f_k(0) = 0$$

2. Recursion ($i = 1, \dots, n$): for all $l \in Q$ calculate

$$f_l(i) = e_l(x_i) \cdot \sum_k f_k(i-1) a_{kl}$$

3. Termination:

$$P(X) = \sum_k f_k(L) a_{k\mathcal{E}}$$

Posterior Decoding

- **Viterbi**: most probable path through the model.
- **Forward**: total probability of sequence X = sum over all paths.
- What about the state at a **particular point** i in a sequence $X = (x_1, \dots, x_i, \dots, x_L)$?
- What is the probability that the observation x_i came from the state k given the observed sequence X ?
- $P(\pi_i = k|X)$: **posterior probability** of state k at time i when the sequence is known.

Posterior Decoding (cont'd)

Problem: Posterior decoding

INPUT: Hidden Markov model $\mathcal{M} = (\Sigma, Q, \Theta)$ and sequence X over alphabet Σ , generating path $\Pi = (\pi_1, \dots, \pi_L)$ is unknown.

QUESTION: For each $1 \leq i \leq L$ and $k \in Q$,
compute the probability $P(\pi_i = k|X)$.

- $P(\pi_i = k|X) = \frac{P(X, \pi_i = k)}{P(X)}$, $X = (\overbrace{x_1, \dots, x_i}^{\text{prefix}}, \overbrace{x_{i+1}, \dots, x_L}^{\text{suffix}})$.

- Memory length is 1 \rightsquigarrow dependency only on the last state

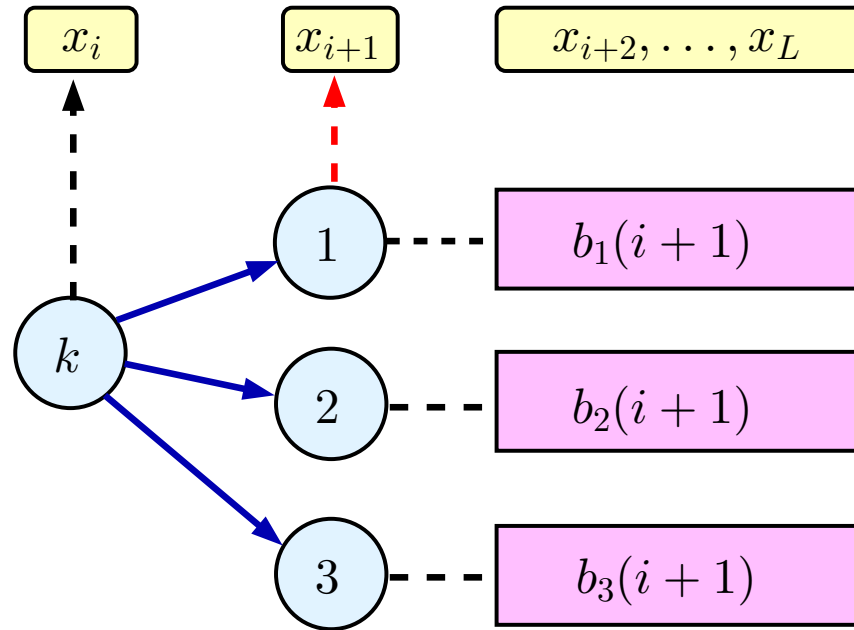
$$P(A, B) = P(A) \cdot P(B|A)$$

$$P(X, \pi_i = k) = P(x_1, \dots, x_i, \pi_i = k) \cdot P(x_{i+1}, \dots, x_L | x_1, \dots, x_i, \pi_i = k)$$

$$= P(x_1, \dots, x_i, \pi_i = k) \cdot P(x_{i+1}, \dots, x_L | \pi_i = k)$$

$$= f_k(i) \cdot b_k(i)$$

The Backward Algorithm



$b_k(i)$: Probability of emitting suffix (x_{i+1}, \dots, x_L) given $\pi_i = k$.

$$\begin{aligned} b_k(i) &= P(x_{i+1}, \dots, x_L | \pi_i = k) \\ &= \sum_{l \in Q} a_{kl} \cdot e_l(x_{i+1}) \cdot b_l(i+1). \end{aligned}$$

The Backward Algorithm

$b_k(i)$: Probability of emitting the suffix (x_{i+1}, \dots, x_L) given $\pi_i = k$:
 $b_k(i) = P(x_{i+1}, \dots, x_L | \pi_i = k)$.

1. Initialization ($i = L$), i.e. empty suffix:

$$\forall k \in Q : b_k(L) = a_{k\epsilon}$$

2. Recursion ($i = L - 1, \dots, 1$):

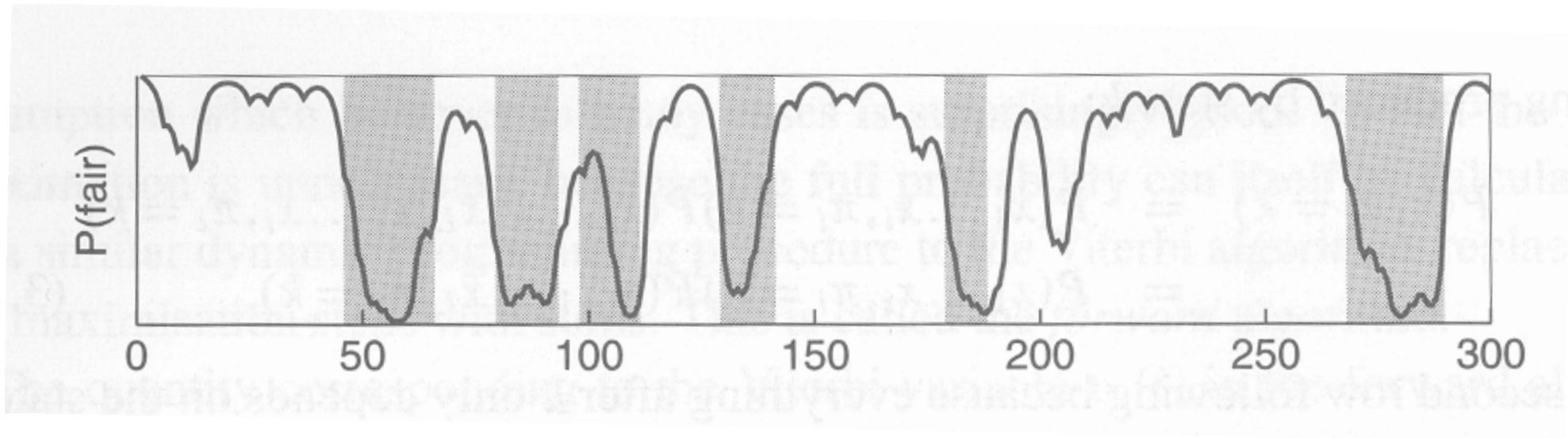
$$b_k(i) = \sum_{l \in Q} a_{kl} \cdot e_l(x_{i+1}) \cdot b_l(i+1)$$

3. Termination:

$$P(X) = \sum_{l \in Q} a_{B_l} \cdot e_l(x_1) \cdot b_l(1)$$

A dishonest casino (cont'd)

Posterior probability of being in the state corresponding to the **fair die**:



Durbin et al., Cambridge University Press. <https://doi.org/10.1017/CBO9780511790492.004>

Shaded areas: loaded die was used.

Uses for Posterior Decoding

- If many paths have almost the same probability as the most probable one, we might want to consider other paths as well.
- E.g. the alternative path Π^{**} , comprised of state sequence $\{k_i\}$ in which **each state k_i has the highest probability** to emit x_i at time i :

$$\Pi^{**} = \arg \max_k \{P(\pi_i = k | X)\}$$

- Π^{**} is a concatenation of **locally most probable states**, but it may not be particularly likely as a path through the whole model.
- Π^{**} may even not be a legitimate path, since some transitions may not be permitted (might have zero probabilities).

Uses for Posterior Decoding (cont'd)

- Often we are interested in some **derived property**. Consider function $g(k) \rightsquigarrow$ an interesting property might be its **expectation under the posterior probability**:

$$G(i|X) = \mathbb{E}_{P(\pi_i|X)}[g] = \sum_k P(\pi_i = k|X)g(k)$$

- **Example:** $g(k) = 1$ for a subset S of states, and 0 for the rest $\rightsquigarrow G(i|X) =$ posterior probability that a state in S emitted x_i .
- In the **CpG island model**, we may define

$$g(k) = \begin{cases} 1 & \text{for } k \in \{A^+, C^+, G^+, T^+\} \\ 0 & \text{for } k \in \{A^-, C^-, G^-, T^-\} \end{cases}$$

$\rightsquigarrow G(i|X)$ is the posterior prob. that x_i is in a CpG island.

Parameter Estimation

- Recall: HMM is a triplet $\mathcal{M} = (\Sigma, Q, \Theta)$.
 Θ is a set of probabilities, comprised of
 - **state transition** probabilities a_{lm} for each $l, m \in Q$
 - **emission** probabilities $e_l(b)$ for each $l \in Q$ and $b \in \Sigma$.
- What can be done if Θ is unknown?
- Consider a set of n **independent training sequences**
 $\mathcal{X} = \{X_1, \dots, X_n\}$ of size L and the corresponding set of paths
 $\Psi = \{\Pi_1, \dots, \Pi_n\}$
- For all n sequences the **total likelihood** is

$$P(\mathcal{X}, \Psi | \Theta) = \prod_{i=1}^n P(X_i, \Pi_i | \Theta) = \prod_{i=1}^n \prod_{k=0}^L e_{\pi_k^i}(x_k^i) a_{\pi_k^i \pi_{k+1}^i}$$

Parameter Estimation: Known state paths

- **Idea:** estimate Θ by **maximizing the total likelihood**, i.e. find the model that **best explains the observed data:**

$$\hat{\Theta} = \arg \max_{\Theta} P(\mathcal{X}, \Psi | \Theta)$$

- Assume that paths of training sequences are known: Count
 - $E_l(b)$: total number of times the b -th observation symbol is emitted from the l -th state,
 - A_{lm} : number of transitions from the m -th state to the l -th state.
- ML estimates are the **observed frequencies:**

$$\hat{e}_l(b) = \frac{E_l(b)}{\sum_{b'} E_l(b')}, \quad \hat{a}_{lm} = \frac{A_{lm}}{\sum_{l'} A_{l'm}}$$

Parameter Estimation: General case

General case: the paths Ψ are unknown

\rightsquigarrow we cannot simply count the total numbers $E_l(b)$ and A_{lm} .

Expectation-Maximization (EM) Algorithm.

E-step: Compute **expected counts**:

$$\mathbb{E}_l(b) = \sum_{\Psi} P(\Psi | \mathcal{X}, \Theta) E_l^{\Psi}(b)$$

$$\mathbb{A}_{lm} = \sum_{\Psi} P(\Psi | \mathcal{X}, \Theta) A_{lm}^{\Psi}$$

M-step:

$$\hat{e}_l(b) = \frac{\mathbb{E}_l(b)}{\sum_{b'} \mathbb{E}_l(b')}, \quad \hat{a}_{lm} = \frac{\mathbb{A}_{lm}}{\sum_{l'} \mathbb{A}_{l'm}}$$

Note that $P(\Psi | \mathcal{X}, \Theta)$ depends on current $\hat{e}_l(b)$ and \hat{a}_{lm} \rightsquigarrow iterate.

Parameter Estimation: General case (cont'd)

E-step: All we need is the forward-/backward algorithm:

$\mathbb{E}_l(b)$ = expected number of times that letter b appears in state l

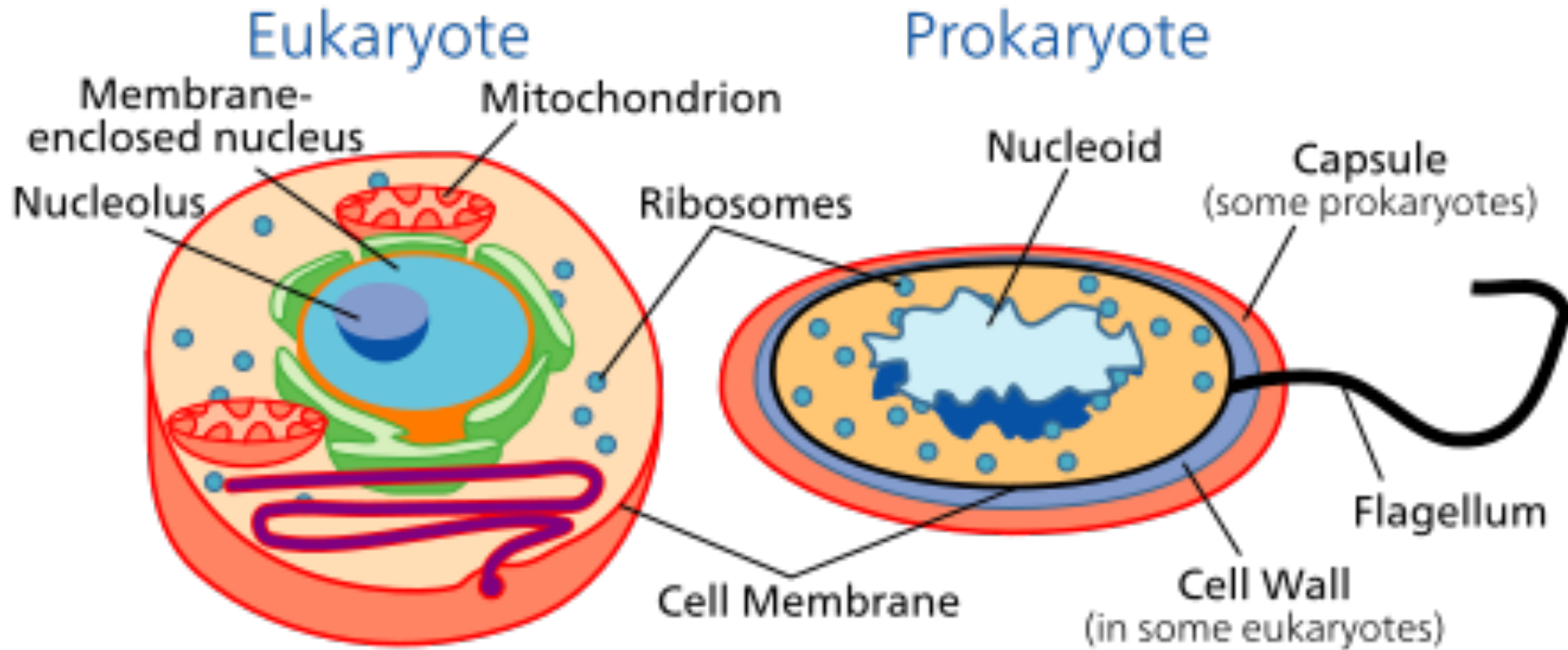
$$= \underbrace{\sum_j \frac{1}{P(X^j)} \sum_{i|x_i^j=b} f_l^j(i) b_l^j(i)}_{P(X^j), f_l^j \text{ and } b_l^j \text{ from forward/backward alg.}}$$

\mathbb{A}_{lm} = expected number of state transitions from m to l

$$= \sum_j \frac{1}{P(X^j)} \sum_i f_l^j(i) \cdot a_{lm} \cdot e_m(x_{i+1}^j) \cdot b_m^j(i+1)$$

Identifying prokaryotic genes

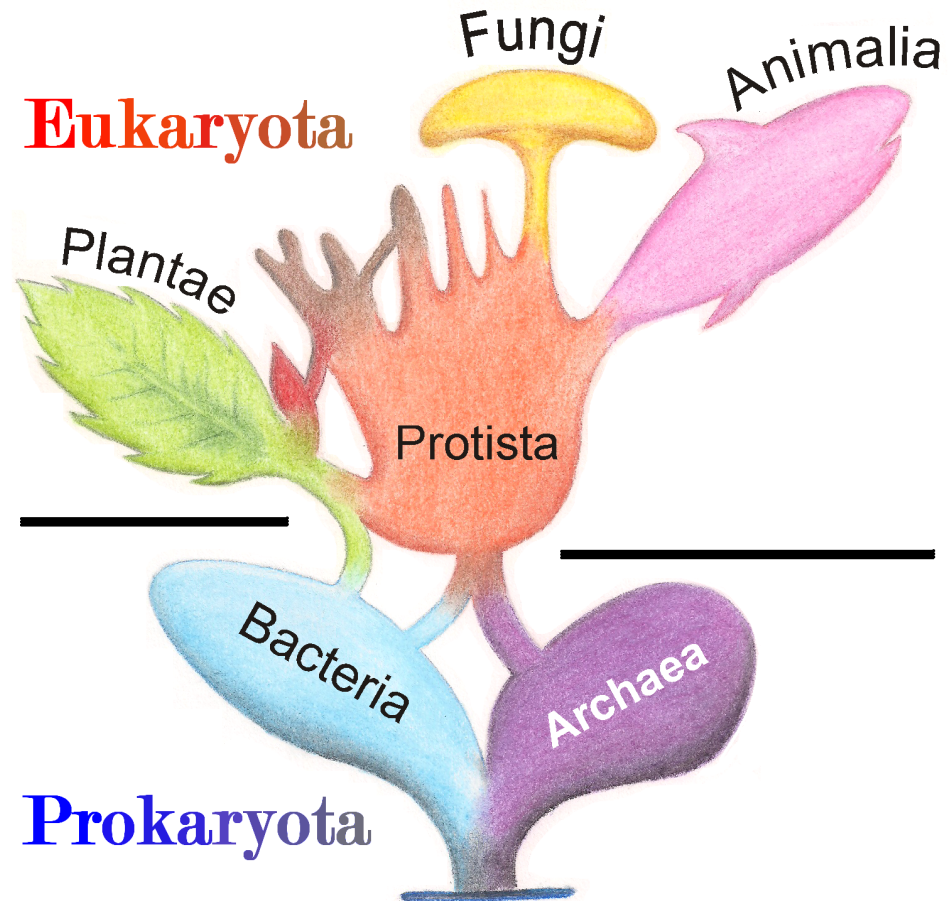
A prokaryote is a cellular organism that lacks an envelope-enclosed nucleus. Organisms with nuclei are called Eukaryota.



By Science Primer, <https://commons.wikimedia.org/w/index.php?curid=2145991>

Identifying prokaryotic genes

Prokaryotes are divided into two domains: **Bacteria and Archaea.**



Identifying prokaryotic genes

Genes of prokaryotes have a simple structure:

Start codon \rightsquigarrow **codons coding for amino acids** \rightsquigarrow **stop codon.**

Codons = nucleotide triplets, 61 for amino acids + 3 stop codons.

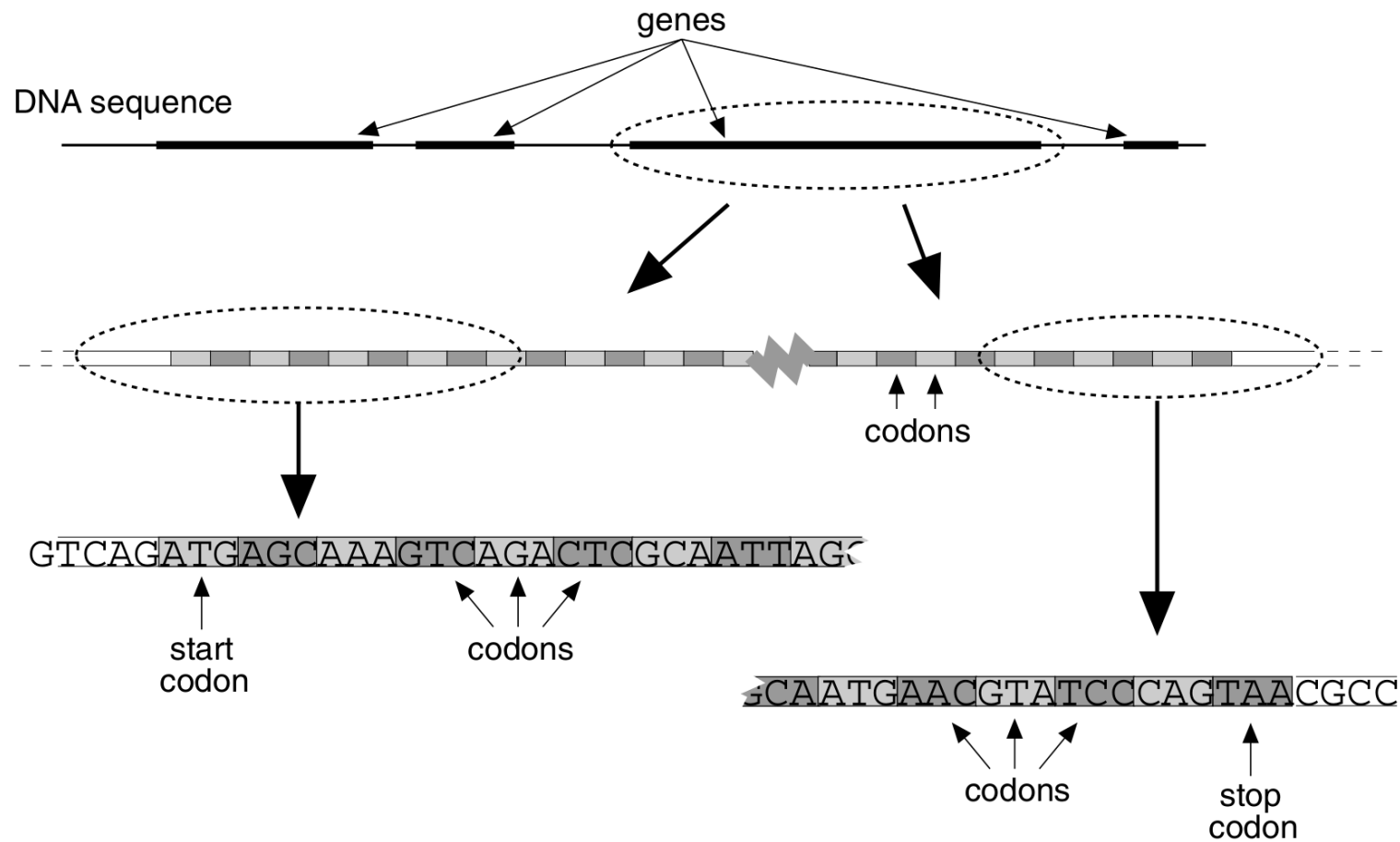
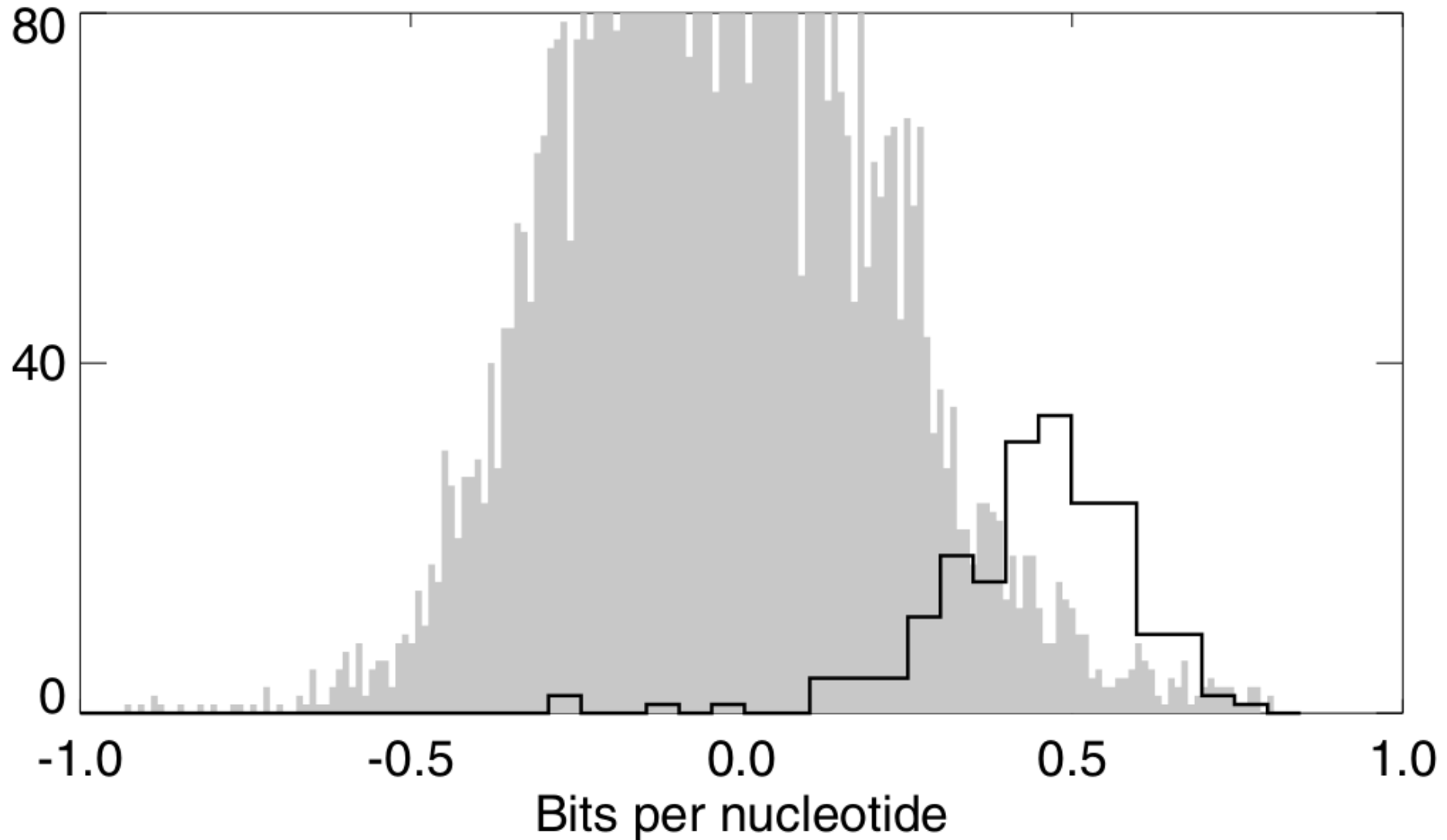


Figure 3.10 *The organisation of genes in prokaryotes.*

Identifying prokaryotic genes

- Finding gene candidates: basically just looking for stretches of DNA with the correct structure.
Such candidate is an **open reading frame (ORF)**.
- But there are **many more ORFs than real genes**.
- Idea: Encode the 64 codons by some characters,
↪ train HMM on these codon sequences.
Same architecture as was used for *CpG* islands:
“+” states for genes, “-” states for NORFs (non-coding ORFs).
- DNA from *E. coli* bacteria (Krogh, Mian & Haussler [1994]):
1100 genes and ≈ 30000 NORFs,
randomly divided into training and testset.

Identifying prokaryotic genes

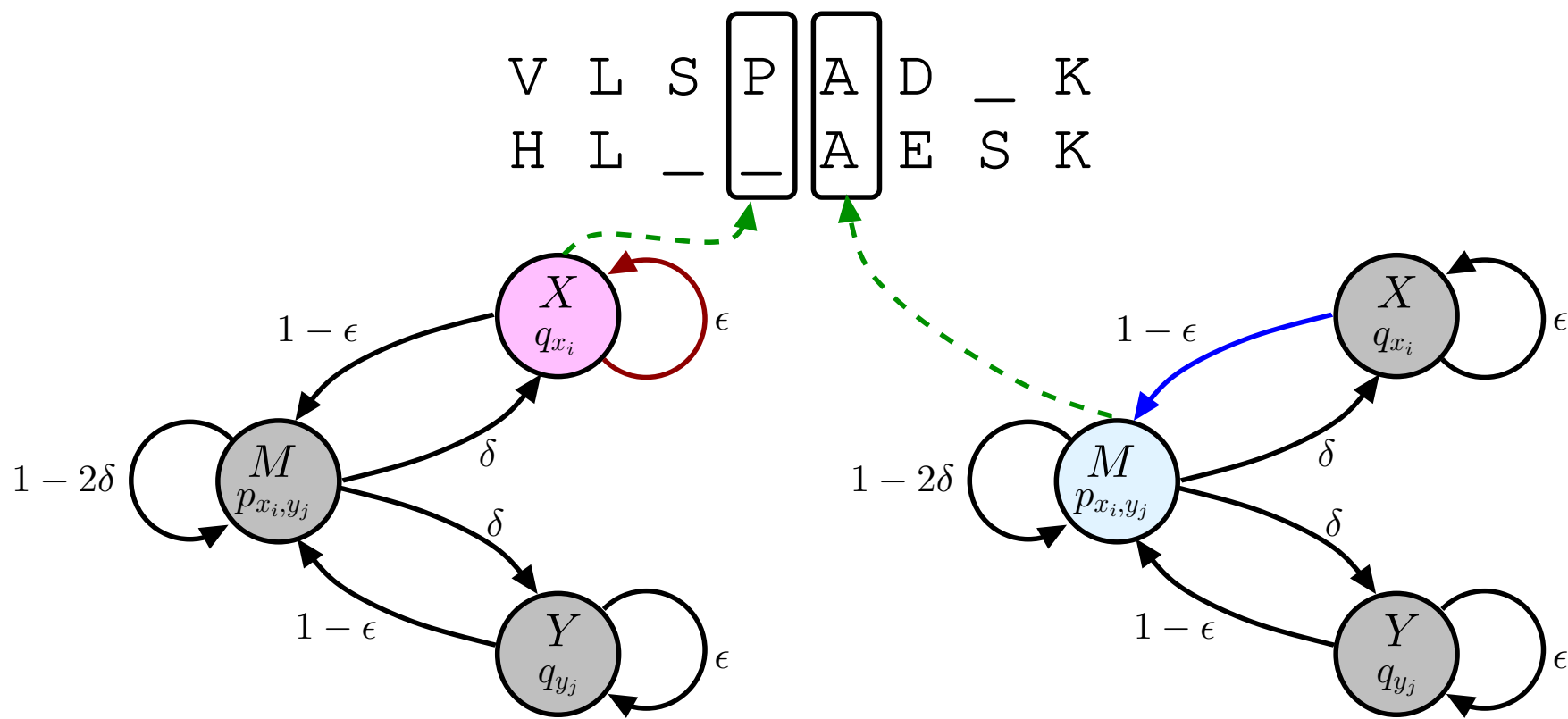


Histograms of the log-odds per nucleotide for all NORFs (grey) and genes (black line).

Chapter 2

Hidden Markov Models

Pair HMMs for Sequence Alignment



Pairwise alignment using HMMs

- The states of an HMM fulfill the **Markov property**: Probability of transition depends only on the last state.
- CpG islands and casino example: HMMs emit **sequence of symbols** (nucleotides or die rolls).
- We only observe the emitted sequences, the **generating state path is unknown** \rightsquigarrow **inference problems**, e.g. estimate the most probable generating path (\rightsquigarrow Viterbi algorithm).
- Knowing the path allows us to **analyze the internal structure** of the string (localizing CpG islands, deciding if the die was fair...)

Pair HMMs for string alignment

HMMs can be used for **sequence alignment**:

emission is not a single string, but a pair of aligned strings

↪ **pair HMMs.**

From a FSA to a pair HMM:

- Define **emission probabilities** for states:

Match state M emits an **aligned pair of symbols** $x_i \diamond y_j$
with probability $p_{x_i y_j}$.

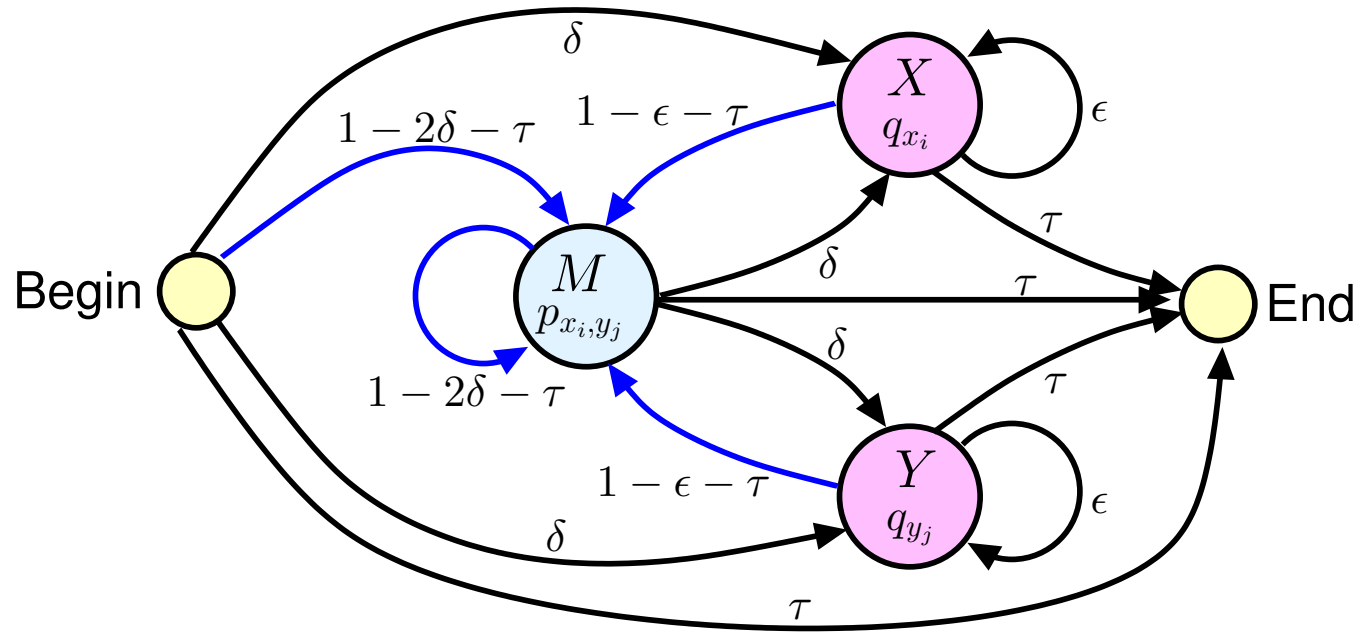
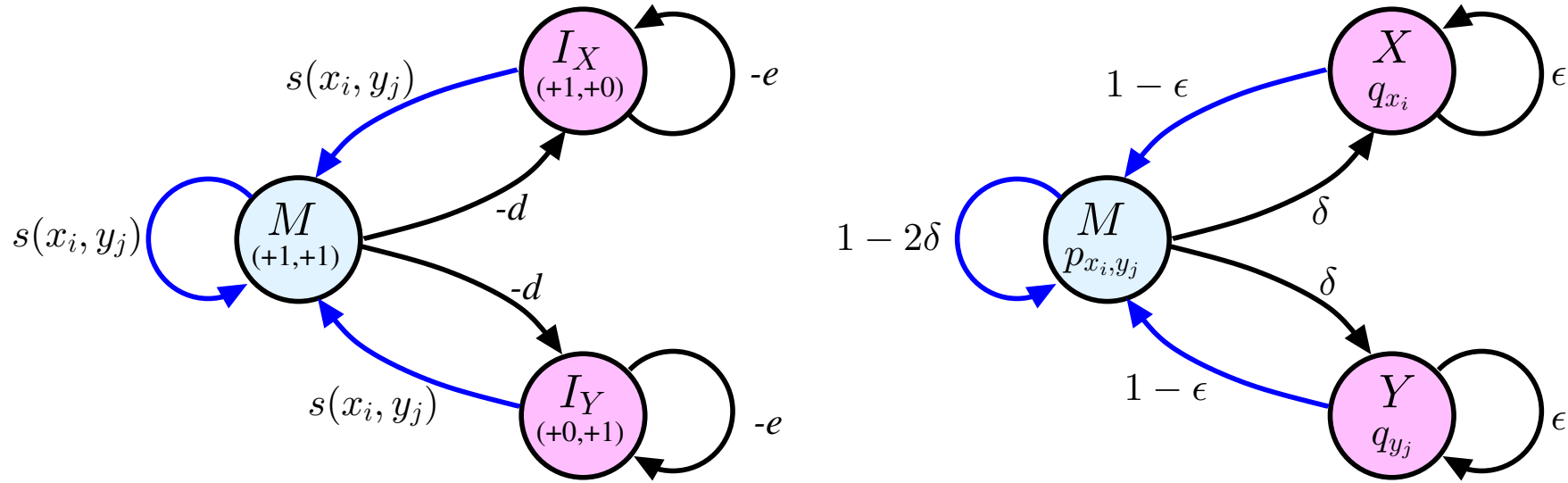
Insert state X emits x_i against a gap with probability q_{x_i} .

- Define **transition probabilities** between the states.

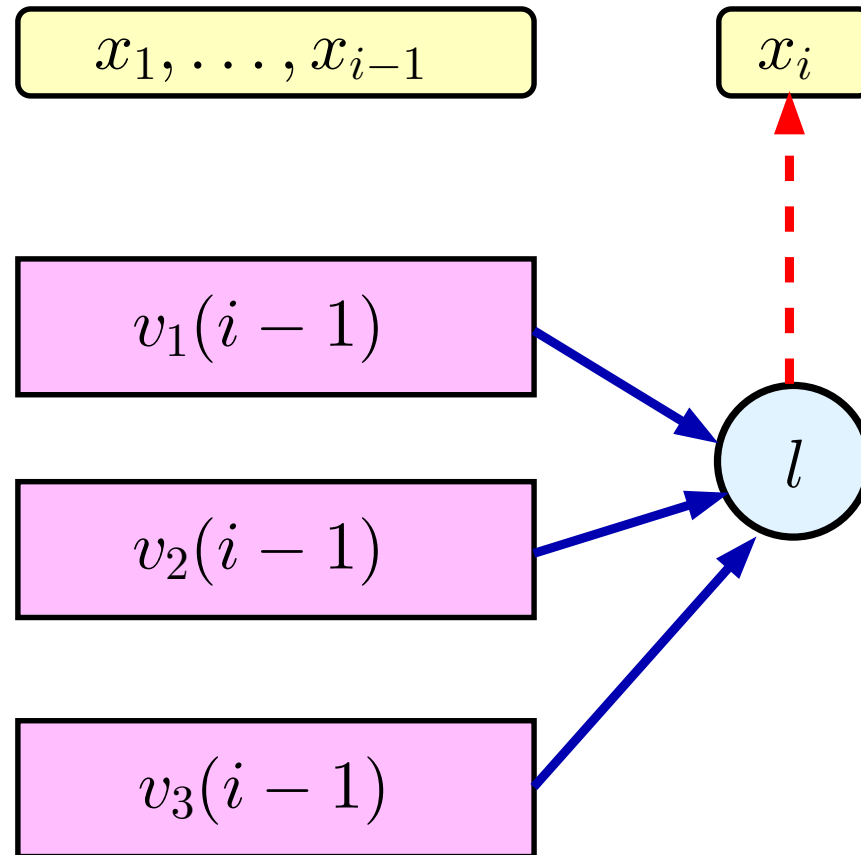
Requirement: probabilities for all the transitions leaving a state must sum to one.

- Add **begin** and **end** states to model the sequence length.

FSAs and Pair HMMs



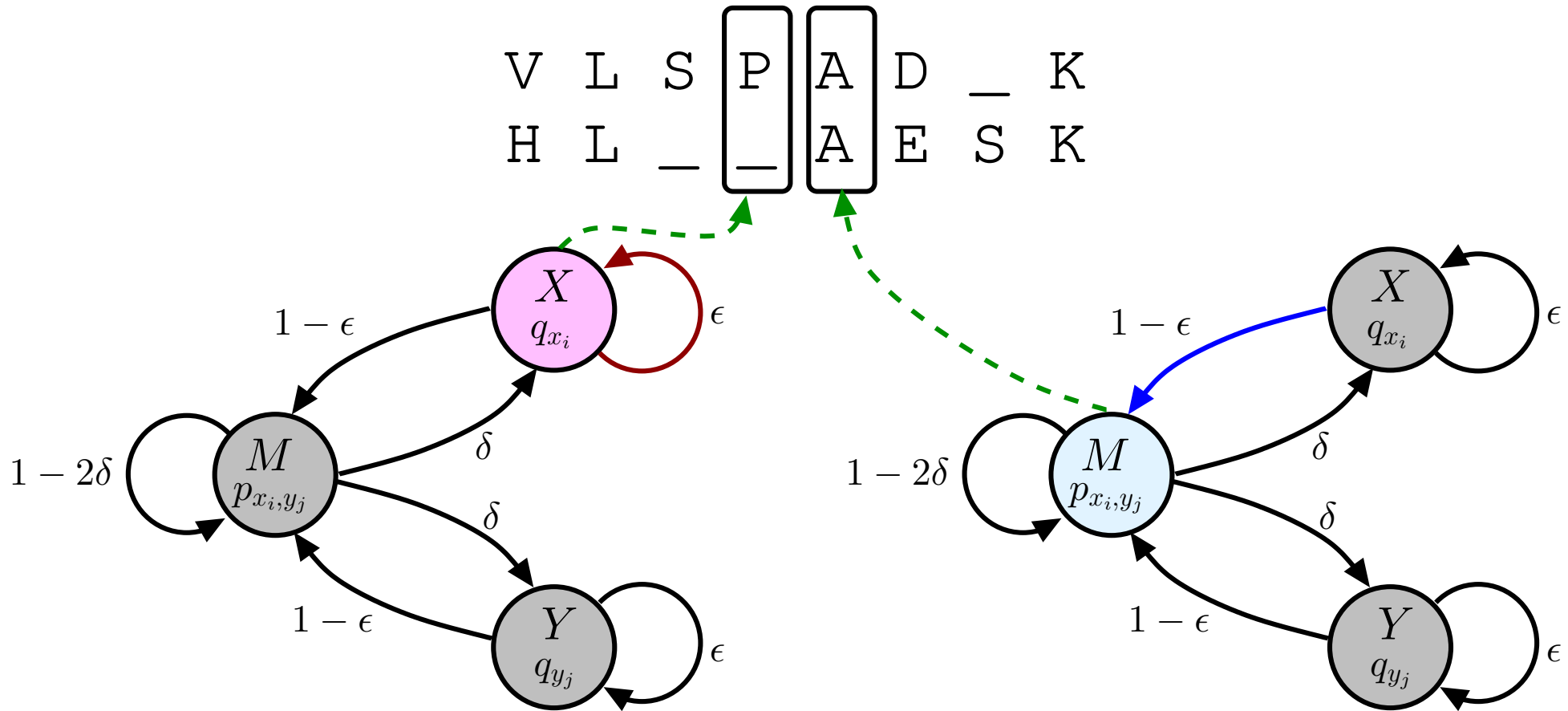
Recall: General structure of Viterbi algorithm



$v_l(i)$: Probability of **most probable path for prefix** (x_1, \dots, x_i) that **ends in state** $\pi_i = l \in Q$.

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

Pair HMMs: Generative models for alignments



$v^{M/X/Y}(i, j)$ = probability of most probable path for
prefix alignment of (x_1, \dots, x_i) and (y_1, \dots, y_j)
that ends in state $M/X/Y$.

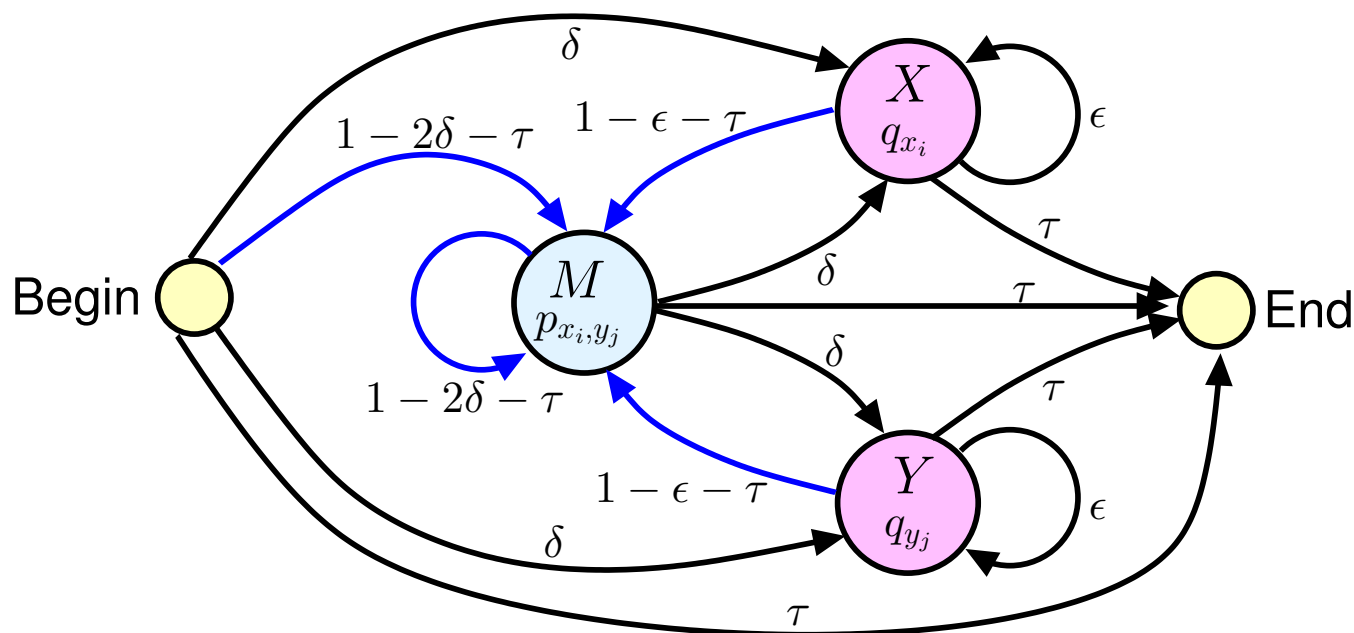
Pair HMMs: Viterbi algorithm (cont'd)

Initialization: $v^M(0, 0) = 1$, $v^M(0, j) = v^M(i, 0) = 0$.

Initialize $v^X(0, j)$ and $v^Y(i, 0)$ (random model).

Recurrence: $i = 1, \dots, n$, $j = 1, \dots, m$:

$$v^M(i, j) = p_{x_i, y_j} \cdot \max \begin{cases} (1 - 2\delta - \tau)v^M(i - 1, j - 1) \\ (1 - \epsilon - \tau)v^X(i - 1, j - 1) \\ (1 - \epsilon - \tau)v^Y(i - 1, j - 1) \end{cases}$$



Pair HMMs: Viterbi algorithm (cont'd)

Initialization: $v^M(0, 0) = 1$, $v^M(0, j) = v^M(i, 0) = 0$.

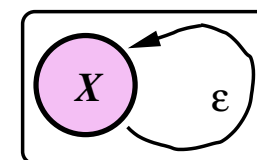
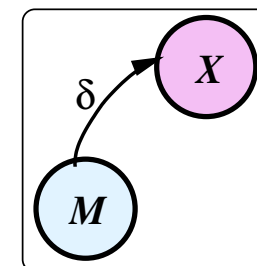
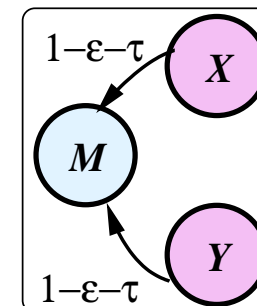
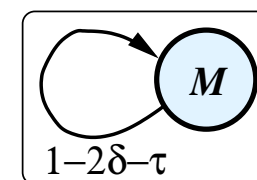
Initialize $v^X(0, j)$ and $v^Y(i, 0)$ (random model).

Recurrence: $i = 1, \dots, n$, $j = 1, \dots, m$:

$$v^M(i, j) = p_{x_i, y_j} \cdot \max \begin{cases} (1 - 2\delta - \tau)v^M(i - 1, j - 1) \\ (1 - \epsilon - \tau)v^X(i - 1, j - 1) \\ (1 - \epsilon - \tau)v^Y(i - 1, j - 1) \end{cases}$$

$$v^X(i, j) = q_{x_i} \cdot \max \begin{cases} \delta v^M(i - 1, j) \\ \epsilon v^X(i - 1, j) \end{cases}$$

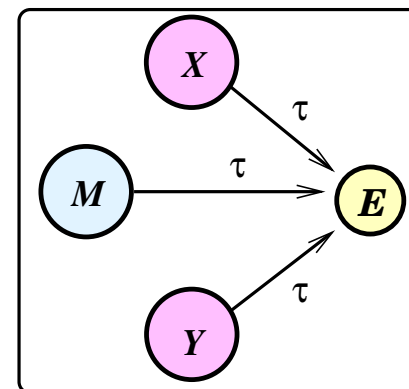
$$v^Y(i, j) = q_{y_j} \cdot \max \begin{cases} \delta v^M(i, j - 1) \\ \epsilon v^Y(i, j - 1) \end{cases}$$



Pair HMMs: Viterbi algorithm (cont'd)

Termination:

$$v^{\mathcal{E}} = \tau \max [v^M(n, m), v^X(n, m), v^Y(n, m)]$$



Traceback:

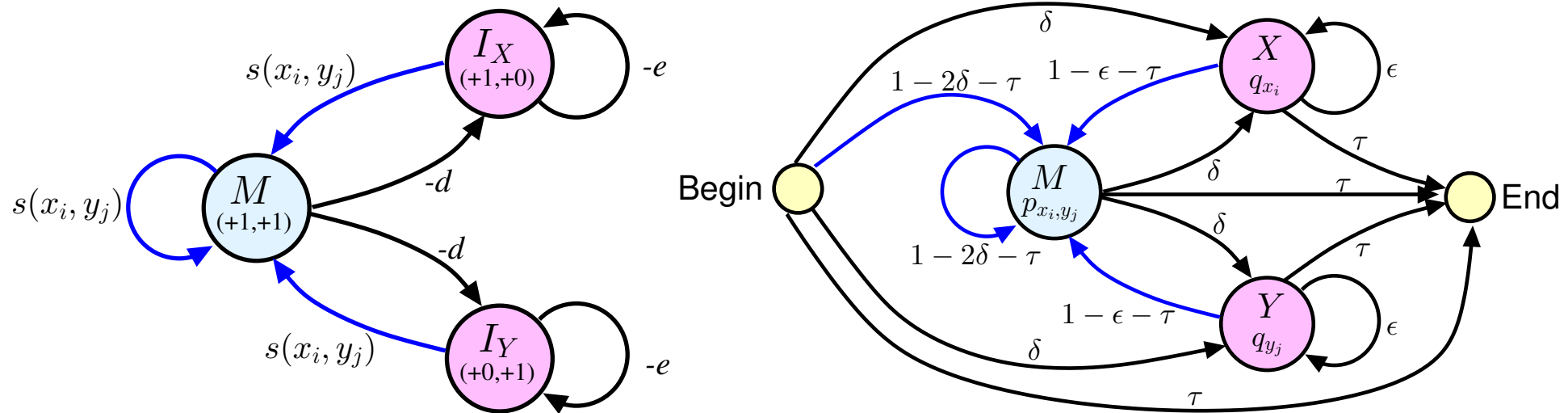
We keep traceback pointers as usual

↪ reconstruct the whole alignment from the pointers.

The Viterbi – FSA Connection

These two models are obviously highly related...

...but what is the precise connection? Is there a specific substitution matrix and affine gap costs such that the FSA alignment is identical with the Viterbi path?



The Viterbi – FSA Connection

Theorem: *The most probable path through the pair HMM for global alignment gives the optimal alignment associated with the substitution matrix*

$$s(x_i, y_j) = \log \frac{p(x_i, y_j)}{q_{x_i} q_{y_j}} + \log \frac{(1 - 2\delta - \tau)}{(1 - \eta)^2}$$

with affine gap penalty $\gamma(g) = -d - (g - 1)e$ with

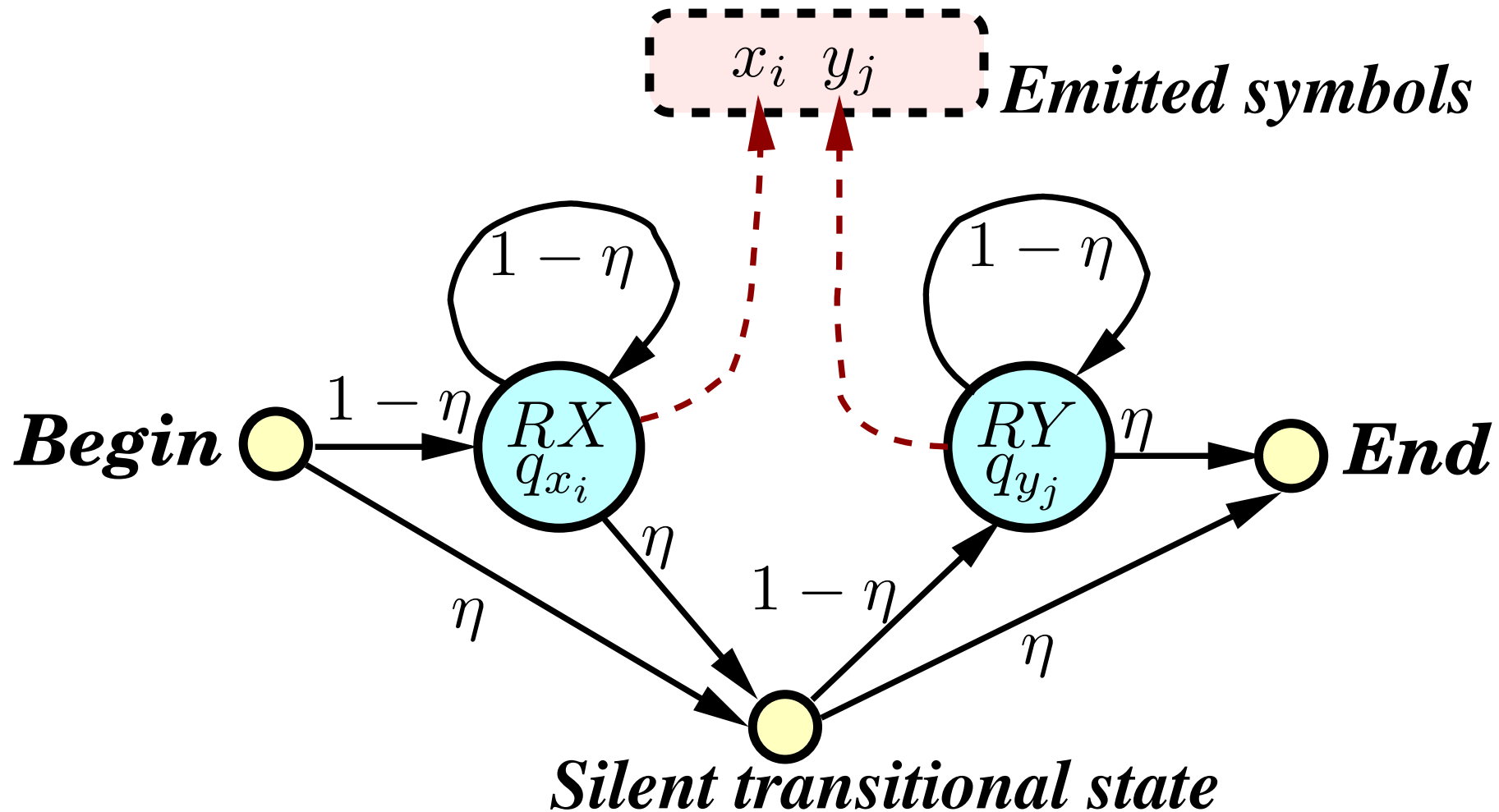
$$d = -\log \frac{\delta(1 - \epsilon - \tau)}{(1 - \eta)(1 - 2\delta - \tau)},$$

$$e = -\log \frac{\epsilon}{1 - \eta}.$$

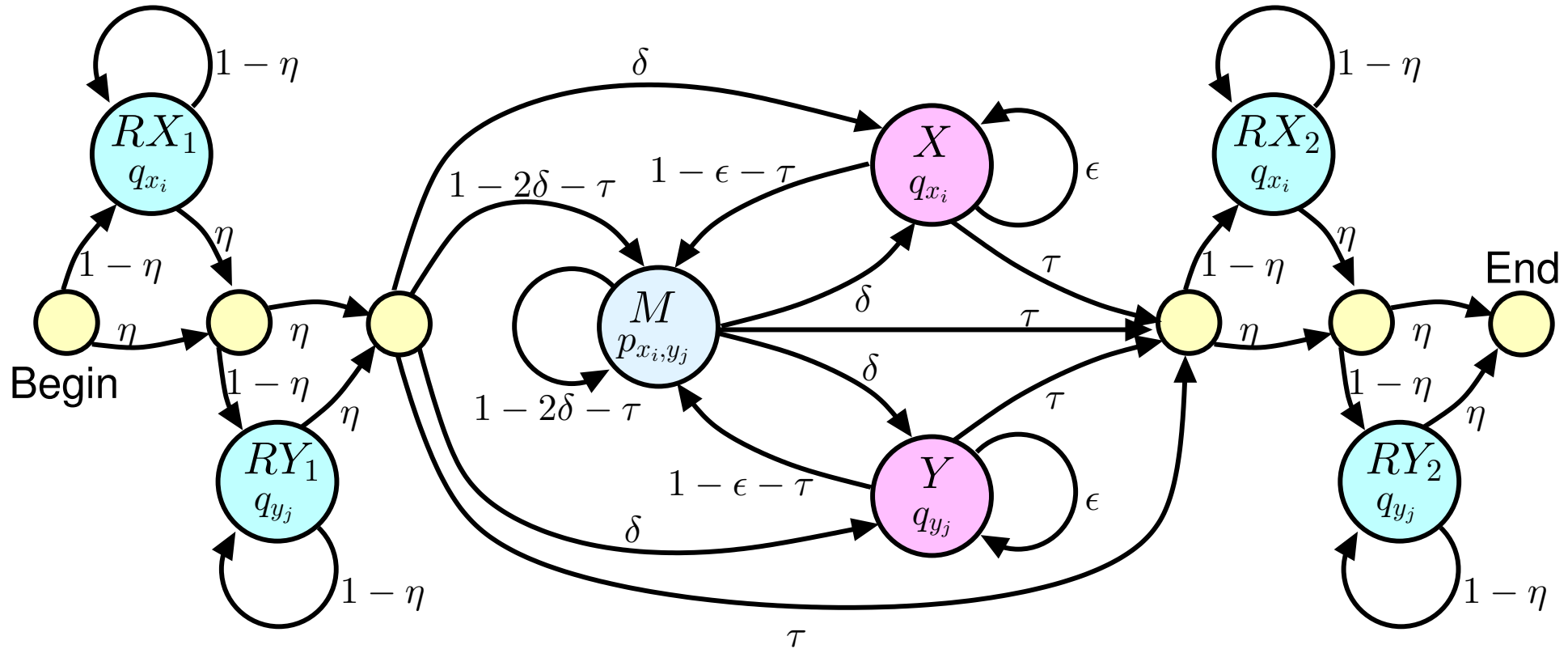
Proof: exercises.

A random model written as a Pair HMM

No match state \rightsquigarrow the states RX and R_Y emit two sequences in turn, **independently of each other**.



A pair HMM for local alignment



Global model flanked by **two copies of the random model**

↪ arbitrary start and stop of alignment.

Sequences in flanking regions are **unaligned** (random model).

The full probability of two aligned sequences

- If the similarity is weak, it is hard to find the correct alignment.
- HMMs allow us to calculate the probability that two sequences are **related by any alignment**:

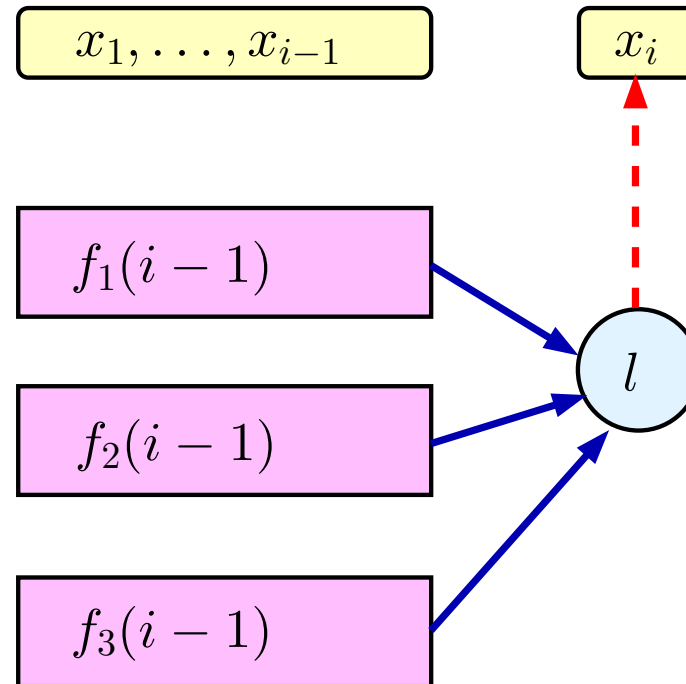
$$P(X, Y) = \sum_{\text{alignments } \Pi} P(X, Y, \Pi)$$

- $P(X, Y)$ always higher than Viterbi-probability $P(X, Y, \Pi^*)$!
Can be **significantly different** when there are many comparable alternative alignments.
- More realistic score: likelihood that two sequences are related by some **unspecified** alignment as opposed to being unrelated:

$$\text{score}(X, Y) = \frac{P(X, Y | \text{Match})}{P(X, Y | \text{Random})} = \frac{\sum_{\Pi} P(X, Y, \Pi)}{q_x q_y}$$

- How to compute $\sum_{\Pi} P(X, Y, \Pi)$? **Forward algorithm.**

Recall: The forward algorithm



$f_l(i)$: Probability of **emitting the prefix** (x_1, \dots, x_i)
and **reaching the state** $\pi_i = l$.

$$f_l(i) = e_l(x_i) \cdot \sum_{k \in Q} f_k(i-1) a_{kl}.$$

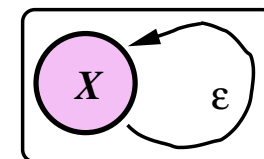
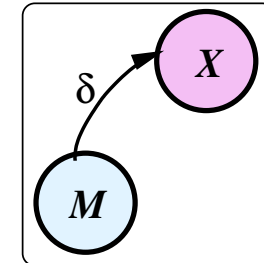
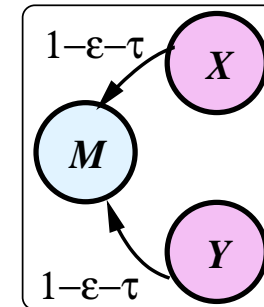
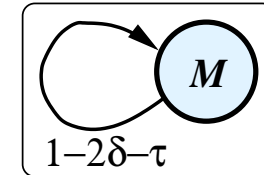
Here: denote by $f^{M/X/Y}(i, j)$ the combined probability of all prefix alignments up to position i and j .

The full probability: forward algorithm

$$f^M(i, j) = p_{x_i, y_j} \left[\begin{aligned} &(1 - 2\delta - \tau) f^M(i - 1, j - 1) \\ &+ (1 - \epsilon - \tau) f^X(i - 1, j - 1) \\ &+ (1 - \epsilon - \tau) f^Y(i - 1, j - 1) \end{aligned} \right]$$

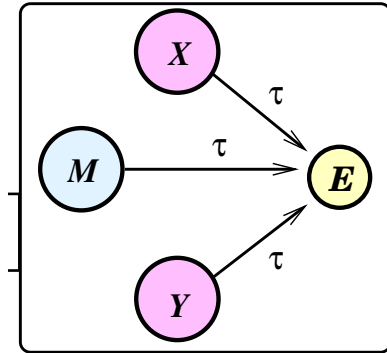
$$f^X(i, j) = q_{x_i} \left[\delta f^M(i - 1, j) + \epsilon f^X(i - 1, j) \right]$$

$$f^Y(i, j) = q_{y_j} \left[\delta f^M(i, j - 1) + \epsilon f^Y(i, j - 1) \right]$$



The full probability (cont'd)

$$P(X, Y) = f^{\mathcal{E}} = \tau [f^M(n, m) + f^X(n, m) + f^Y(n, m)]$$



Important use of $P(X, Y)$: posterior distribution over alignments Π given two sequences X, Y :

$$P(\Pi|X, Y) = \frac{P(X, Y, \Pi)}{P(X, Y)}.$$

Example: set $\Pi = \Pi^*$, the Viterbi path:

$P(\Pi^*|X, Y)$: Posterior probability of observing the Viterbi path
= probability that the optimal scoring alignment is “correct”.

The full probability (cont'd)

Globin example:

```
HBA_HUMAN  GSAQVKGHGKGVADALTNVAHV---D--DMPNALSALSDLHAHKL
              ++  +++++H+  KV    +  +A  ++                +L+  L++++H+  K
LGB2_LUPLU  NNPELQAHAGKVFKLVYEAAIQVVTGTVVTDATLKNLGSVHVS
```

$$P(\Pi^* | X, Y) = 4.6 \cdot 10^{-6}.$$

~> **Alarming observation** if one was hoping that standard alignment algorithms would find the “correct” alignment !

Explanation: there are many small variants of alignments with nearly the same score.

The posterior probability

- Degree of conservation along the sequence may vary depending on **functional** / **structural** constraints \Rightarrow some parts of the alignment will be clear, other regions may be less certain.
- Local view: what about the **local accuracy** of an alignment?
- We are interested in a **reliability measure for each part** of an alignment: probability of two characters (x_i, y_j) being aligned, given the complete sequences:

$$P(x_i \diamond y_j | X, Y)$$

- How to compute? \rightsquigarrow **backward algorithm.**

The backward algorithm

- The quantity we are interested in:

$$P(x_i \diamond y_j | X, Y) = \frac{P(x_i \diamond y_j, X, Y)}{P(X, Y)}.$$

- The denominator: final result from forward algorithm:

$$P(X, Y) = f^{\mathcal{E}}(n, m).$$

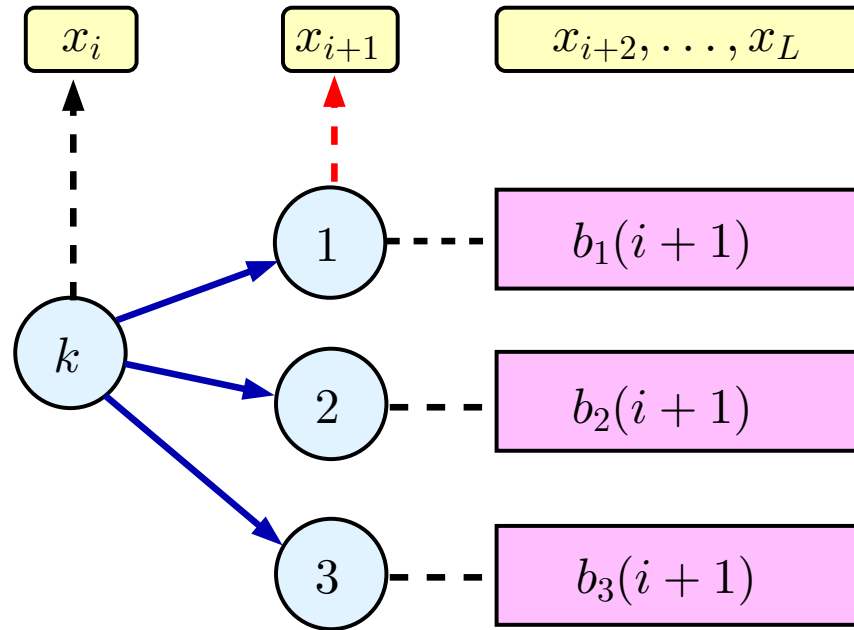
- Numerator: $P(X, Y, x_i \diamond y_j) =$

$$= P(\underbrace{x_1, \dots, i, y_1, \dots, j, x_i \diamond y_j}_A) \cdot P(x_{i+1}, \dots, n, y_{j+1}, \dots, m | \underbrace{x_1, \dots, i, y_1, \dots, j, x_i \diamond y_j}_A)$$

$$\stackrel{\text{Markov}}{=} P(x_1, \dots, i, y_1, \dots, j, x_i \diamond y_j) \cdot P(x_{i+1}, \dots, n, y_{j+1}, \dots, m | x_i \diamond y_j)$$

$$= f^M(i, j) \cdot b^M(i, j).$$

Recall: The backward algorithm



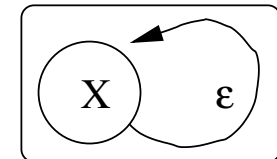
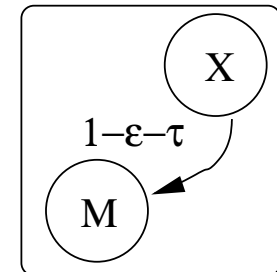
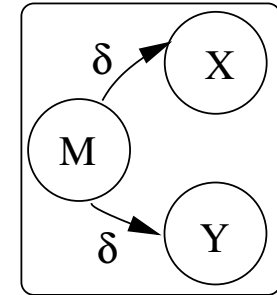
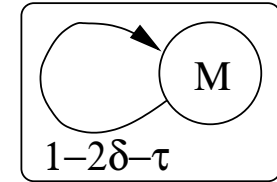
$b_k(i)$: Probability of emitting suffix (x_{i+1}, \dots, x_L) given $\pi_i = k$.

$$b_k(i) = \sum_{l \in Q} a_{kl} \cdot e_l(x_{i+1}) \cdot b_l(i+1).$$

Here: $b^{M/X/Y}(i, j)$: Probability of suffix alignment starting at $i+1$ and $j+1$ given state $\pi_i \in \{M, X, Y\}$.

The backward algorithm: recursion

$$b^M(i, j) = (1 - 2\delta - \tau) p_{x_{i+1}, y_{j+1}} b^M(i + 1, j + 1) + \delta \left[q_{x_{i+1}} b^X(i + 1, j) + q_{y_{j+1}} b^Y(i, j + 1) \right];$$



$$b^X(i, j) = (1 - \epsilon - \tau) p_{x_{i+1}, y_{j+1}} b^M(i + 1, j + 1) + \epsilon q_{x_{i+1}} b^X(i + 1, j);$$

$$b^Y(i, j) = (1 - \epsilon - \tau) p_{x_{i+1}, y_{j+1}} b^M(i + 1, j + 1) + \epsilon q_{y_{j+1}} b^Y(i, j + 1).$$

Chapter 2

Hidden Markov Models

Profile HMMs

Pairwise alignment

x_1	x_2	x_3	-	x_4	x_5
y_1	-	y_2	y_3	y_4	y_5

Profile alignment

x_1	x_2	x_3	-	x_4	x_5
$e_1(b)$	-	$e_2(b)$	$e_3(b)$	$e_4(b)$	$e_5(b)$
V		E	-	-	V
V		K	-	N	Y
V		Y	-	N	Y
F		N	A	N	Y

Profile HMMs

Consider these ungapped blocks first.

Definition: A profile \mathcal{P} of length L is the set of probabilities $e_i(b)$ of observing letter $b \in \Sigma$ at the i -th position.

The probability of a new sequence X according to a given profile \mathcal{P} is

$$P(X|\mathcal{P}) = \prod_{i=1}^L e_i(x_i).$$

x_1	x_2	x_3	x_4	x_5
$e_1(x_1)$	$e_2(x_2)$	$e_3(x_3)$	$e_4(x_4)$	$e_5(x_5)$
V	E	A	N	V
V	K	E	N	Y
V	Y	A	N	Y
F	N	A	N	Y

Position specific score matrices

- **Typical question:** does a new sequence X belong to the family of sequences from which the profile was built?
- Align the sequence against the profile. Test the membership in the family by evaluating the **likelihood score**

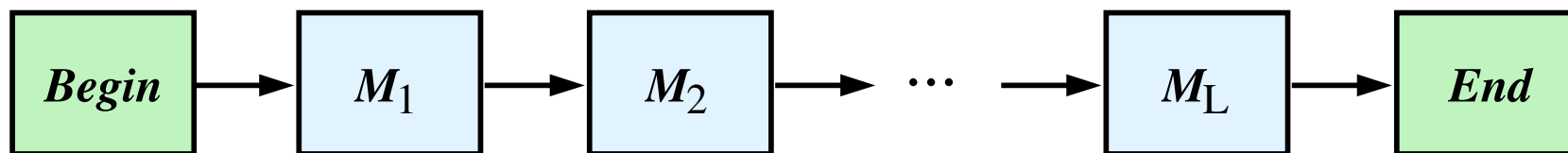
$$score(X|\mathcal{P}) = \sum_{i=1}^L \log \frac{e_i(x_i)}{q_{x_i}}.$$

- The values $\log \frac{e_i(x_i)}{q_{x_i}}$ behave like usual scores $s(a, b)$, where the second index is **position i rather than amino acid b** .
⇒ **Position specific score matrix (PSSM).**

Position specific score matrices (cont'd)

- **A PSSM is a trivial HMM:**

Series of sequentially linked match states M_1, \dots, M_L .



Emission probabilities are $e_{M_j}(b)$, transition probabilities are 1.

- **No choice of transitions**

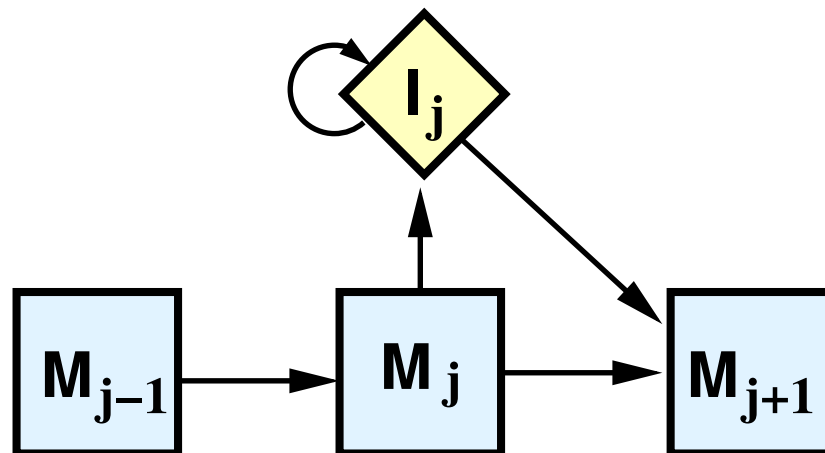
⇒ alignment of a new sequence against the profile is trivial.

- PSSM for an ungapped block captures only some information

⇒ need to find a way to **take account of gaps**.

- **Add insertion states** I_1, \dots, I_l to the model. Assume that emission probabilities of insert states equals random probability: $e_{I_j}(a) = q_a$ (as in pair HMMs for seeing an unaligned symbol).

Position specific score matrices (cont'd)

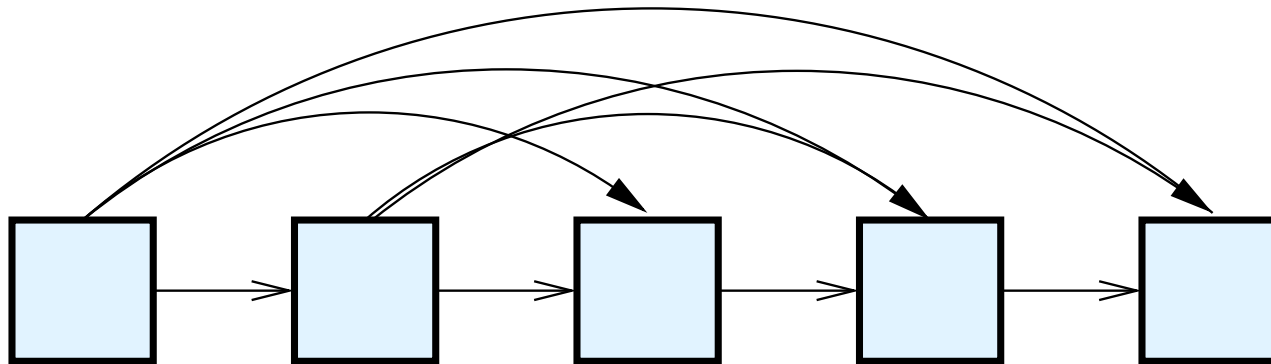


- **Transitions:** $M_j \rightarrow I_j$ (gap open), $I_j \rightarrow I_j$ (gap extension), $I_j \rightarrow M_{j+1}$ (next match).
- log-odd costs of insert: Sum of logarithms of transition probabilities and emission probabilities (assuming gap of length k)

$$\underbrace{\log(a_{M_j I_j}) + \log(a_{I_j M_{j+1}})}_{\text{gap open}} + \underbrace{(k-1) \log(a_{I_j I_j})}_{\text{gap extension}} + \underbrace{\log \frac{e_{I_j}(x)}{q_x}}_{=0}$$

PSSMs: Allowing Deletions

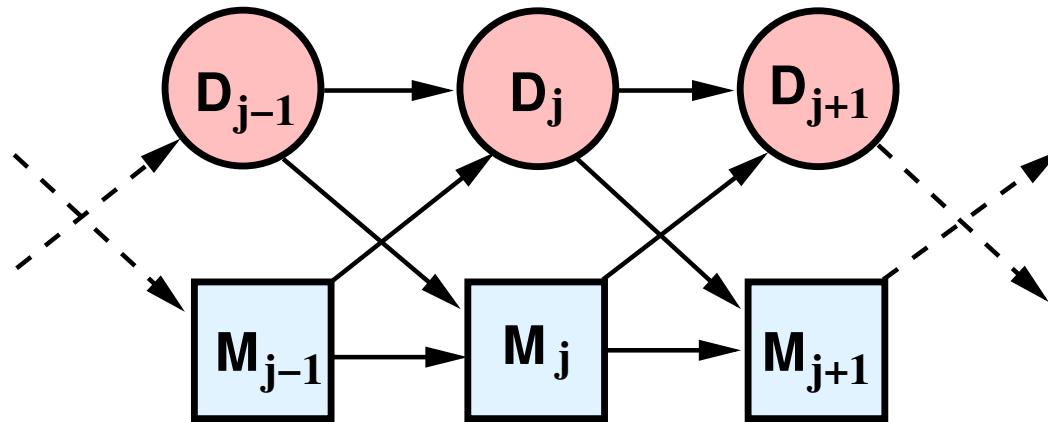
- **Allow deletions** (segments of the multiple alignment that are not matched by any character in x): forward jump transitions.



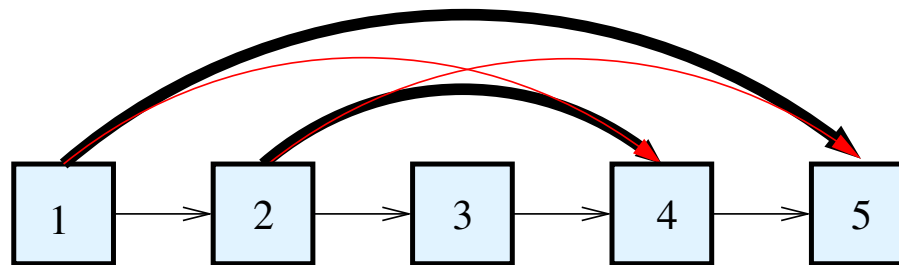
- **Problem:** too many jump transitions required.
- **Solution: deletion states** D_i, \dots, D_L .
They cannot emit any symbol \rightsquigarrow silent states.

Deletions (cont'd)

Model with deletion states:

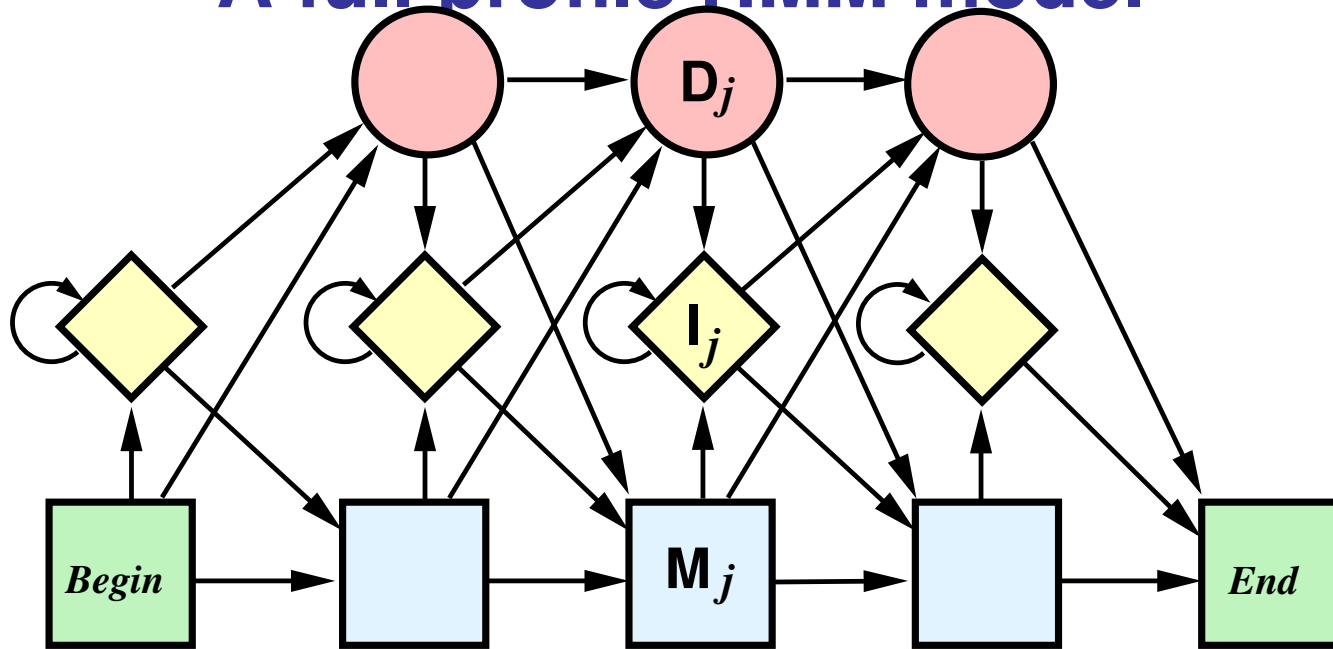


But a price is paid for the reduction in the number of parameters: fully connected model can have **high** transition probabilities from $1 \rightarrow 5$ and $2 \rightarrow 4$, but **low** ones from $1 \rightarrow 4$ and $2 \rightarrow 5$.



Not possible for the model with silent states.

A full profile HMM model



Pairwise alignment

x_1	x_2	x_3	-	x_4	x_5
y_1	-	y_2	y_3	y_4	y_5

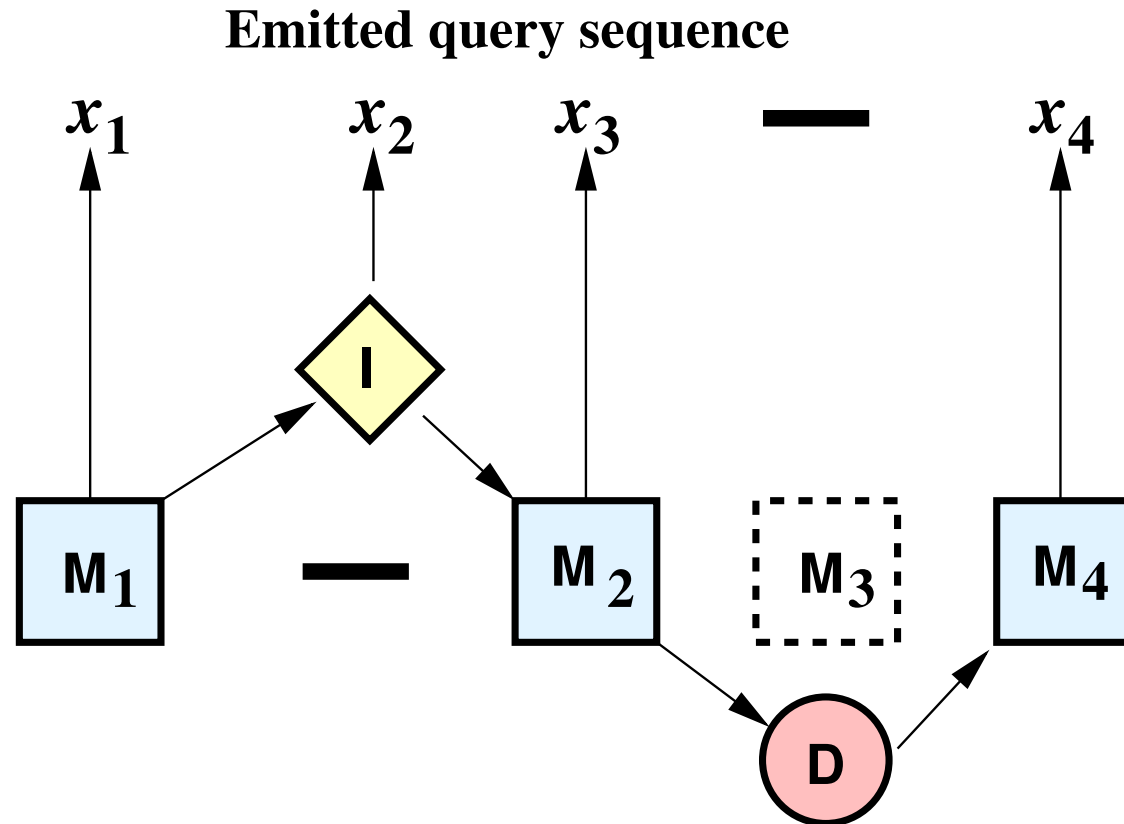
Profile alignment

x_1	x_2	x_3	-	x_4	x_5
$e_1(b)$	-	$e_2(b)$	$e_3(b)$	$e_4(b)$	$e_5(b)$
V		E	-	-	V
V		K	-	N	Y
V		Y	-	N	Y
F		N	A	N	Y

Profile HMM are generalized pair HMMS

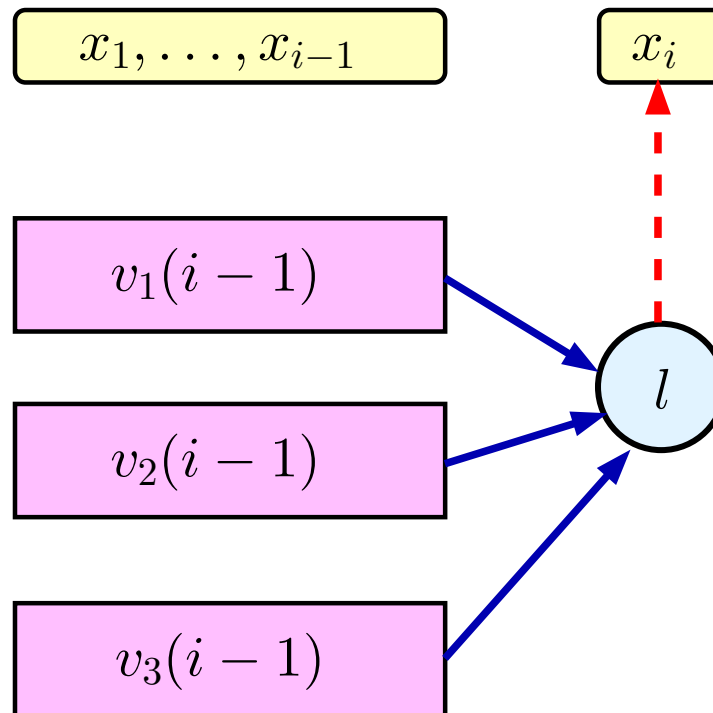
Special case: multiple alignment consist of **one** sequence.

Then, the profile HMM is an **unrolled version of a pair HMM.**



Profile HMM is effectively the model obtained by conditioning the pair HMM on emitting the query sequence X as one of the aligned sequence.

Recall: General structure of Viterbi algorithm



$v_l(i)$: Probability of **most probable path for prefix** (x_1, \dots, x_i)
that **ends in state** $\pi_i = l \in Q$:

$$v_l(i) = e_l(x_i) \cdot \max_{k \in Q} [v_k(i-1) a_{kl}]$$

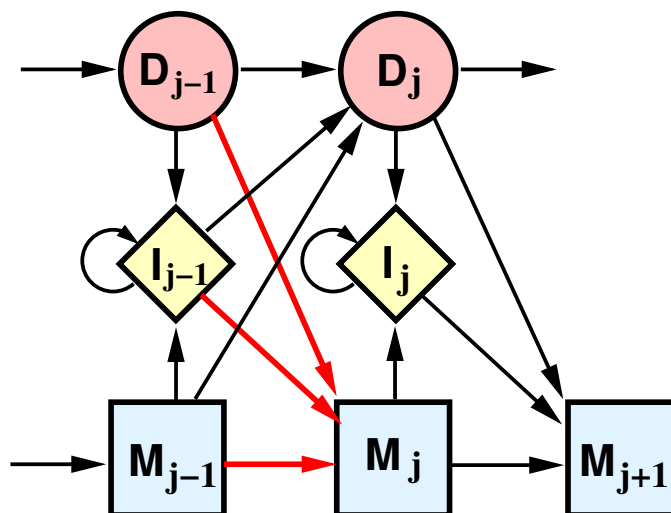
Log-odds variant: $V_l(i) = \log \frac{e_l(x_i)}{q_{x_i}} + \max_{k \in Q} [V_k(i-1) + \log(a_{kl})]$

Log-likelihood score of best path for prefix that ends in state $\pi_i = l \in Q$.

Profile HMM: Viterbi algorithm

Recurrence: Predecessors of match state M_j are the three states of the previous layer $j - 1$:

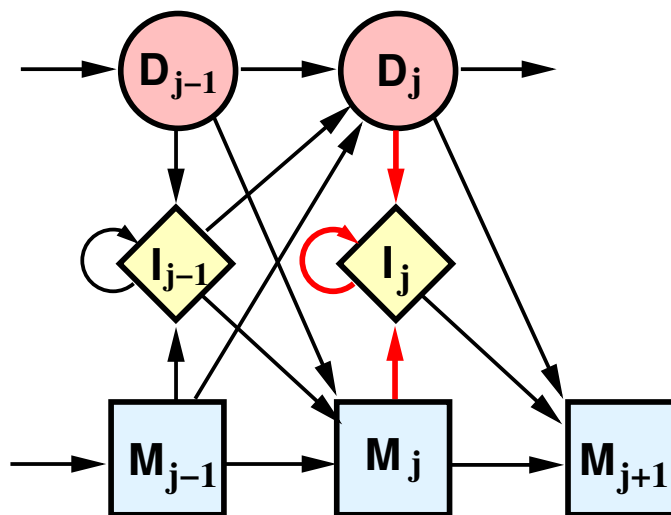
$$V_j^M(i) = \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{j-1}^M(i-1) + \log(a_{M_{j-1}, M_j}) \\ V_{j-1}^I(i-1) + \log(a_{I_{j-1}, M_j}) \\ V_{j-1}^D(i-1) + \log(a_{D_{j-1}, M_j}) \end{cases}$$



Profile HMM: Viterbi algorithm

Predecessors of **insertion state** I_j are the three states of the same layer j :

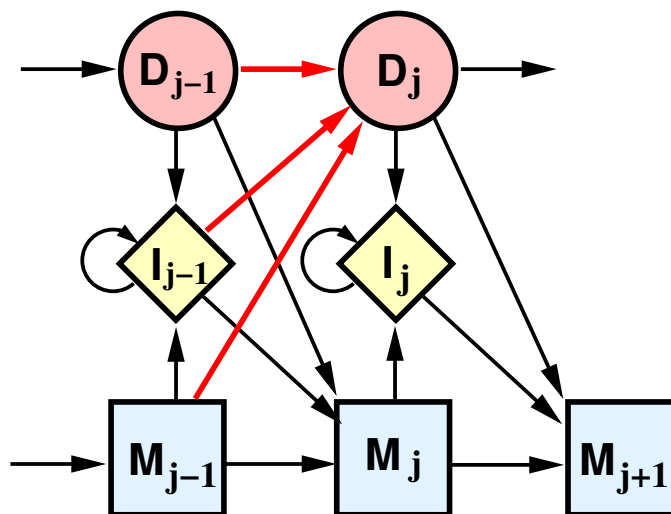
$$V_j^I(i) = \begin{cases} =0, & \text{if } e_{I_j}(x_i) = q_{x_i} \\ \log \frac{e_{I_j}(x_i)}{q_{x_i}} & \end{cases} + \max \begin{cases} V_j^M(i-1) + \log(a_{M_j, I_j}) \\ V_j^I(i-1) + \log(a_{I_j, I_j}) \\ V_j^D(i-1) + \log(a_{D_j, I_j}) \end{cases}$$



Profile HMM: Viterbi algorithm

Predecessors of **deletion state** D_j are the three states of the layer $j - 1$. D_j is a silent state \rightarrow no emission:

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



Profile HMM: Viterbi algorithm

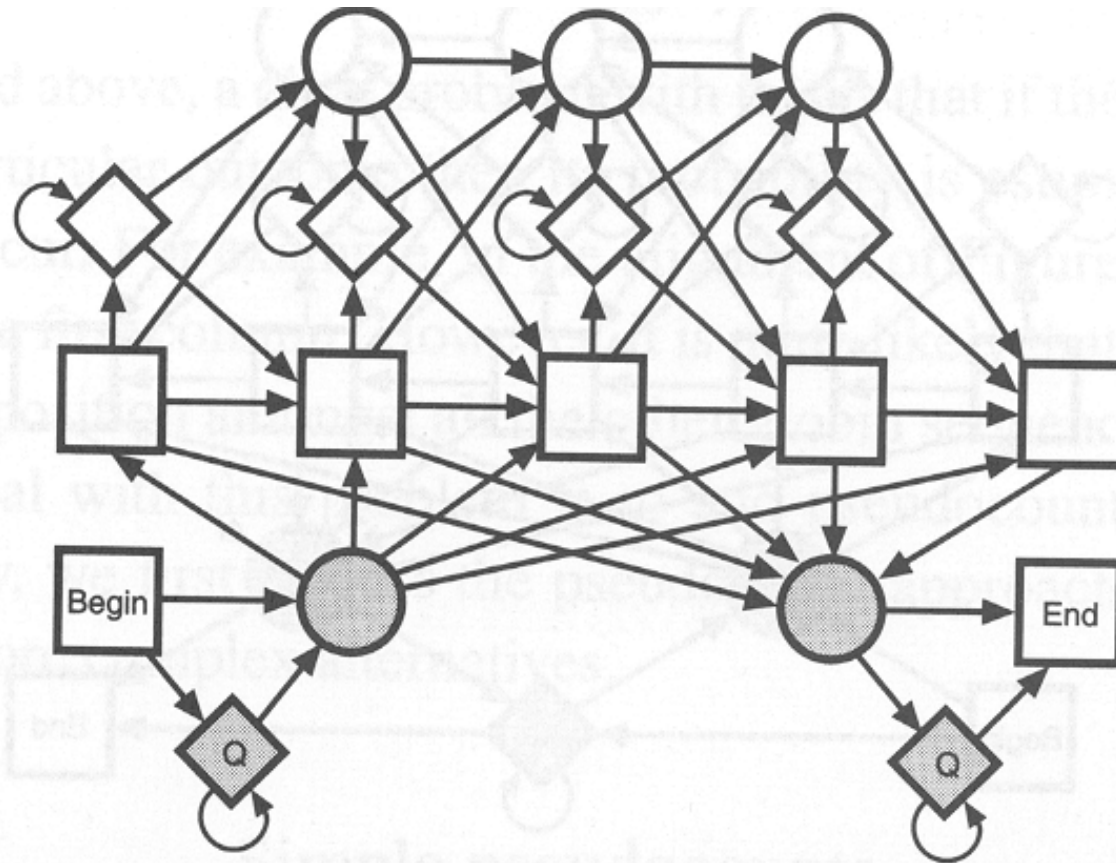
Complexity:

We have to calculate $O(L \cdot m)$ values, while calculating each value takes $O(1)$ operations (since we only need to consider the scores of at most three predecessors).

We therefore need $O(L \cdot m)$ time and $O(L \cdot m)$ space.

Profile HMM: local alignment

Replace original Begin/End states with silent transitional states that are connected to all match-states. Add two independent random models \rightsquigarrow alignment can start and stop everywhere.



Application example

Profile HMM was trained on 300 globin proteins, used for searching other globin sequences in the SWISS-PROT database.

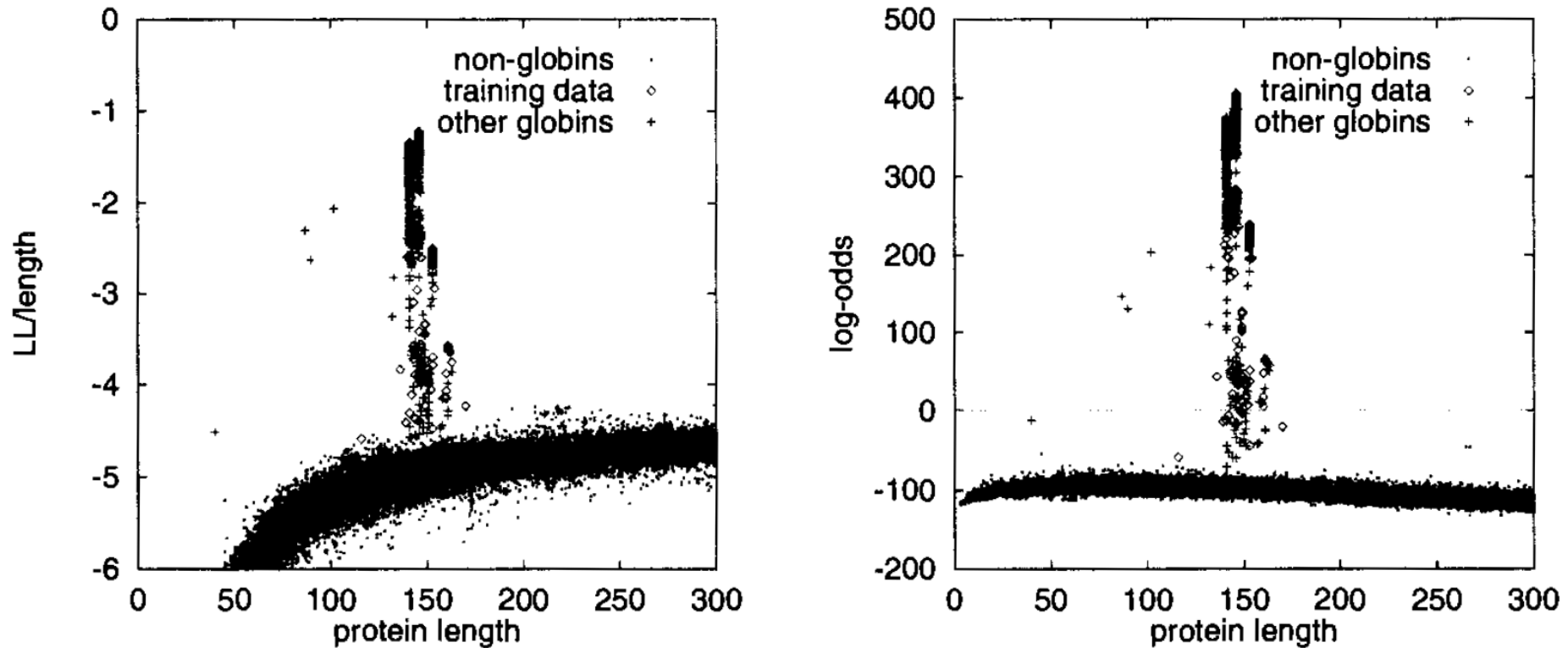


Figure 5.5 *To the left the length-normalized LL score is shown as a function of sequence length. The right plot shows the same for the log-odds score.*

Profile HMM Software



Github: github.com/EddyRivasLab/hmmer. By The logo may be obtained from HMMER., Fair use,
<https://en.wikipedia.org/w/index.php?curid=75738060>

- **HMMER:** free and commonly used software package for sequence analysis written by Sean Eddy.
- General usage: identify homologous protein or nucleotide sequences, perform sequence alignments.
- It detects homology by comparing a profile-HMM to either a single sequence or a database of sequences.
- **Core utility in protein family databases Pfam and InterPro.**
- Other bioinformatics tools such as **UGENE** also use HMMER.
- jackHMMER: iteratively search sequences against a protein database. **Core preprocessing routine in Alphafold.**