

# COM SCI 122 Week 9

Aidan Jan

March 8, 2025

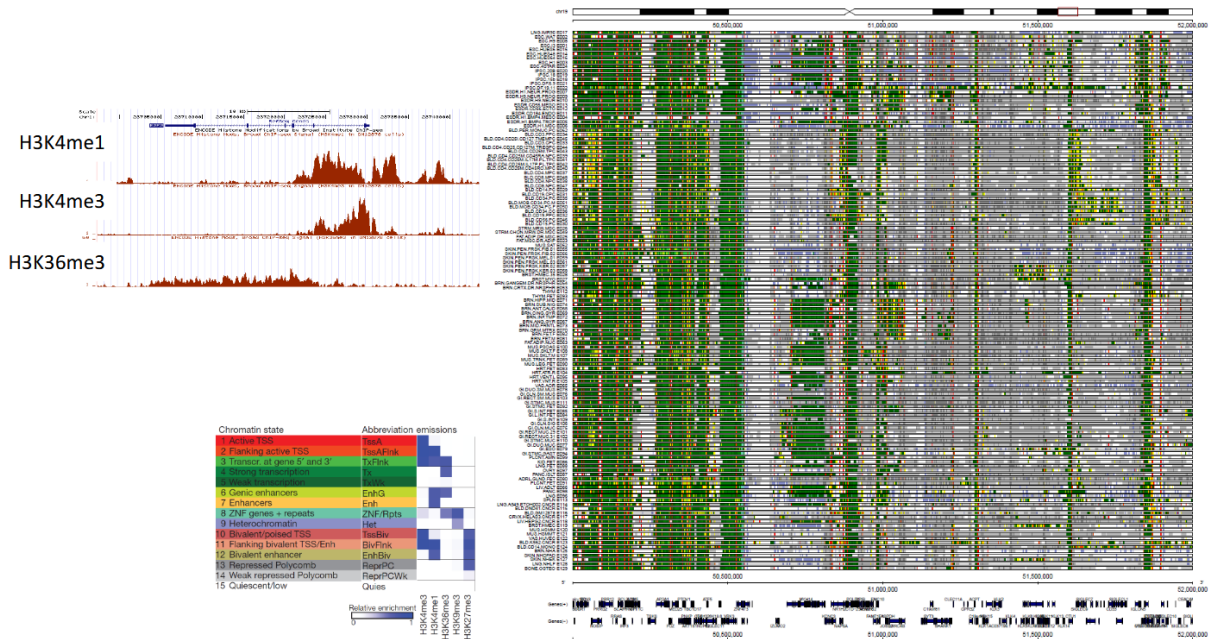
## Applications of Hidden Markov Models (HMMs)

From Wikipedia:

A **hidden Markov model** (HMM) is a Markov model in which the observations are dependent on a latent (or hidden) Markov process (referred to as  $X$ ). An HMM requires that there be an observable process  $Y$  whose outcomes depend on the outcomes of  $X$  in a known way. Since  $X$  cannot be observed directly, the goal is to learn about state of  $X$  by observing  $Y$ . By definition of being a Markov model, an HMM has an additional requirement that the outcome of  $Y$  at time  $t = t_0$  must be "influenced" exclusively by the outcome of  $X$  at  $t = t_0$  and that the outcomes of  $X$  and  $Y$  at  $t < t_0$  must be conditionally independent of  $Y$  at  $t = t_0$  given  $X$  at time  $t = t_0$ . Estimation of the parameters in an HMM can be performed using maximum likelihood estimation. For linear chain HMMs, the Baum-Welch algorithm can be used to estimate parameters.

Hidden Markov models are known for their applications to thermodynamics, statistical mechanics, physics, chemistry, economics, finance, signal processing, information theory, pattern recognition—such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bioinformatics.

### HMMs for Analyzing and Modeling Epigenetic Data



Ernst and Kellis, *Nature Methods* 2012; Roadmap Epigenomics Consortium et al, *Nature* 2015

## HMMs for Genotype Imputation and Phasing

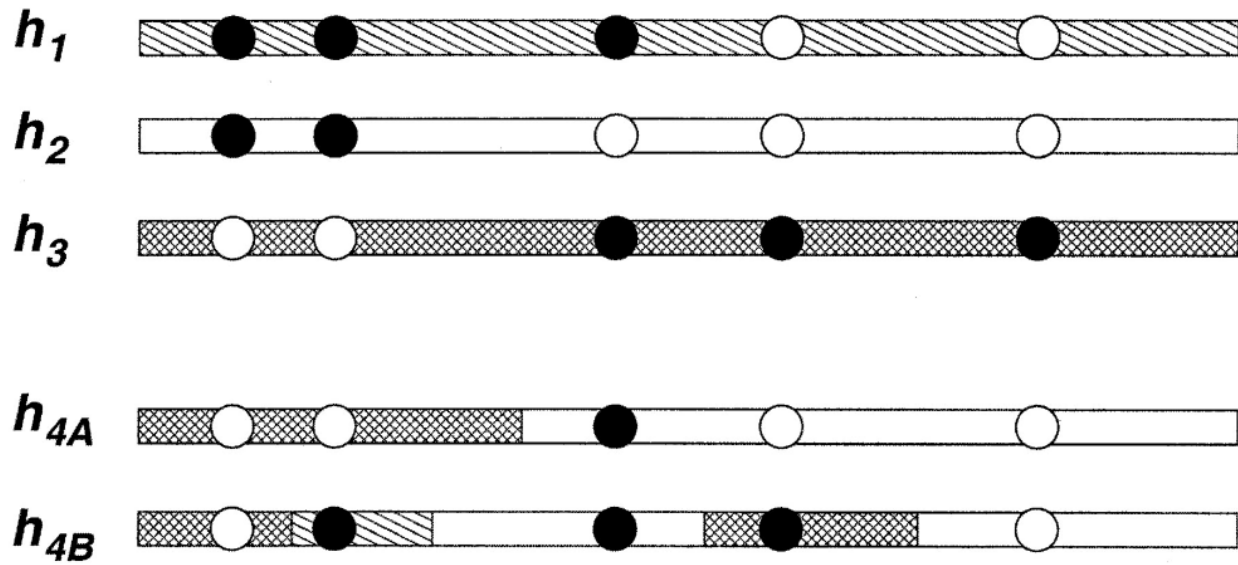
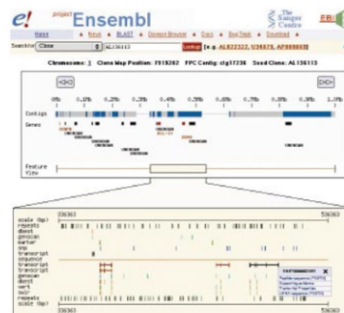


Image from Li and Stephens, Genetics 2003

## HMMs for Identifying Protein Coding Genes



Human genome paper used HMMs to estimate number of protein coding genes

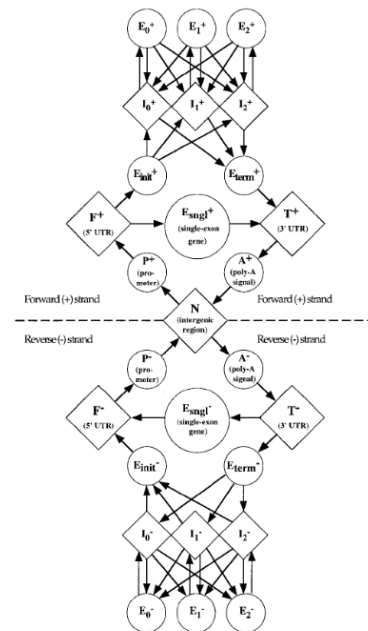
**JMB**

J. Mol. Biol. (1997) 268, 70-94



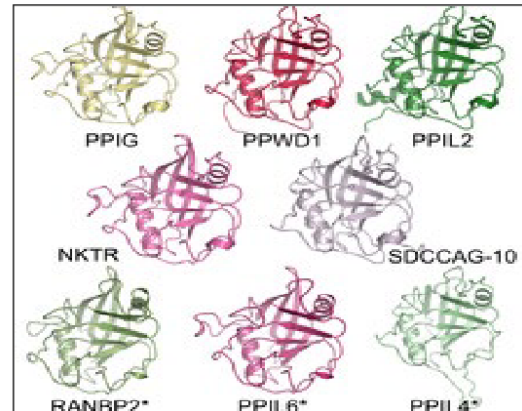
## Prediction of Complete Gene Structures in Human Genomic DNA

Chris Burge\* and Samuel Karlin

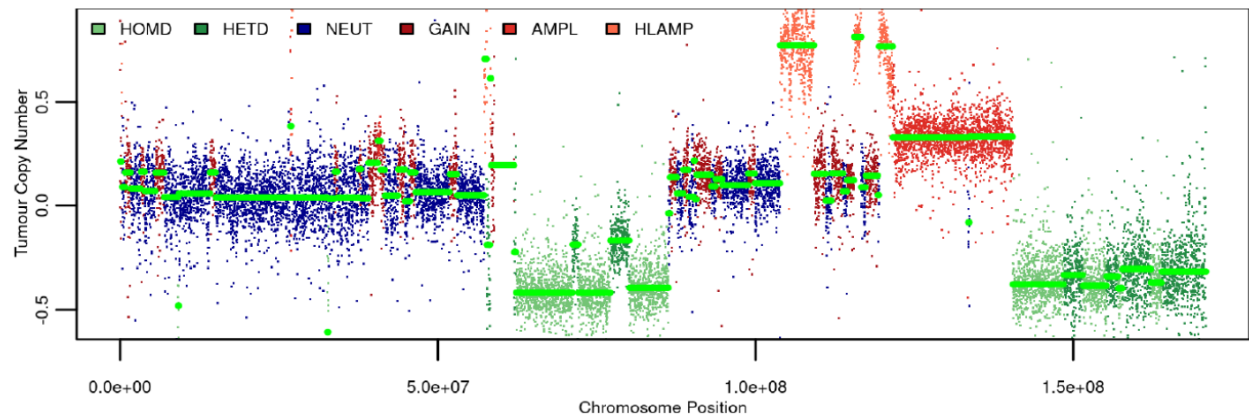


Genscan

## Profile HMMs



## HMMs for Calling Copy Number Variation



- In genomics problems we often want to "annotate" or "cluster" positions along a sequence
- Considering each position along a sequence in isolation is often inadequate.
- For example, promoter regions (i.e., regions near transcription start sites) have elevated frequency of G or C nucleotides
- Would like to annotate each base of the genome as likely promoter ('P') or background ('B') based on this

...TAA**GAATTGTGT**CATAAACCCGGACGCGCTCGCC**TAGAA**GATT**GAG**ATTA...

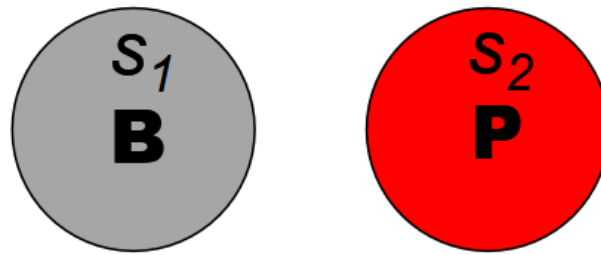
Example “true” annotation we would want to infer:

The diagram illustrates a genomic region divided into three segments: Background, Promoter, and Background. The top bar shows a sequence of nucleotides (A, C, G, T) with the central Promoter region highlighted in red. Below the bar, brackets indicate the extent of each segment: Background (grey), Promoter (red), and Background (grey).



## Markov Chain

Has  $K$  - states, denoted  $s_1, \dots, s_k$ .



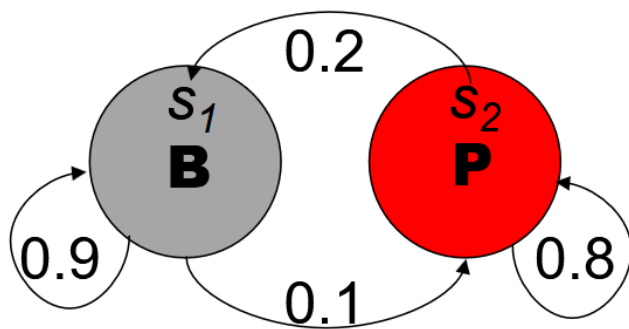
In this example,  $K = 2$  and denote  $s_1$  and  $s_2$  by 'B' and 'P', respectively.

- There are discrete sequential positions denoted  $i = 1, i = 2, \dots$
- At the  $i$ -th position the chain is in exactly one of the  $K$  states, denoted by  $\pi_i$ , and  $\pi_i \in \{s_1, s_2, \dots, s_k\}$
- We start the chain ( $i = 1$ ) based on an initial probability distribution determined by  $\tau_1, \dots, \tau_k$
- The current state determines the probability distribution of the next state. We have parameters

$$a_{kj} = P(\pi_{i+1} = s_j | \pi_i = s_k)$$

corresponding to arcs in the figure below

- Probabilities out of a state sum to 1.



$$\tau_1 = \tau_B = 1 \text{ and } \tau_2 = \tau_P = 0$$

$$a_{11} = 0.9; a_{12} = 0.1$$

$$a_{21} = 0.2; a_{22} = 0.8$$

### Markov Property or Assumption:

$\pi_{i+1}$  is conditionally independent of  $\{\pi_{i-1}, \pi_{i-2}, \dots, \pi_1\}$  given  $\pi_i$  that is

$$P(\pi_{i+1}=s_j | \pi_i=s_k) = P(\pi_{i+1} = s_j | \pi_i = s_k, \text{ any earlier history})$$

### Example Path

Consider the path:

$$\begin{aligned} \pi_1 &= B & i &= 1 \text{ (starting)} \\ \pi_2 &= B & i &= 2 \text{ (prob. 0.9)} \\ \pi_3 &= P & i &= 3 \text{ (prob. 0.1)} \end{aligned}$$

- HMMs extend Markov chains by allowing states to "emit" observations with different probabilities that depend on the state
- Let  $\{\sigma_1, \dots, \sigma_M\}$  be a set of symbols that are possible observations (e.g.,  $\{A, C, G, T\}$ )

- Let  $x = x_1x_2 \dots x_n$  denote a sequence of observations (e.g., AACGGC)
- We have emission parameters corresponding to probability of observing a symbol emitted in a given state:

$$b_k(\sigma_m) = P(x_i = \sigma_m | \pi_i = s_k)$$

for  $m = 1, \dots, M$  possible symbols  $\sigma_m$  and  $k = 1, \dots, K$  states

We will use the following emission parameters for this example:

Emission parameters:	$s_1 (B)$	$s_2 (P)$
A	1/4	1/6
C	1/4	1/3
G	1/4	1/3
T	1/4	1/6

- First, we start from  $\pi_1$ .  $\pi_1 = B, i = 1$ . Therefore, the probability is 1/4. Suppose we generate  $x_1 = A$ ; according to our emission parameters, the probability is 1/4
- When we move to the next base,  $\pi_2 = B, i = 2$ . The probability is 9/10, according to our model. ( $B \rightarrow B$ ) Suppose we generate  $x_2 = G$ ; according to our emission parameters, the probability is 1/4.
- Finally, to the third base,  $\pi_3 = P, i = 3$ . The probability is 1/10, according to our model. ( $B \rightarrow P$ ) Suppose we generate  $x_3 = C$ ; according to our emission parameters, the probability is 1/3.

## HMM Definition

An HMM,  $\lambda$ , is a 5-tuple consisting of:

- $\Sigma$  - alphabet of  $M$  emitted symbols  $\{\sigma_1, \dots, \sigma_M\}$
- A set  $S$  of  $K$  hidden states  $\{s_1, \dots, s_K\}$
- A set  $S$  of  $K$  initial state probabilities  $\{\tau_1, \dots, \tau_K\}$
- A  $K$  by  $K$  set of state transition probabilities

$$a_{kj} = P(\pi_{i+1} = s_j | \pi_i = s_k)$$

- A  $K$  by  $M$  set of state emission probabilities

$$b_k(\sigma_m) = P(x_i = \sigma_m | \pi_i = s_k)$$

## HMM Problems and Algorithms

1. State Emission (Forward-Backward Algorithm, Dynamic Programming)
2. Most Probable Path (Viterbi, Dynamic Programming)
3. Learning a HMM (Baum-Welch, Expectation-Maximization)

## Problem 1: State Emission

Given a sequence of observations of length  $n$ ,  $x = x_1x_2 \dots x_n$ , and a model  $\lambda$  we want to compute:

$$P(\pi_i = s_k | x, \lambda)$$

that is the probability at position  $i$  the state is  $s_k$  given the observation sequence.

For example, we observe the sequence AGC and want to infer what is the probability at position 3 we are in state 'P'?

$$P(\pi_3 = P | x = AGC, \lambda)$$



## Brute Force Approach to State Estimation

$$\begin{aligned}
 P(\pi_i = s_k | x, \lambda) &= \sum_{\pi \text{ s.t. } \pi_i = s_k} P(\pi | x, \lambda) \\
 &= \sum_{\pi \text{ s.t. } \pi_i = s_k} P(\pi, x | \lambda) / P(x | \lambda) \\
 &= \frac{\sum_{\pi \text{ s.t. } \pi_i = s_k} P(\pi | \lambda) P(x | \pi, \lambda)}{\sum_{\pi} P(\pi | \lambda) P(x | \pi, \lambda)}
 \end{aligned}$$

- $\pi$  - hidden path variables
- $x$  - observations
- $i$  - position
- $s_k$  - state
- $\lambda$  - model parameters

For example:

$$\begin{aligned}
 P(\pi_i = s_k | x, \lambda) &= P(\pi = BBP | x = AGC, \lambda) + P(\pi = BPP | x = AGC, \lambda) + P(\pi = PBP | x = AGC, \lambda) + P(\pi = PPP | x = AGC, \lambda) \\
 &= [P(\pi = BBP, x = AGC | \lambda) + P(\pi = BPP, x = AGC | \lambda) + P(\pi = PBP, x = AGC | \lambda) + P(\pi = PPP, x = AGC | \lambda)] / P(x = AGC | \lambda) \\
 &= [P(\pi = BBP | \lambda) P(x = AGC | \pi = BBP, \lambda) + P(\pi = BPP | \lambda) P(x = AGC | \pi = BPP, \lambda) + P(\pi = PBP | \lambda) P(x = AGC | \pi = PBP, \lambda) + P(\pi = PPP | \lambda) P(x = AGC | \pi = PPP, \lambda)] / P(x = AGC | \lambda)
 \end{aligned}$$

The terms in the final equation can be computed directly.

How do we compute  $P(\pi | \lambda)$  for an arbitrary path  $\pi$  of length 3?

$$\begin{aligned}
 P(\pi | \lambda) &= P(\pi_1, \pi_2, \pi_3 | \lambda) \\
 &= P(\pi_1 | \lambda) P(\pi_2, \pi_3 | \lambda, \pi_1) \\
 &= P(\pi_1 | \lambda) P(\pi_2 | \lambda, \pi_1) P(\pi_3 | \lambda, \pi_1, \pi_2) \\
 &= P(\pi_1 | \lambda) P(\pi_2 | \lambda, \pi_1) P(\pi_3 | \lambda, \pi_2)
 \end{aligned}$$

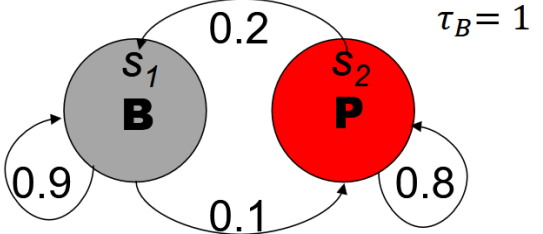
How do we compute  $P(x | \pi, \lambda)$  for an arbitrary path  $\pi$  of length 3 and observation sequence  $x$ ?

$$\begin{aligned}
 P(x | \pi, \lambda) &= P(x_1, x_2, x_3 | \pi_1, \pi_2, \pi_3, \lambda) \\
 &= P(x_1 | \pi_1, \lambda) P(x_2 | \pi_2, \lambda) P(x_3 | \pi_3, \lambda)
 \end{aligned}$$

Example in the case  $\pi = BBP$  and  $x = AGC$

$$\begin{aligned}
 &= P(x_1 = A | \pi_1 = \mathbf{B}, \lambda) P(x_2 = G | \pi_2 = \mathbf{B}, \lambda) P(x_3 = C | \pi_3 = \mathbf{P}, \lambda) \\
 &= \frac{1}{4} \times \frac{1}{4} \times \frac{1}{3} = \frac{1}{48}
 \end{aligned}$$

	<b>B</b>	<b>P</b>
A	1/4	1/6
C	1/4	1/3
G	1/4	1/3
T	1/4	1/6



In general, what is the complexity of the brute force approach for a sequence of length  $n$  and  $K$  states?

- $O(nk^n)$
- This sucks! We need a more efficient approach.

## State Estimation with Forward and Backward Variables

Define:

- Forward variables:  $\alpha_i(k) = P(x_1 x_2 \dots x_i, \pi_i = s_k | \lambda)$
- Backward variables:  $\beta_i(k) = P(x_{i+1} x_{i+2} \dots x_n | \pi_i = s_k, \lambda)$
- Forward variables  $\alpha_i(k)$  is the probability under the model  $\lambda$  that both:
  - We have seen the first  $i$  observations
  - We are in state  $s_k$  at the  $i$ -th position
- Backward variables  $\beta_i(k)$  is the probability under the model  $\lambda$  given we are in state  $s_k$  at position  $i$  that:
  - We have seen the last  $n - i$  observations
- Claim:

$$P(\pi_i = s_k | x, \lambda) = \frac{\alpha_i(k) \beta_i(k)}{\sum_{j=1}^K \alpha_i(j) \beta_i(j)}$$

- We can efficiently compute state estimates variables for all positions  $i$  and states  $k$  if we can efficiently compute forward and backward variables for them.
- We need to show that this claim is true and that we can compute the forward and backward variables efficiently.

### Proof

Let's break down the claim:

$$\begin{aligned} &= \alpha_i(k) \beta_i(k) = P(x_1 x_2 \dots x_i, \pi_i = s_k | \lambda) P(x_{i+1} x_{i+2} \dots x_n | \pi_i = s_k, \lambda) \\ &= P(x_1 x_2 \dots x_i, \pi_i = s_k | \lambda) P(x_{i+1} x_{i+2} \dots x_n | x_1 x_2 \dots x_i, \pi_i = s_k, \lambda) \\ &= P(x_1 x_2 \dots x_i, x_{i+1} x_{i+2} \dots x_n, \pi_i = s_k | \lambda) \\ &= P(x, \pi_i = s_k | \lambda) \end{aligned}$$

Therefore, we get

$$\sum_{j=1}^K \alpha_i(j) \beta_i(j) = \sum_{j=1}^K P(x, \pi_i = s_j | \lambda) = P(x | \lambda)$$

We have shown the claim is true. Now, we have to find a way to compute the variables.

### Computing Forward Variables

$$\alpha_i(k) = P(x_1 x_2 \dots x_i, \pi_i = s_k | \lambda)$$

Forward variable  $\alpha_i(k)$  is the probability under the model that both:

- We have seen the first  $i$  observations
- We are in state  $s_k$  at the  $i$ -th position

We use dynamic programming to compute this efficiently.

Beginning with the base case:

$$\begin{aligned} \alpha_1(k) &= P(x_1, \pi_1 = s_k | \lambda) \quad \text{for } k = 1, \dots, K \\ &= P(\pi_1 = s_k | \lambda) P(x_1 | \pi_1 = s_k, \lambda) \\ &= \tau_k b_k(x_1) \end{aligned}$$



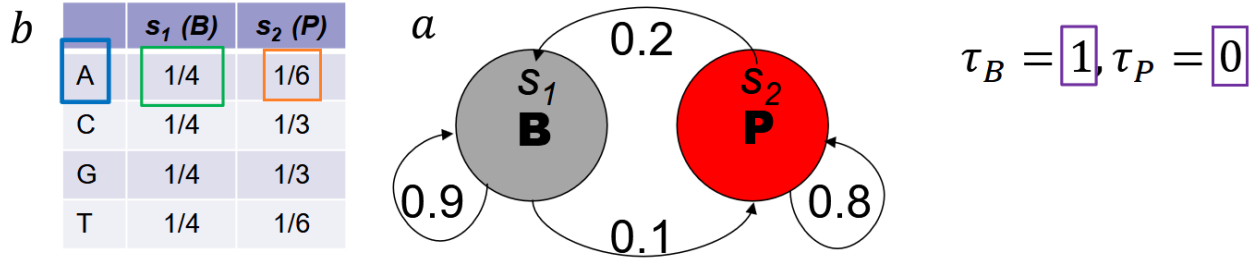
- $\tau_k$  is the initial state probability
- $b_k$  is the emission probability for the first observation

Now, for the induction step:

$$\begin{aligned}
\alpha_{i+1}(j) &= P(x_1 x_2 \dots x_i x_{i+1}, \pi_{i+1} = s_j | \lambda) \\
&= \sum_{k=1}^K P(x_1 x_2 \dots x_i, \pi_i = s_k, x_{i+1}, \pi_{i+1} = s_j | \lambda) \\
&= \sum_{k=1}^K P(x_{i+1}, \pi_{i+1} = s_j | x_1 x_2 \dots x_i, \pi_i = s_k, \lambda) P(x_1 x_2 \dots x_i, \pi_i = s_k | \lambda) \\
&= \sum_{k=1}^K P(x_{i+1}, \pi_{i+1} = s_j | \pi_i = s_k, \lambda) \alpha_i(k) \\
&= \sum_{k=1}^K P(\pi_{i+1} = s_j | \pi_i = s_k, \lambda) P(x_{i+1} | \pi_{i+1} = s_j, \lambda) \alpha_i(k) \\
&= \sum_{k=1}^K a_{kj} b_j(x_{i+1}) \alpha_i(k)
\end{aligned}$$

- where  $\alpha_{kj}$  is the transmission probability
- $b_j(x_{i+1})$  is the emission probability for the current observation
- $\alpha_i(k)$  is the previous forward variable
- The time complexity to compute all forward variables is  $O(nK^2)$  since time for one position and state is  $O(K)$  and need to do it for all  $nK$  pairs of positions and states.

### Forward Variable Example

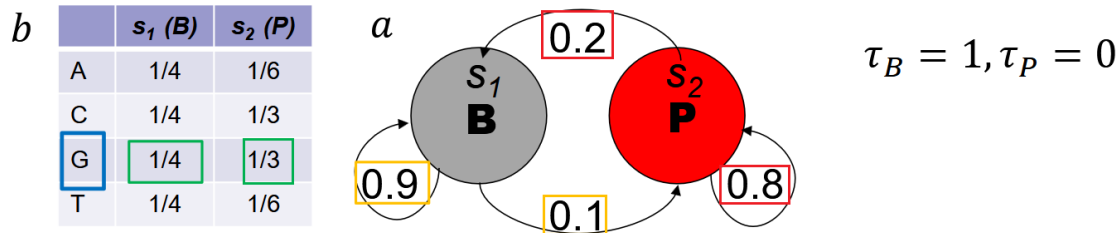


Suppose we observe the sequence  $x = \text{AGC}$

Recall

$$\begin{aligned}
\alpha_1(k) &= \tau_k b_k(x_1) & \alpha_{i+1}(j) &= \sum_{k=1}^K a_{kj} b_j(x_{i+1}) \alpha_i(k) \\
\alpha_1(B) &= 1 \times \frac{1}{4} = \frac{1}{4} & \alpha_1(P) &= 0 \times \frac{1}{6} = 0
\end{aligned}$$

Repeat for  $\alpha_2$  and  $\alpha_3 \dots$



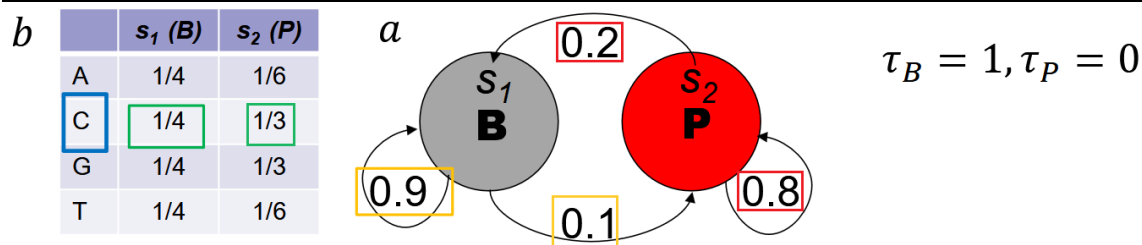
Suppose we observe the sequence  $x = \text{AGC}$

Recall

$$\alpha_1(k) = \tau_k b_k(x_1) \quad \alpha_{i+1}(j) = \sum_{k=1}^K a_{kj} b_j(x_{i+1}) \alpha_i(k)$$

$$\alpha_1(B) = \frac{1}{4} \quad \alpha_1(P) = 0$$

$$\alpha_2(B) = \frac{9}{10} \times \frac{1}{4} \times \frac{1}{4} + \frac{2}{10} \times \frac{1}{4} \times 0 = \frac{9}{160} \quad \alpha_2(P) = \frac{1}{10} \times \frac{1}{3} \times \frac{1}{4} + \frac{8}{10} \times \frac{1}{3} \times 0 = \frac{1}{120}$$



Suppose we observe the sequence  $x = \text{AGC}$

Recall

$$\alpha_1(k) = \tau_k b_k(x_1) \quad \alpha_{i+1}(j) = \sum_{k=1}^K a_{kj} b_j(x_{i+1}) \alpha_i(k)$$

$$\alpha_1(B) = \frac{1}{4} \quad \alpha_1(P) = 0$$

$$\alpha_2(B) = \frac{9}{160} \quad \alpha_2(P) = \frac{1}{120}$$

$$\alpha_3(B) = \frac{9}{10} \times \frac{1}{4} \times \frac{9}{160} + \frac{2}{10} \times \frac{1}{4} \times \frac{1}{120} = 0.013$$

$$\alpha_3(P) = \frac{1}{10} \times \frac{1}{3} \times \frac{9}{160} + \frac{8}{10} \times \frac{1}{3} \times \frac{1}{120} = 0.0041$$

### Computing Backwards Variables

$$\beta_i(k) = P(x_{i+1}x_{i+2}\dots x_n | \pi_i = s_k, \lambda)$$

Backward variable  $\beta_i(k)$  is the probability under the model given we are in state  $s_k$  at position  $i$  that:

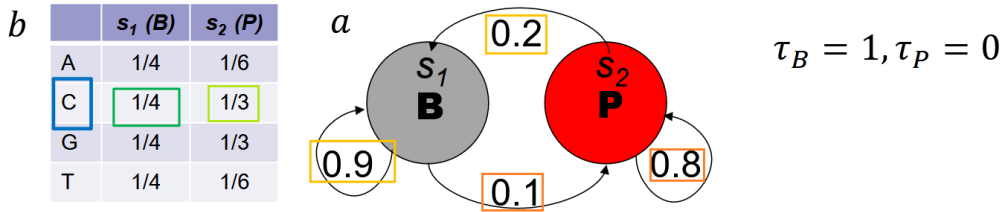
- We have seen the last  $n - i$  observations

Base case:  $\beta_n(k) = 1$  for  $k = 1, \dots, K$ . (By definition, probability of observing nothing after the last position)

$$\begin{aligned}
\beta_i(k) &= P(x_{i+1}x_{i+2} \dots x_n | \pi_i = s_k, \lambda) \\
&= \sum_{j=1}^K P(x_{i+1}x_{i+2} \dots x_n, \pi_{i+1} = s_j | \pi_i = s_k, \lambda) \\
&= \sum_{j=1}^K P(\pi_{i+1} = s_j | \pi_i = s_k, \lambda) P(x_{i+1}x_{i+2} | \pi_i = s_k, \pi_{i+1} = s_j, \lambda) \\
&= \sum_{j=1}^K P(\pi_{i+1} = s_j | \pi_i = s_k, \lambda) P(x_{i+1}x_{i+2} \dots x_n | \pi_{i+1} = s_j, \lambda) \\
&= \sum_{j=1}^K P(\pi_{i+1} = s_j | \pi_i = s_k, \lambda) P(x_{i+1} | \pi_{i+1} = s_j, \lambda) P(x_{i+2} \dots x_n | \pi_{i+1} = s_j, \lambda) \\
&= \sum_{j=1}^K a_{kj} b_j(x_{i+1}) \beta_{i+1}(j)
\end{aligned}$$

- $a_{kj}$  is the transition probability
- $b_j(x_{i+1})$  is the emission probability for the next observation
- $\beta_{i+1}(j)$  is the next backward variable
- The time complexity to compute all backwards variables is  $O(nK^2)$  since time for one position and state is  $O(K)$  and need to do it for all  $nK$  pairs of positions and states

### Backward Variable Example



Suppose we observe the sequence  $x = AGC$

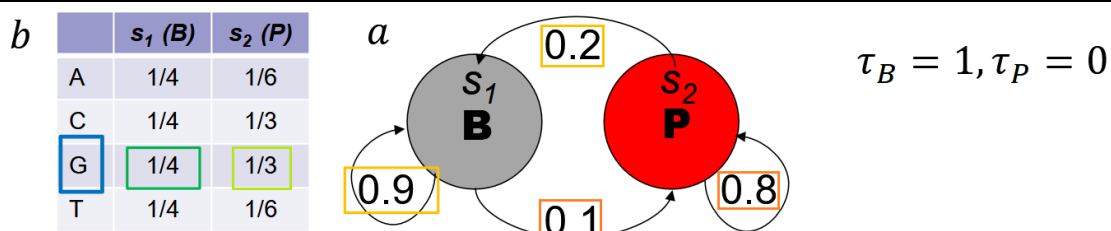
Recall

$$\beta_i(k) = \sum_{j=1}^K a_{kj} b_j(x_{i+1}) \beta_{i+1}(j)$$

$$\beta_3(B) = 1$$

$$\beta_3(P) = 1$$

$$\begin{aligned}
\beta_2(B) &= \frac{9}{10} \times \frac{1}{4} \times 1 + \frac{1}{10} \times \frac{1}{3} \times 1 \\
&= \frac{31}{120}
\end{aligned}
\quad
\begin{aligned}
\beta_2(P) &= \frac{2}{10} \times \frac{1}{4} \times 1 + \frac{8}{10} \times \frac{1}{3} \times 1 \\
&= \frac{19}{60}
\end{aligned}$$



Suppose we observe the sequence  $x = \text{AGC}$

Recall

$$\beta_i(k) = \sum_{j=1}^K a_{kj} b_j(x_{i+1}) \beta_{i+1}(j)$$

$$\beta_3(B) = 1$$

$$\beta_3(P) = 1$$

$$\beta_2(B) = \frac{31}{120}$$

$$\beta_2(P) = \frac{19}{60}$$

$$\beta_1(B) = \frac{9}{10} \times \frac{1}{4} \times \frac{31}{120} + \frac{1}{10} \times \frac{1}{3} \times \frac{19}{60} = 0.0687$$

$$\beta_1(P) = \frac{2}{10} \times \frac{1}{4} \times \frac{31}{120} + \frac{8}{10} \times \frac{1}{3} \times \frac{19}{60} = 0.0974$$

State Estimation Example

$$P(\pi_i = s_k | x, \lambda) = \frac{\alpha_i(k) \beta_i(k)}{\sum_{j=1}^K \alpha_i(j) \beta_i(j)}$$

$x = \text{AGC}$

$\alpha_1(B) = \frac{1}{4}$	$\alpha_1(P) = 0$	$\beta_1(B) = 0.0687$	$\beta_1(P) = 0.0974$
$\alpha_2(B) = \frac{9}{160}$	$\alpha_2(P) = \frac{1}{120}$	$\beta_2(B) = \frac{31}{120}$	$\beta_2(P) = \frac{19}{60}$
$\alpha_3(B) = 0.013$	$\alpha_3(P) = 0.0041$	$\beta_3(B) = 1$	$\beta_3(P) = 1$

$$P(\pi_1 = P | x, \lambda) = 0$$

$$P(\pi_2 = P | x, \lambda) = \frac{\frac{1}{120} \times \frac{19}{60}}{\frac{1}{120} \times \frac{19}{60} + \frac{9}{160} \times \frac{31}{120}} = 0.15$$

$$P(\pi_3 = P | x, \lambda) = \frac{0.0041}{0.0041 + 0.013} = 0.24$$

Multiplication by 1 is omitted.

From this result, we can calculate:

$$P(\pi_1 = \mathbf{P}|x, \lambda) = 0$$

$$P(\pi_1 = \mathbf{B}|x, \lambda) = 1$$

$$P(\pi_2 = \mathbf{P}|x, \lambda) = 0.15$$

$$P(\pi_2 = \mathbf{B}|x, \lambda) = 0.85$$

$$P(\pi_3 = \mathbf{P}|x, \lambda) = 0.24$$

$$P(\pi_3 = \mathbf{B}|x, \lambda) = 0.76$$

$$\text{since } P(\pi_i = \mathbf{B}|x, \lambda) = 1 - P(\pi_i = \mathbf{P}|x, \lambda)$$

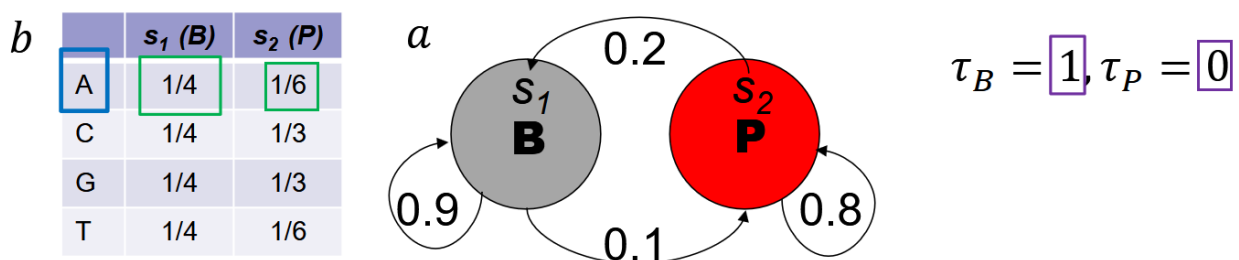
- One limitation of this is that forward and backward variables can approach limits of floating point representations. (We're multiplying lots of fractions)
- To increase numerical stability, we can first normalize the  $\alpha_i$  and  $\beta_i$  values by constants.

## Scaling

$$P(\pi_i = s_i | x, \lambda) = \frac{\alpha_i(k)\beta_i(k)}{\sum_{j=1}^K \alpha_i(j)\beta_i(j)} = \frac{c_i \times \alpha_i(k)\beta_i(k)}{\sum_{j=1}^K c_i \times \alpha_i(j)\beta_i(j)}$$

- One strategy is to normalize  $\alpha_i(j)$  based on the sum of their values after computing them for a position  $i$  in the forward step
- e.g., replace  $\alpha_i(k)$  with  $\frac{\alpha_i(k)}{\sum_{j=1}^K \alpha_i(j)}$
- similarly for the  $\beta_i(j)$  values in the backwards step, replace  $\beta_i(k)$  with  $\frac{\beta_i(k)}{\sum_{j=1}^K \beta_i(j)}$

## Forward Variable Example with Scaling



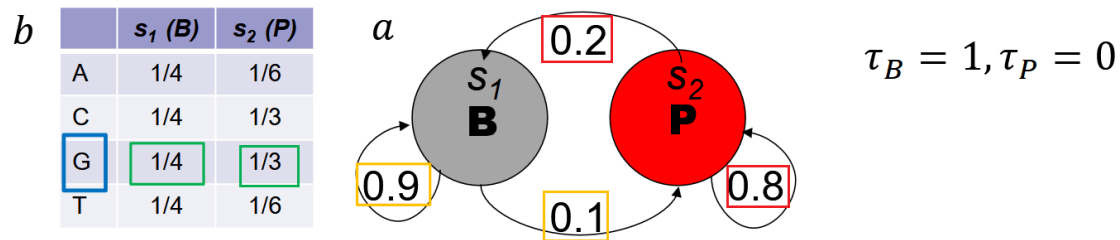
Suppose we observe the sequence  $x = \text{AGC}$

$$\alpha_1(k) \sim \tau_k b_k(x_1)$$

$$\alpha_{i+1}(j) \sim \sum_{k=1}^K a_{kj} b_j(x_{i+1}) \alpha_i(k)$$

$$\alpha_1(B) = 1 \times \frac{1}{4} = \frac{1}{4} \Rightarrow 1$$

$$\alpha_1(P) = 0 \times \frac{1}{6} = 0 \Rightarrow 0$$



Suppose we observe the sequence  $x = \text{AGC}$

$$\alpha_1(k) \sim \tau_k b_k(x_1) \quad \alpha_{i+1}(j) \sim \sum_{k=1}^K a_{kj} b_j(x_{i+1}) \alpha_i(k)$$

$$\alpha_1(B) = 1 \quad \alpha_1(P) = 0$$

$$\alpha_2(B) = \frac{9}{10} \times \frac{1}{4} \times 1 + \frac{2}{10} \times \frac{1}{4} \times 0 = \frac{9}{40} \quad \alpha_2(P) = \frac{1}{10} \times \frac{1}{3} \times 1 + \frac{8}{10} \times \frac{1}{3} \times 0 = \frac{1}{30}$$

$$\Downarrow \quad \Downarrow$$

$$\frac{27}{31} \quad \frac{4}{31}$$

The final forward variables are scaled relative to each other, such that they would add to 1. Repeat this process for every other forward variable.

## Computing Data Likelihood

To compute the data likelihood  $P(x|\lambda)$  before scaling, we did:

- Forward variables:  $\alpha_i(k) = P(x_1 x_2 \dots x_i, \pi_i = s_k | \lambda)$
- Backward variables:  $\beta_i(k) = P(x_{i+1} x_{i+2} \dots x_n | \pi_i = s_k, \lambda)$

$$P(x|\lambda) = \sum_{j=1}^K \alpha_i(j) \beta_i(j)$$

- For  $i = n$ , this becomes:

$$P(x|\lambda) = \sum_{j=1}^K \alpha_n(j)$$

If we have scaling, we now have:

$$P(x|\lambda) = \left( \prod_{i=1}^n \frac{1}{c_i} \right) \sum_{j=1}^K \alpha_n(j)$$

## Posterior Decoding

- How can we use the state estimates to obtain hard assignments (e.g., 'P' or 'B' assignments?)
- We simply take the state with the maximum posterior probability at each position.

Recall from earlier:

$$\begin{aligned}
P(\pi_1 = B|x, \lambda) &= 0 & P(\pi_1 = B|x, \lambda) &= 1 \\
P(\pi_2 = B|x, \lambda) &= 0.15 & P(\pi_2 = B|x, \lambda) &= 0.85 \\
P(\pi_3 = B|x, \lambda) &= 0.24 & P(\pi_3 = B|x, \lambda) &= 0.76
\end{aligned}$$

Therefore, our posterior decoding is BBB.

## Problem 2: Most Probable Path

### Most Probable Path Given Observations

- What is the most probable path  $\pi$  given a sequence of observations  $x_1x_2 \dots x_n$  that is

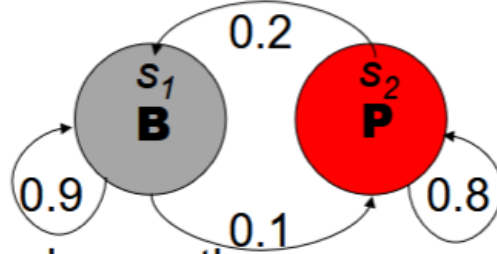
$$\arg \max_{\pi} P(\pi|x_1x_2 \dots x_n, \lambda)$$

- Brute force approach:

$$\begin{aligned}
&\arg \max_{\pi} P(\pi|x_1x_2 \dots x_n, \lambda) \\
&= \arg \max_{\pi} \frac{P(x_1x_2 \dots x_n, \pi|\lambda)}{P(x_1x_2 \dots x_n|\lambda)} \\
&= \arg \max_{\pi} P(x_1x_2 \dots x_n, \pi|\lambda) \\
&= \arg \max_{\pi} P(x_1x_2 \dots x_n|\pi, \lambda)P(\pi|\lambda)
\end{aligned}$$

### Brute Force Example

	$s_1 (B)$	$s_2 (P)$
A	1/4	1/6
C	1/4	1/3
G	1/4	1/3
T	1/4	1/6



$$\tau_B = 1, \tau_P = 0$$

Suppose we observe the sequence  $x = AGC$

$\pi_1$	$\pi_2$	$\pi_3$	$P(x \pi, \lambda)$	$P(\pi \lambda)$	$P(x \pi, \lambda) P(\pi \lambda)$
B	B	B	$1/(4 \times 4 \times 4)$	$1 \times 0.9 \times 0.9$	0.01266
B	B	P	$1/(4 \times 4 \times 3)$	$1 \times 0.9 \times 0.1$	0.00188
B	P	B	$1/(4 \times 3 \times 4)$	$1 \times 0.1 \times 0.2$	0.00042
B	P	P	$1/(4 \times 3 \times 3)$	$1 \times 0.1 \times 0.8$	0.00222
P	B	B	$1/(6 \times 4 \times 4)$	0	0
P	B	P	$1/(6 \times 4 \times 3)$	0	0
P	P	B	$1/(6 \times 3 \times 4)$	0	0
P	P	P	$1/(6 \times 3 \times 3)$	0	0

We would pick BBB since it has the highest probability.

- The time complexity of this approach is  $O(nK^n)$  time.
- We need a more efficient approach!



## Efficient Most Probable Path Computation

Let's define:

$$\delta_i(k) = \max_{\pi_1 \pi_2 \dots \pi_{i-1}} P(\pi_1 \pi_2 \dots \pi_{i-1}, \pi_i = s_k, x_1 x_2 \dots x_i | \lambda)$$

$$mpp_i(k) = \arg \max_{\pi_1 \pi_2 \dots \pi_{i-1}} P(\pi_1 \pi_2 \dots \pi_{i-1}, \pi_i = s_k, x_1 x_2 \dots x_i | \lambda)$$

$mpp_i(k)$  is the path of length  $i - 1$  with maximum probability of

1. Occurring
2. Ending up in state  $s_k$  at position  $i$
3. Producing output  $x_1 x_2 \dots x_i$

$\delta_i(k)$  is probability of  $mpp_i(k)$  doing all this.

## The Viterbi Algorithm

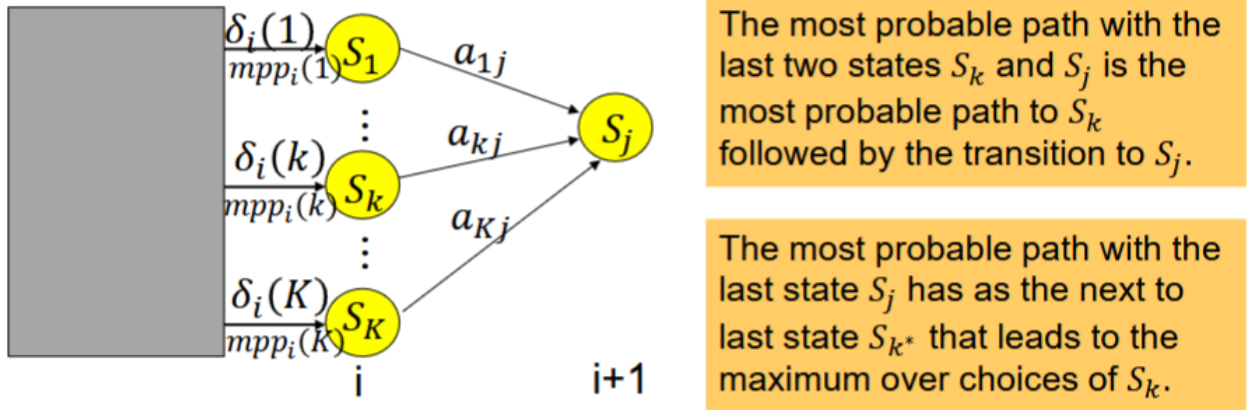
Refer to the equations above.

Base case  $i = 1$  (only one choice for each  $k$ )

$$\begin{aligned} \delta_1(k) &= P(\pi_1 = s_k, x_1 | \lambda) \\ &= P(\pi_1 = s_k | \lambda) P(x_1 | \pi_1 = s_k, \lambda) \\ &= \tau_k b_k(x_1) \end{aligned}$$

Inductive case:

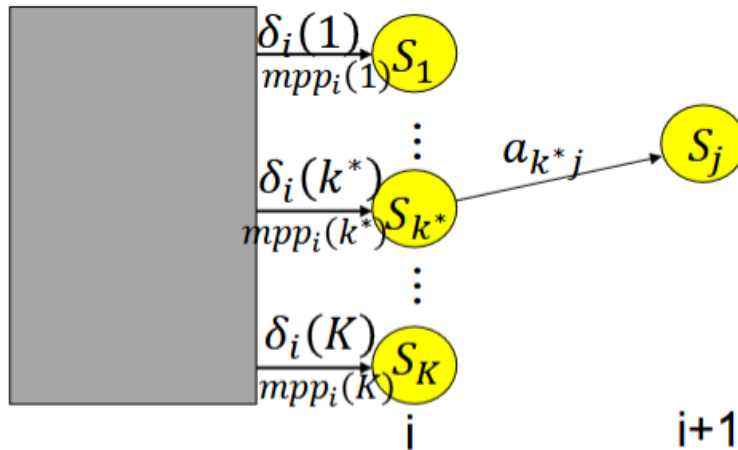
$$\delta_{i+1}(j) = \max_{\pi_1 \pi_2 \dots \pi_i} P(\pi_1 \pi_2 \dots \pi_i, \pi_{i+1} = s_j, x_1 x_2 \dots x_i x_{i+1} | \lambda)$$



$$\begin{aligned} &= \max_{\pi_1 \pi_2 \dots \pi_i} P(\pi_1 \pi_2 \dots \pi_i = s_k, x_1 x_2 \dots x_i | \lambda) P(\pi_{i+1} = s_j | \pi_i = s_k, \lambda) P(x_{i+1} | \pi_{i+1} = s_j, \lambda) \\ &= \max_k \delta_i(k) a_{kj} b_j(x_{i+1}) \end{aligned}$$

- $\delta_i(k)$  is the previous mpp probability
- $a_{kj}$  is the transition probability
- $b_j(x_{i+1})$  is the emission probability for the current observation

Now, we define  $k^* = \arg \max_k \delta_i(k) a_{kj} b_j(x_{i+1}) mpp_{i+1}(j) = mpp_i(k^*) k^*$

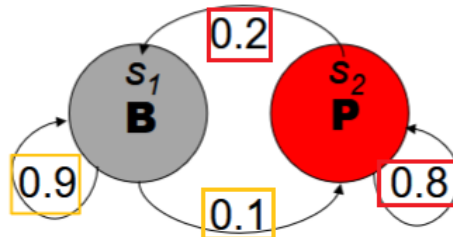


Record the maximizing choice for each state  $j$  at each position. To get the Viterbi path, trace back from maximizing choice at the last position.

- The time complexity of the Viterbi algorithm is  $O(nK^2)$  since time for one position and state is  $O(K)$  and need to do it for all  $nK$  pairs of positions and states.
- Note: faster run-times are possible if some transition probabilities are zero.

### Viterbi Path Example

	B	P
A	1/4	1/6
C	1/4	1/3
G	1/4	1/3
T	1/4	1/6



$$\tau_B = 1, \tau_P = 0$$

Suppose we observe the sequence  $x = \text{AGC}$

Recall  $\delta_1(k) = \tau_k b_k(x_1)$      $\delta_{i+1}(j) = \max_k \delta_i(k) a_{kj} b_j(x_{i+1})$

$$\delta_1(B) = \frac{1}{4}$$

$$\delta_1(P) = 0$$

$$\delta_2(B) = \max \left\{ \frac{1}{4} \times \frac{9}{10} \times \frac{1}{4}, 0 \times \frac{2}{10} \times \frac{1}{4} \right\}$$

$$\delta_2(P) = \max \left\{ \frac{1}{4} \times \frac{1}{10} \times \frac{1}{3}, 0 \times \frac{8}{10} \times \frac{1}{3} \right\}$$

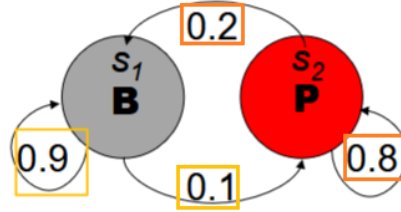
Record B as maximizing choice

Record B as maximizing choice

$$= \frac{9}{160}$$

$$= \frac{1}{120}$$

	B	P
A	1/4	1/6
C	1/4	1/3
G	1/4	1/3
T	1/4	1/6



$$\tau_B = 1, \tau_P = 0$$

Suppose we observe the sequence  $x = \text{AGC}$

Recall  $\delta_1(k) = \tau_k b_k(x_1)$      $\delta_{i+1}(j) = \max_k \delta_i(k) a_{kj} b_j(x_{i+1})$

$$\delta_1(B) = \frac{1}{4}$$

$$\delta_1(P) = 0$$

$$\delta_2(B) = \frac{9}{160}$$

$$\delta_2(P) = \frac{1}{120}$$

Record B as maximizing choice

Record B as maximizing choice

$$\delta_3(B) = \max \left\{ \begin{array}{l} \frac{9}{160} \times \frac{9}{10} \times \frac{1}{4} \\ \frac{1}{120} \times \frac{2}{10} \times \frac{1}{4} \end{array} \right\} = 0.013$$

$$\delta_3(P) = \max \left\{ \begin{array}{l} \frac{9}{160} \times \frac{1}{10} \times \frac{1}{3} \\ \frac{1}{120} \times \frac{8}{10} \times \frac{1}{3} \end{array} \right\} = 0.0022$$

Record B as maximizing choice

Record P as maximizing choice

To extract the path, we record maximizing choice of the last choice, then trace back.

$$\delta_1(B) = \frac{1}{4}$$

$$\delta_1(P) = 0$$

$$\delta_2(B) = \frac{9}{160}$$

$$\delta_2(P) = \frac{1}{120}$$

Record B as maximizing choice

Record B as maximizing choice

$$\delta_3(B) = 0.013 \quad \leftarrow \text{max}$$

$$\delta_3(P) = 0.0022$$

Record B as maximizing choice

Record P as maximizing choice

### Posterior Decoding vs. Viterbi Path

- Path from posterior decoding can be different from Viterbi path.
- Posterior decoding path can have zero probability through model, but still be informative about each position in isolation
- Viterbi path could represent very small amount of probability based on all paths

### Problem 3: Learning a HMM

- Problem: We are given an observation sequence  $x$  and want to learn the HMM parameters (emission, transition, and initial parameters). For simplicity, we are assuming a single sequence.
- There are two settings:
  - Supervised Learning: We are also given the true path  $\pi$  that generated observation sequence  $x$

- Unsupervised Learning: We only are given observation sequence  $x$

## HMM Supervised Learning

- Given path  $\pi$  and observation sequence  $x$  we want to set HMM  $\lambda$  parameters to maximize their likelihood. i.e.,

$$\arg \max_{\lambda} P(\pi, x | \lambda)$$

- For emission parameters

$$b_k(\sigma_m) = P(x_i = \sigma_m | \pi_i = s_k)$$

maximum likelihood estimate is empirical frequency, i.e.,

$$\frac{\text{number of times in state } k \text{ and observe symbol } \sigma_m}{\text{number of times in state } k}$$

- For transition parameters

$$a_{kj} = P(\pi_{i+1} = s_j | \pi_i = s_k)$$

maximum likelihood estimate is empirical frequency, i.e.,

$$\frac{\text{number of times transition from state } k \text{ to state } j}{\text{number of times transition from state } k}$$

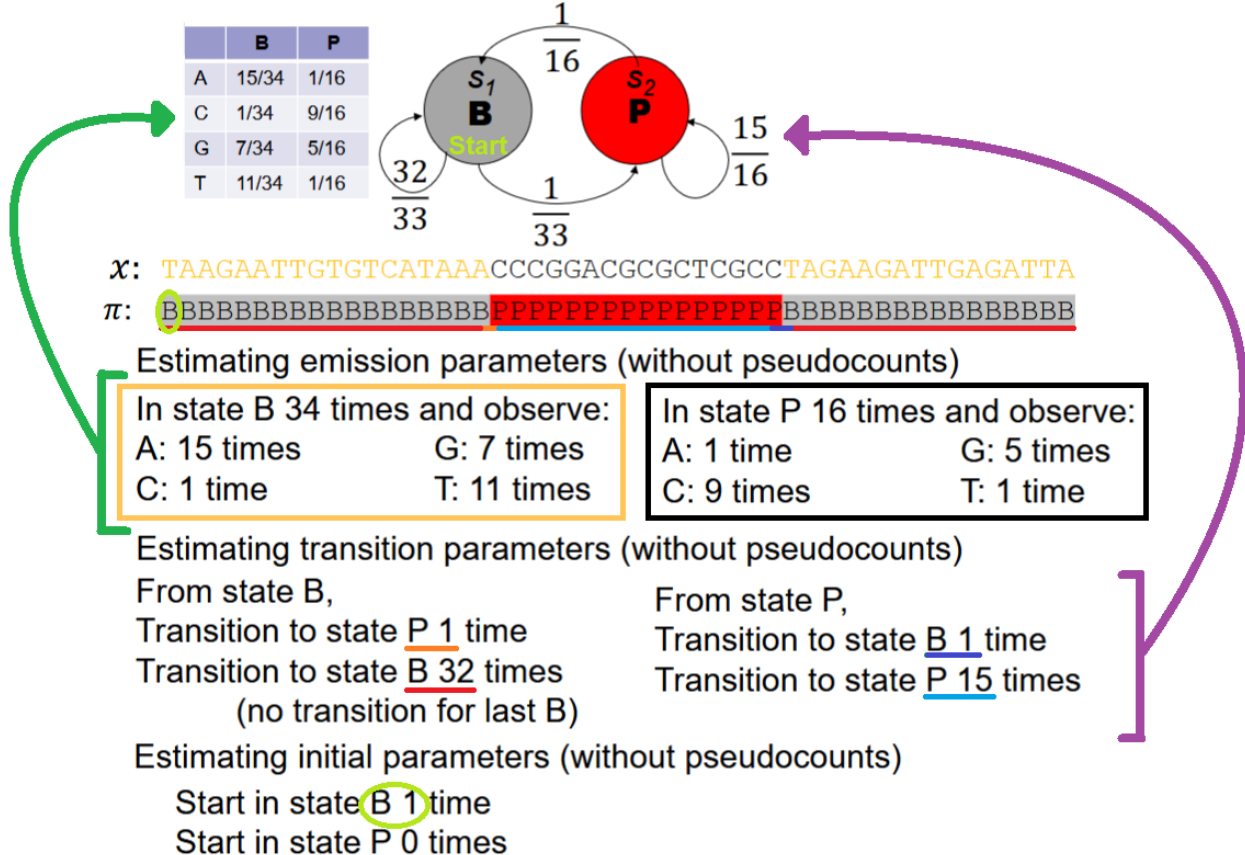
- For initial parameters

$$\tau_j = P(\pi_i = s_j)$$

maximum likelihood estimate is empirical frequency, i.e.,

$$\frac{\text{number of times start a sequence in state } j}{\text{number of sequences}}$$

- Could optionally use pseudocounts here



## HMM Unsupervised Learning

- Given observation sequence  $x$  we want to set HMM  $\lambda$  parameters to maximize their likelihood, i.e.

$$\arg \max_{\lambda} P(x|\lambda)$$

- We are trying to estimate the parameters of the HMM just from the observation sequence. We will assume given the number of state.
  - For this example, we will assume there are two states,  $s_1$  and  $s_2$ .
  - This is analogous to estimating gaussian mixture model cluster parameters for unlabeled data points, or estimating parameters of a PWM from a set of DNA sequences.

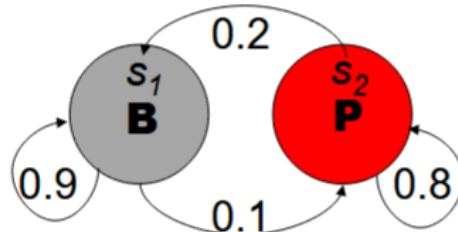
## Unsupervised Learning with EM

- (E-Step) If we had an estimate on the parameters HMM and weren't concerned with time-complexity we could directly compute probability of every path
- (M-step) If we had the probability of every path and weren't concerned with time-complexity we could directly compute the most likely parameters of the HMM
- This is slow, because we have to do this for every possible sequence of states, which is exponential time.

### Example: Brute Force EM

Based on the sequence  $x = \text{AGC}$ , we can fill out the table:

	$s_1 (B)$	$s_2 (P)$
A	1/4	1/6
C	1/4	1/3
G	1/4	1/3
T	1/4	1/6



$$\tau_B = 1, \tau_P = 0$$

Suppose we observe the sequence  $x = \text{AGC}$  and want to re-estimate current HMM parameters

$\pi_1$	$\pi_2$	$\pi_3$	$P(x \pi, \lambda)$	$P(\pi \lambda)$	$P(x \pi, \lambda) P(\pi \lambda)$	$P(\pi x, \lambda)$
B	B	B	$1/(4 \times 4 \times 4)$	$1 \times 0.9 \times 0.9$	0.01266	0.737
B	B	P	$1/(4 \times 4 \times 3)$	$1 \times 0.9 \times 0.1$	0.00188	0.109
B	P	B	$1/(4 \times 3 \times 4)$	$1 \times 0.1 \times 0.2$	0.00042	0.024
B	P	P	$1/(4 \times 3 \times 3)$	$1 \times 0.1 \times 0.8$	0.00222	0.129
P	B	B	$1/(6 \times 4 \times 4)$	0	0	0
P	B	P	$1/(6 \times 4 \times 3)$	0	0	0
P	P	B	$1/(6 \times 3 \times 4)$	0	0	0
P	P	P	$1/(6 \times 3 \times 3)$	0	0	0

Now, we need to re-estimate the HMM parameters.

- For emission parameters maximum likelihood estimate of

$$b_k(\sigma_m) = P(x_i = \sigma_m | \pi_i = s_k)$$

now becomes:

$$\frac{\text{expected number of times in state } k \text{ and observe symbol } \sigma_m}{\text{expected number of times in state } k}$$

For the top left box of the emission parameters:

$$\begin{aligned} & \frac{\text{expected number of times in state } B \text{ and observe symbol } A}{\text{expected number of times in state } B} \\ &= \frac{0.737 + 0.109 + 0.024 + 0.129}{3 \times 0.737 + 2 \times 0.109 + 2 \times 0.024 + 1 \times 0.129} = 0.384 \end{aligned}$$

- Symbol  $A$  appears at  $\pi_1$  in the sequence, hence the numerator is the sum of all state  $B$  in position 1.

Next, for the second from the top box, we get

$$\begin{aligned} & \frac{\text{expected number of times in state } B \text{ and observe symbol } C}{\text{expected number of times in state } B} \\ &= \frac{0.737 + 0.024}{3 \times 0.737 + 2 \times 0.109 + 2 \times 0.024 + 1 \times 0.129} = 0.29 \end{aligned}$$

- Symbol  $C$  appears in the sequence at  $\pi_3$  in the sequence, hence the numerator is the sum of all state  $B$  in position 3.

If we repeat this process for all the cells in the emission table, we get the following:

	$s_1 (B)$	$s_2 (P)$
A	0.38	0
C	0.29	0.61
G	0.32	0.39
T	0	0

Next, we want to calculate the transitions:

$$\begin{aligned} & \frac{\text{expected number of times transition from state } B \text{ to state } B}{\text{expected number of times transition from state } B} \\ &= \frac{2 \times 0.737 + 1 \times 0.109}{2 \times 0.737 + 2 \times 0.109 + 1 \times 0.024 + 1 \times 0.129} = 0.86 \end{aligned}$$

Since  $B$  can only transition to  $B$  or  $P$ , and we know the probability for  $B$ , we can find the probability of transitioning to  $P$  since they add to 1 ( $P \rightarrow 0.14$ ). Repeating the process for  $P$ ,

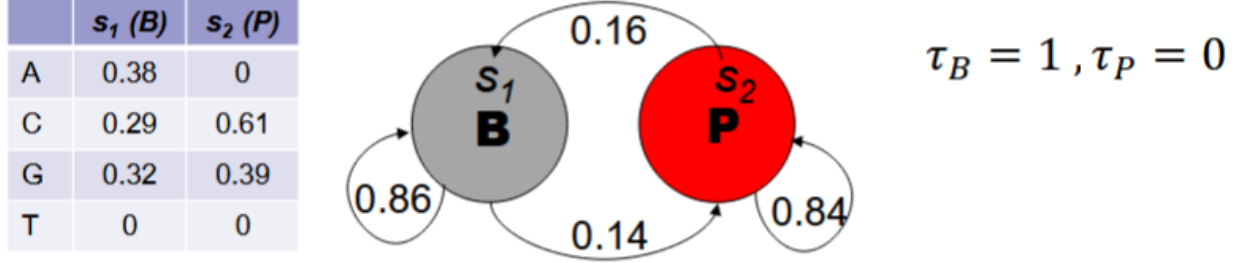
$$\begin{aligned} & \frac{\text{expected number of times transition from state } P \text{ to state } P}{\text{expected number of times transition from state } P} \\ &= \frac{1 \times 0.129}{1 \times 0.024 + 1 \times 0.129} = 0.84 \end{aligned}$$

The probability from  $P \rightarrow B$  is 0.16.

Finally, we want to calculate the starting states.

- The probability we start with  $B$  is  $0.737 + 0.109 + 0.024 + 0.129 = 1$ . (We have a small rounding error in this example.)
- The probability we start with  $P$  is  $0 + 0 + 0 + 0 = 0$ .

Our final result is:



Now, we can go back to the E-step.

## Efficient E-Step

How do we make this process efficient?

For emission parameters  $b_k(\sigma_m) = P(x_i = \sigma_m | \pi_i = s_k)$ , we need the quantities in the numerator and denominator.

$$\frac{\text{expected number of times in state } k \text{ and observe symbol } \sigma_m}{\text{expected number of times in state } k}$$

Recall from the forward-backward algorithm we saw how to efficiently compute the probability we are in state  $k$  at position  $i$  given the observation sequence, i.e.,

$$P(\pi_i = s_k | x, \lambda)$$

Expected number of times in state  $k$ :

$$\sum_{i=1}^N P(\pi_i = s_k | x_1 x_2 \dots x_n, \lambda)$$

We can find the expected number of times in state  $k$  and observe  $\sigma_m$ :

$$\sum_{i.s.t. x_i = \sigma_m} P(\pi_i = s_k | x_1 x_2 \dots x_n, \lambda)$$

Now, we need the transition parameters:

- The expected number of times transition from state  $k$ :

$$\sum_{i=1}^{n-1} P(\pi_i = s_k | x_1 x_2 \dots x_n, \lambda)$$

- The expected number of times transition from state  $k$  to state  $j$ :

$$\sum_{i=1}^{n-1} P(\pi_i = s_k, \pi_{i+1} = s_j | x_1 x_2 \dots x_n, \lambda)$$

– Note: we cannot write this as a product of two state estimates since they are not independent.



$$\begin{aligned}
& \sum_{i=1}^{n-1} P(\pi_i = s_k, \pi_{i+1} = s_j | x_1 x_2 \dots x_n, \lambda) \\
&= \sum_{i=1}^{n-1} \frac{P(x_1 x_2 \dots x_i, \pi_i = s_k, \pi_{i+1} = s_j, x_{i+1}, x_{i+2} \dots x_n | \lambda)}{P(x | \lambda)} \\
&= \sum_{i=1}^{n-1} \frac{\boxed{P(x_1 x_2 \dots x_i, \pi_i = s_k | \lambda)} \underbrace{P(\pi_{i+1} = s_j | \pi_i = s_k)}_{P(x | \lambda)} \underbrace{P(x_{i+1} | \pi_{i+1} = s_j)}_{P(x | \lambda)} \boxed{P(x_{i+2} \dots x_n | \pi_{i+1} = s_j, \lambda)}}{P(x | \lambda)} \\
&\quad \text{Forward variable } \alpha_i(k) \quad \quad \quad \text{Backward variable } \beta_{i+1}(j) \\
&\quad \quad \quad \text{Transition parameter } a_{kj} \quad \quad \text{Emission parameter } b_j(x_{i+1}) \\
&= \sum_{i=1}^{n-1} \frac{\alpha_i(k) a_{kj} b_j(x_{i+1}) \beta_{i+1}(j)}{P(x | \lambda)} \\
&= \sum_{i=1}^{n-1} \frac{\alpha_i(k) a_{kj} b_j(x_{i+1}) \beta_{i+1}(j)}{\sum_{k=1}^K \sum_{j=1}^K P(\pi_i = s_k, \pi_{i+1} = s_j, x_1 x_2 \dots x_n | \lambda)} \\
&= \sum_{i=1}^{n-1} \frac{\alpha_i(k) a_{kj} b_j(x_{i+1}) \beta_{i+1}(j)}{\sum_{k=1}^K \sum_{j=1}^K \alpha_i(k) a_{kj} b_j(x_{i+1}) \beta_{i+1}(j)}
\end{aligned}$$

### Efficient M-step

- For transition parameters maximum likelihood estimate of

$$a_{kj} = P(\pi_{i+1} = s_j | \pi_i = s_k)$$

we have the quantities in the numerator and denominator

$$\frac{\text{expected number of times transition from state } k \text{ to state } j}{\text{expected number of times transition from state } k}$$

- For emission parameters

$$b_k(\sigma_m) = P(x_i = \sigma_m | \pi_i = s_k)$$

we have the quantities in the numerator and denominator

$$\frac{\text{expected number of times in state } k \text{ and observe symbol } \sigma_m}{\text{expected number of times in state } k}$$