# Genome Annotation

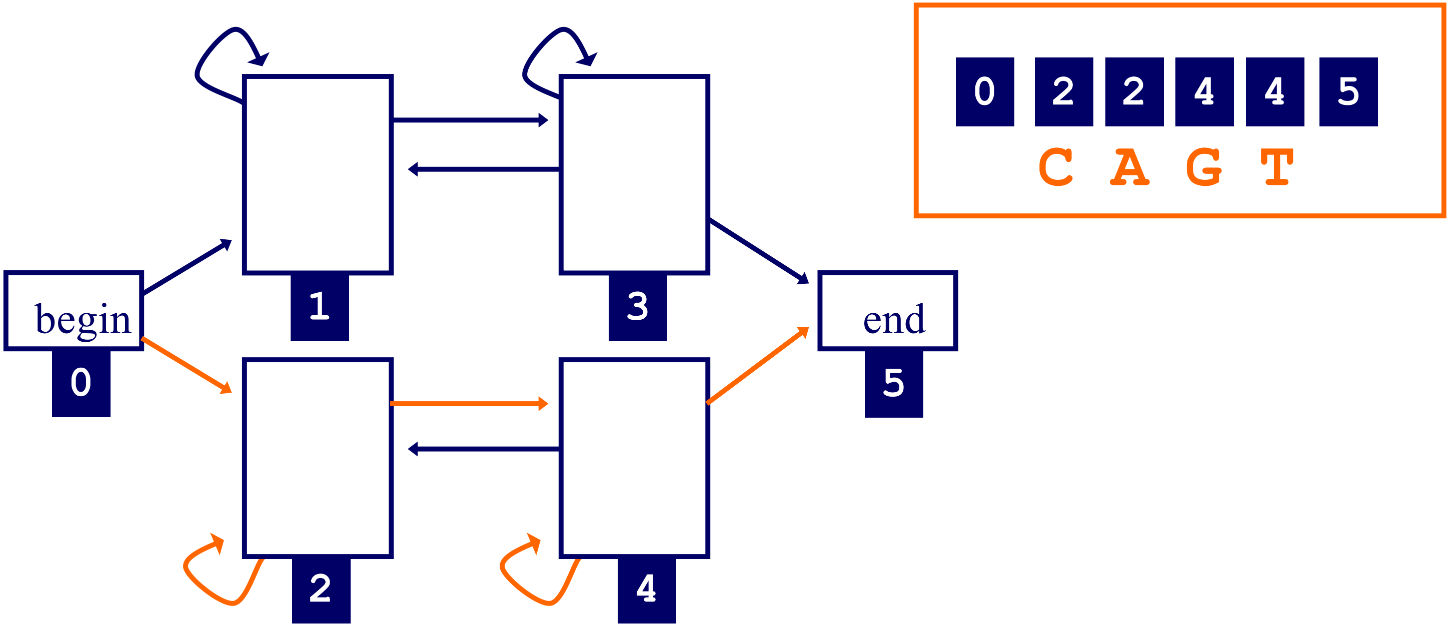
## The Baum–Welch algorithm

# Outline

- HMM parameter estimation
  - Fully observed scenario
  - Hidden states scenario
- Baum–Welch algorithm
  - Instance of Expectation-Maximization algorithm

# Fully observed case for HMM parameter estimation

- estimation is simple if we know the correct path for each sequence in our training set




- estimate parameters by counting the number of times each parameter is used across the training set

# Maximum likelihood estimates for fully observed case

# times transition from state *k* to state *l* is observed

$$\hat{a}_{kl} = \frac{n_{k \to l}}{\sum_m n_{k \to m}}$$

transition parameters

# times emission of character c from state *k* is observed

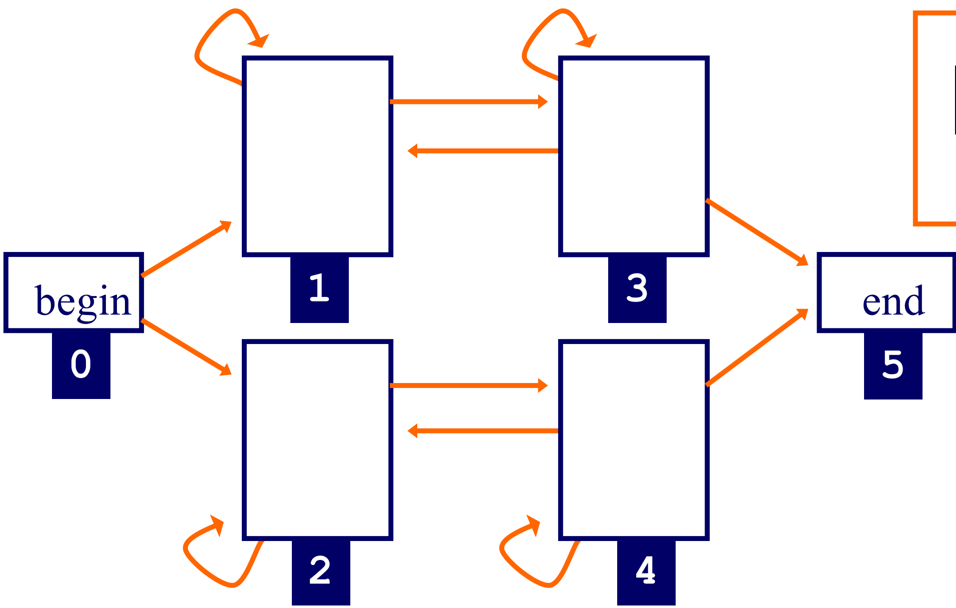
$$\hat{e}_k(c) = \frac{n_{k,c}}{\sum_{c'} n_{k,c'}}$$

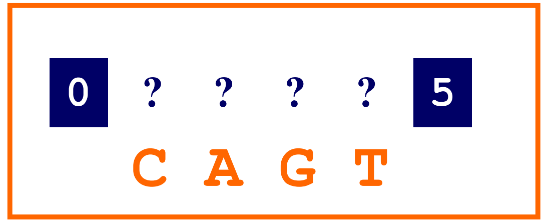
emission parameters

Can use Laplace or *m*-estimate smoothing if there is limited training data

# Estimation with Hidden State

- if we don't know the correct path for each sequence in our training set, consider all possible paths for the sequence




- estimate parameters through a procedure that counts the expected number of times each transition and emission occurs across the training set

# Estimating parameters with hidden state: The Baum-Welch Algorithm

- *a.k.a.* the Forward-Backward algorithm
- an *Expectation–Maximization* (EM) algorithm
  - EM is a family of algorithms for learning probabilistic models in problems that involve hidden state
  - generally used to find *maximum likelihood* estimates for parameters of a model
- in this context, the hidden state is the path that explains each training sequence

# Sketch of the Baum-Welch Algorithm

- initialize the parameters of the model
- iterate until convergence
  - calculate the *expected* number of times each transition or emission is used, using current parameters (Expectation step)
  - adjust the parameters to *maximize* the likelihood of these expected values (Maximization step)

# The Expectation step: Overview

- Calculate the expected # of times that
  - letter $c$ is emitted by state $k$
  - transition from $k$ to $l$ is used
- Key values in these calculations
  - Forward values $f_k(i)$
  - Backward values $b_k(i)$
  - Posterior probabilities: $P(\pi_i = k|x)$

# Expectation step for emissions

- Define a couple of random variables for emission events
  - $I_{i,j,k}$: Indicator random variable indicating whether state $k$ generated position $i$ in sequence $j$

$$I_{i,j,k} = \begin{cases} 1 & \text{if } \pi_i = k \text{ for sequence } j \\ 0 & \text{otherwise} \end{cases}$$

  - $C_{k,c}$: Random variable for number of times state $k$ generated character $c$ across all sequences

$$C_{k,c} = \sum_{j} \sum_{\{i|x_i^j = c\}} I_{i,j,k}$$

sum over sequences

sum over positions where $c$ occurs in $x^j$

# Expectation step for emissions

$$n_{k,c} = E[C_{k,c} \mid x^1 \ldots x^j] = \sum_j \sum_{\{i \mid x_i^j = c\}} E[I_{i,j,k} \mid x^j]$$

number of times state k emitted c

$$n_{k,c} = \sum_j \frac{1}{f_N^j(L)} \sum_{\{i \mid x_i^j = c\}} f_k^j(i) b_k^j(i)$$

sum over sequences

sum over positions where *c* occurs in x

# Expectation step for transitions

- and we can calculate the expected number of times that the transition from $k$ to $l$ is used

$$n_{k \to l} = \sum_{x^j} \frac{\sum_i \underbrace{f_k^j(i)}_{\text{forward}} \underbrace{a_{kl}}_{\text{transition}} \underbrace{e_l(x_{i+1}^j)}_{\text{emission}} \underbrace{b_l^j(i+1)}_{\text{backward}}}{f_N^j(L)}$$

- or if $l$ is a silent state

$$n_{k \to l} = \sum_{x^j} \frac{\sum_i f_k^j(i) a_{kl} b_l^j(i)}{f_N^j(L)}$$

# The Maximization step

- With the expected values $n_{k \to l}$ and $n_{k,c}$ computed from the Expectation step, we update the parameters of the model
- Equations are identical to the fully-observed case, but with expected values of counts instead of observed counts

**Expected** # times transition from state *k* to state *l* is observed

$$\hat{a}_{kl} = \frac{n_{k \to l}}{\sum_m n_{k \to m}}$$

transition parameters

**Expected** # times emission of character c from state *k* is observed

$$\hat{e}_k(c) = \frac{n_{k,c}}{\sum_{c'} n_{k,c'}}$$

emission parameters

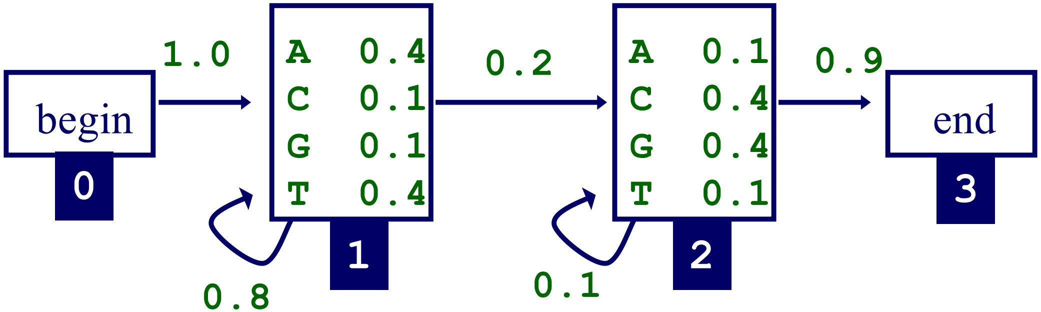
Can use Laplace or *m*-estimate smoothing if there is limited training data

# The Baum-Welch Algorithm

- initialize the parameters of the HMM
- iterate until convergence
  - initialize $n_{k,c}$ , $n_{k \to l}$ with pseudocounts
  - **E-step**: for each training set sequence $j = 1 \ldots n$
    - calculate $f_k(i)$ values for sequence $j$
    - calculate $b_k(i)$ values for sequence $j$
    - add the contribution of sequence $j$ to $n_{k,c}$ , $n_{k \to l}$
  - **M-step**: update the HMM parameters using $n_{k,c}$, $n_{k \to l}$

# Baum-Welch Algorithm Example

- given
  - the HMM with the parameters initialized as shown
  - the training sequences **TAG**, **ACG**




- We'll work through one iteration of Baum-Welch

# Baum-Welch Example (Cont)

- determining the forward values for TAG

$$f_0(0) = 1$$

$$f_1(1) = e_1(T) \times a_{01} \times f_0(0) = 0.4 \times 1 = 0.4$$

$$f_1(2) = e_1(A) \times a_{11} \times f_1(1) = 0.4 \times 0.8 \times 0.4 = 0.128$$

$$f_2(2) = e_2(A) \times a_{12} \times f_1(1) = 0.1 \times 0.2 \times 0.4 = 0.008$$

$$f_2(3) = e_2(G) \times \left(a_{12} \times f_1(2) + a_{22} \times f_2(2)\right) =$$
$$0.4 \times (0.0008 + 0.0256) = 0.01056$$

$$f_3(3) = a_{23} \times f_2(3) = 0.9 \times 0.01056 = 0.009504$$

- here we compute just the values that represent events with non-zero probability
- in a similar way, we also compute forward values for ACG

# Baum-Welch Example (Cont)

- determining the backward values for TAG

$$b_3(3) = 1$$

$$b_2(3) = a_{23} \times b_3(3) = 0.9 \times 1 = 0.9$$

$$b_2(2) = a_{22} \times e_2(G) \times b_2(3) = 0.1 \times 0.4 \times 0.9 = 0.036$$

$$b_1(2) = a_{12} \times e_2(G) \times b_2(3) = 0.2 \times 0.4 \times 0.9 = 0.072$$

$$b_1(1) = a_{11} \times e_1(A) \times b_1(2) + a_{12} \times e_2(A) \times b_2(2) =$$

$$0.8 \times 0.4 \times 0.072 + 0.2 \times 0.1 \times 0.036 = 0.02376$$

$$b_0(0) = a_{01} \times e_1(T) \times b_1(1) = 1.0 \times 0.4 \times 0.02376 = 0.009504$$

- here we compute just the values that represent events with non-zero probability
- in a similar way, we also compute backward values for ACG

# Baum-Welch Example (Cont)

- determining the expected emission counts for state 1

| | contribution of TAG | | contribution of ACG | | pseudocount |
|---|---|---|---|---|---|
| $n_{1,A} =$ | $\dfrac{f_1(2)b_1(2)}{f_3(3)}$ | $+$ | $\dfrac{f_1(1)b_1(1)}{f_3(3)}$ | $+$ | $1$ |
| $n_{1,C} =$ | | | $\dfrac{f_1(2)b_1(2)}{f_3(3)}$ | $+$ | $1$ |
| $n_{1,G} =$ | | | | | $1$ |
| $n_{1,T} =$ | $\dfrac{f_1(1)b_1(1)}{f_3(3)}$ | | | $+$ | $1$ |

*note that the forward/backward values in these two columns differ; in each column they are computed for the sequence associated with the column

# Baum-Welch Example (Cont)

- determining the expected transition counts for state 1 (not using pseudocounts)

contribution of TAG

contribution of ACG

$$n_{1\to 1} = \frac{f_1(1)a_{11}e_1(A)b_1(2)}{f_3(3)} + \frac{f_1(1)a_{11}e_1(C)b_1(2)}{f_3(3)}$$

$$n_{1\to 2} = \frac{f_1(1)a_{12}e_2(A)b_2(2) + f_1(2)a_{12}e_2(G)b_2(3)}{f_3(3)} + \frac{f_1(1)a_{12}e_2(C)b_2(2) + f_1(2)a_{12}e_2(G)b_2(3)}{f_3(3)}$$

- in a similar way, we also determine the expected emission/transition counts for state 2

# Baum-Welch Example (Cont)

- determining probabilities for state 1

$$e_1(A) = \frac{n_{1,A}}{n_{1,A} + n_{1,C} + n_{1,G} + n_{1,T}}$$

$$e_1(C) = \frac{n_{1,C}}{n_{1,A} + n_{1,C} + n_{1,G} + n_{1,T}}$$

$$\vdots$$

$$a_{11} = \frac{n_{1\to1}}{n_{1\to1} + n_{1\to2}} \qquad a_{22} = \frac{n_{2\to2}}{n_{2\to2} + n_{2\to3}}$$

$$a_{12} = \frac{n_{1\to2}}{n_{1\to1} + n_{1\to2}} \qquad a_{23} = \frac{n_{2\to3}}{n_{2\to2} + n_{2\to3}}$$

# Baum-Welch Convergence

- some convergence criteria
  - likelihood of the training sequences changes little
  - fixed number of iterations reached
  - parameters are not changing significantly
- usually converges in a small number of iterations
- will converge to a *local* maximum (in the likelihood of the data given the model)

$$\log \Pr(\text{sequences} \mid \theta) = \sum_{x^j} \log \Pr(x^j \mid \theta)$$

parameters

# Summary

- Fully observed scenario for HMMs uses simple maximum likelihood parameter equations
- When the state paths are not observed, Baum-Welch is required
- Baum-Welch
  - An Expectation-Maximization algorithm
  - Alternates between computing expected counts and maximizing parameter values
  - Converges to a local maximum