Genome Annotation

The Forward and Backward algorithms

Outline

- Two tasks, given an HMM
  - Computing the probability of a sequence
  - Computing the posterior probability of a hidden state at a given position in a sequence
- HMM algorithms
  - The Forward algorithm
  - The Backward algorithm

How Likely is a Given Sequence?

- We usually only observe the sequence, not the path
- To find the probability of a sequence, we must sum over *all* possible paths

$$\Pr(X_1 \ldots X_L) = \sum_{\pi} \Pr(X_1 \ldots X_L, \underbrace{\pi_1 \ldots \pi_L}_{\pi})$$

- but the number of paths can be exponential in the length of the sequence...
- the Forward algorithm enables us to compute this efficiently

How Likely is a Given Sequence: The Forward Algorithm

- Dynamic programming algorithm
- Analogous to Viterbi but with summation instead of maximization
- subproblem: define $f_k(i)$ to be the probability of generating the first *i* characters and ending in state *k*

$$f_k(i) = P(x_1, \ldots, x_i, \pi_i = k)$$

- we want to compute $f_N(L)$, the probability of generating the entire sequence (x) and ending in the end state (state N)
- can define this recursively

The Forward Algorithm

- initialization:

$$f_0(0) = 1$$

probability that we're in start state and have observed 0 characters from the sequence

$$f_k(0) = 0, \quad \text{for } k \text{ that are not silent states}$$

The Forward Algorithm

- recursion for emitting states (i =1…L):

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

- recursion for silent states:

$$f_l(i) = \sum_k f_k(i) a_{kl}$$

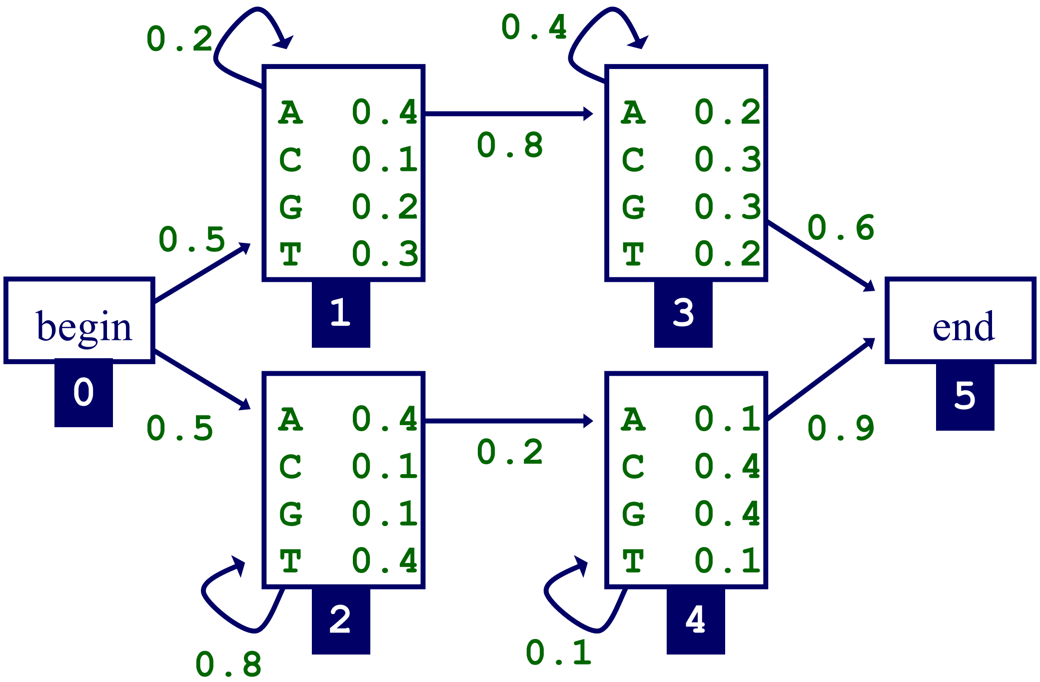
The Forward Algorithm

- termination:

$$\Pr(X) = \Pr(X_1 \ldots X_L) = f_N(L) = \sum_k f_k(L) a_{kN}$$

probability that we're in the end state and have observed the entire sequence

Forward Algorithm Example



- given the sequence x = **TAGA**

Forward Algorithm Example

- given the sequence x = TAGA
- initialization

$$f_0(0) = 1 \qquad f_1(0) = 0 \ \ldots \ f_5(0) = 0$$

- computing other values

$$f_1(1) = e_1(T) \times (f_0(0) \times a_{01} + f_1(0) a_{11}) =$$
$$0.3 \times \left(1 \times 0.5 + 0 \times 0.2\right) = 0.15$$
$$f_2(1) = 0.4 \times \left(1 \times 0.5 + 0 \times 0.8\right)$$

$$f_1(2) = e_1(A) \times (f_0(1) \times a_{01} + f_1(1) a_{11}) =$$
$$0.4 \times \left(0 \times 0.5 + 0.15 \times 0.2\right)$$

• • •

$$\Pr(TAGA) = f_5(4) = (f_3(4) \times a_{35} + f_4(4) a_{45})$$

Posterior probabilities

- It is often useful to compute the probability that the *i*th character of a sequence was produced by state *k,* given the sequence *x*

$$P(\pi_i = k|x)$$

- Uses of these probabilities:
  - Giving local predictions of the hidden states
  - Measures of uncertainty for positions in predicted paths
  - Estimating parameters of an HMM when the the training data do not have state paths (via the Baum-Welch algorithm)

Computing posterior probabilities

- the probability of of producing x with the i th symbol being produced by state k is

$$\begin{aligned} P(\pi_i = k|x) &= \frac{P(\pi_i = k, x)}{P(x)} \\ &= \frac{P(x_1, \ldots, x_i, \pi_i = k) P(x_{i+1}, \ldots, x_L | \pi_i = k)}{P(x)} \\ &= \frac{f_k(i) b_k(i)}{f_N(L)} \end{aligned}$$

- the first term in the numerator, $f_k(i)$, is computed by the forward algorithm
- the second term in the numerator, $b_k(i)$, is computed by the backward algorithm

The Backward Algorithm

- Dynamic programming algorithm
- Essentially the Forward algorithm in reverse
- **subproblem**: define $b_k(i)$ to be the probability of the suffix of *x* starting at position *i+1* given that the hidden state at position *i* was *k*.

$$b_k(i) = P(x_{i+1}, \ldots, x_L | \pi_i = k)$$

- can define this recursively

The Backward Algorithm

- initialization:

$$b_k(L) = a_{kN}$$

for states with a transition to *end* state

The Backward Algorithm

- recursion ($i = L\text{-}1 \ldots 0$):

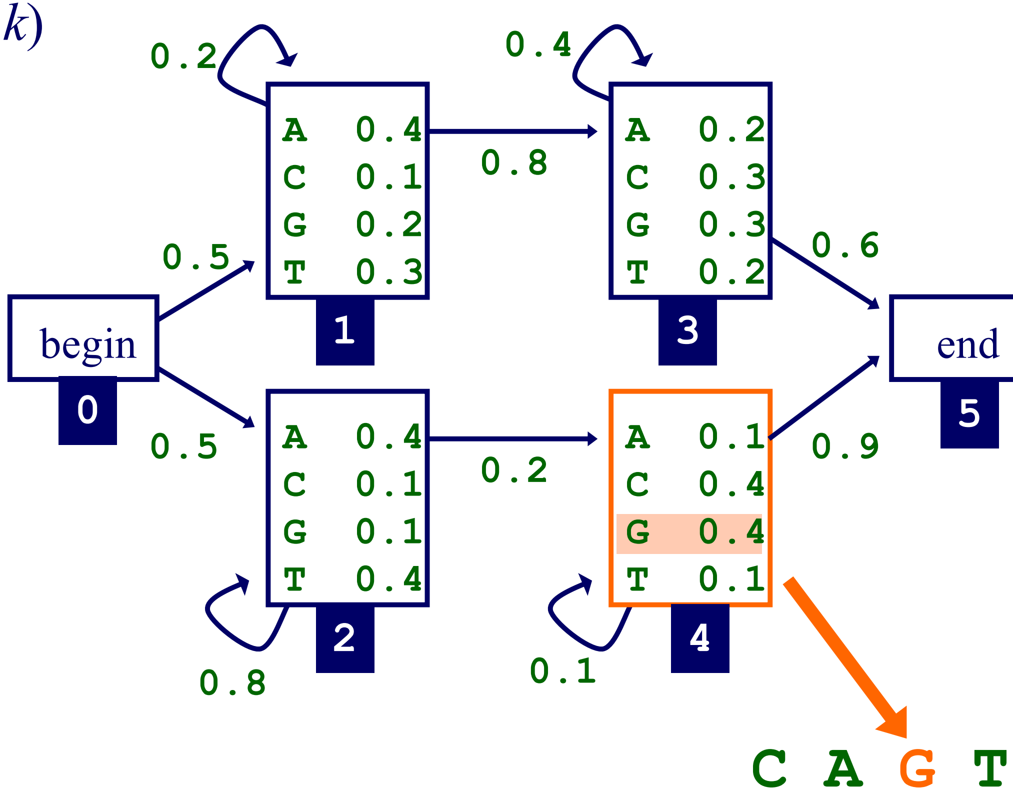
$$b_k(i) = \sum_l \begin{Bmatrix} a_{kl} b_l(i), & \text{if } l \text{ is silent state} \\ a_{kl} e_l(x_{i+1}) b_l(i+1), & \text{otherwise} \end{Bmatrix}$$

- An alternative to the forward algorithm for computing the probability of a sequence:

$$\Pr(x) = \Pr(x_1 \ldots x_L) = b_0(0)$$

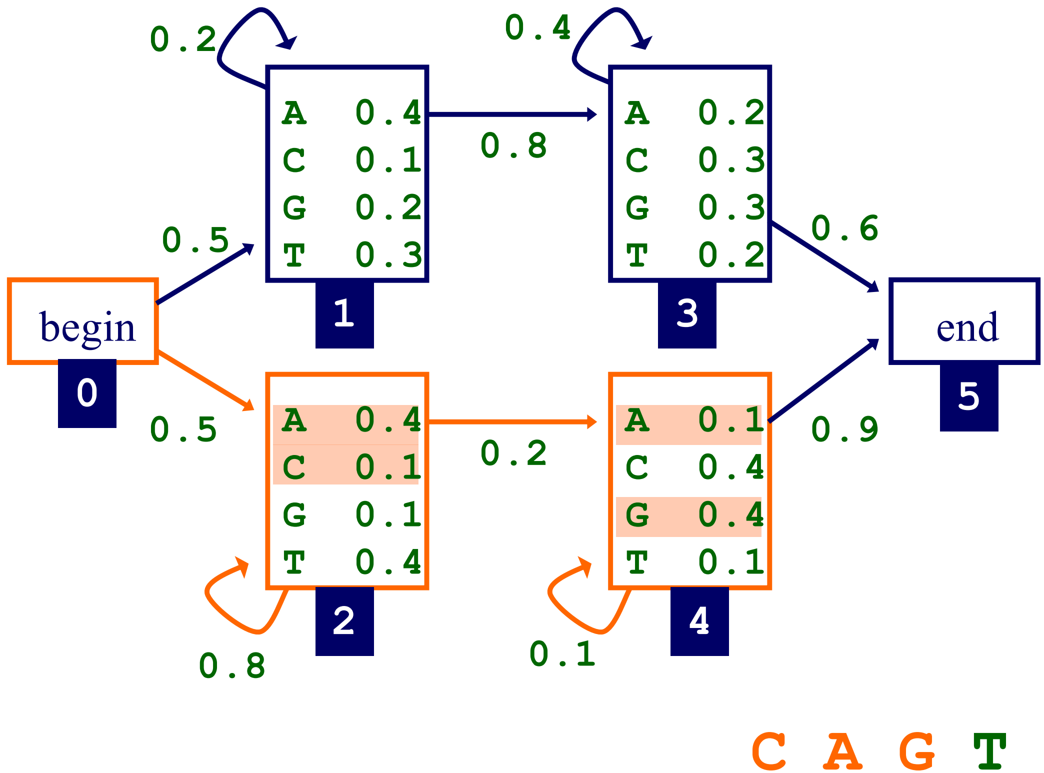
A visual example of the posterior probability calculation

- we want to know the probability of producing sequence x with the i th symbol being produced by state k (for all x, i and k)



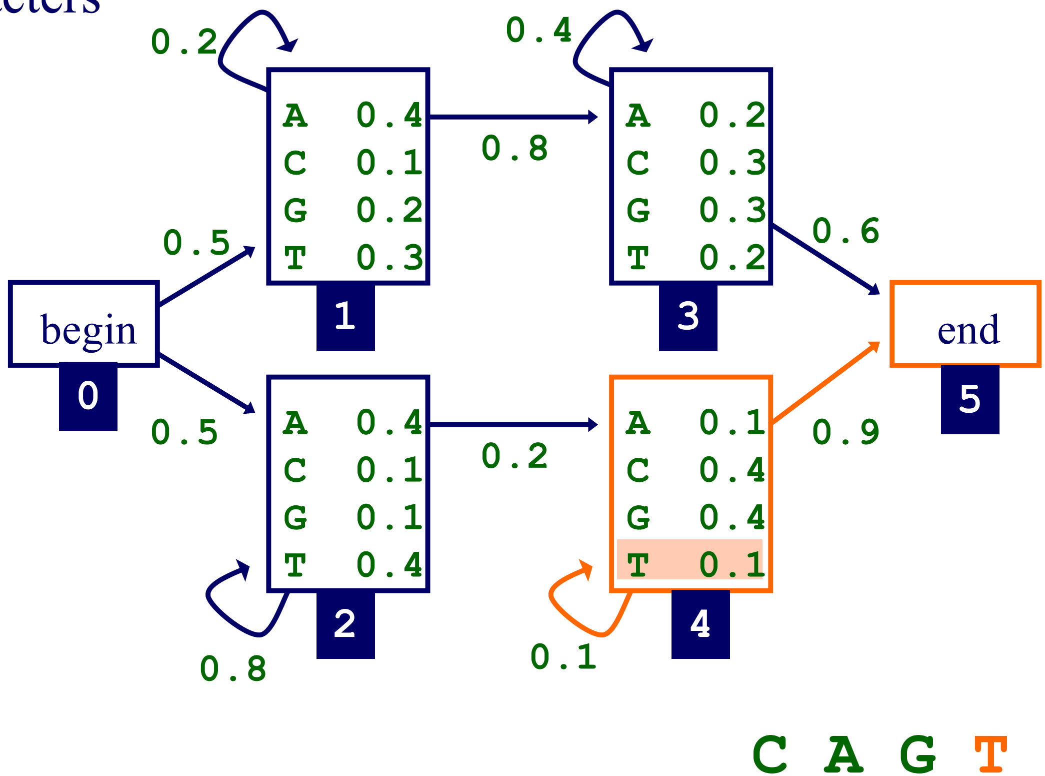
A visual example of the posterior probability calculation

- the forward algorithm gives us $f_k(i)$, the probability of being in state k having observed the first i characters of x



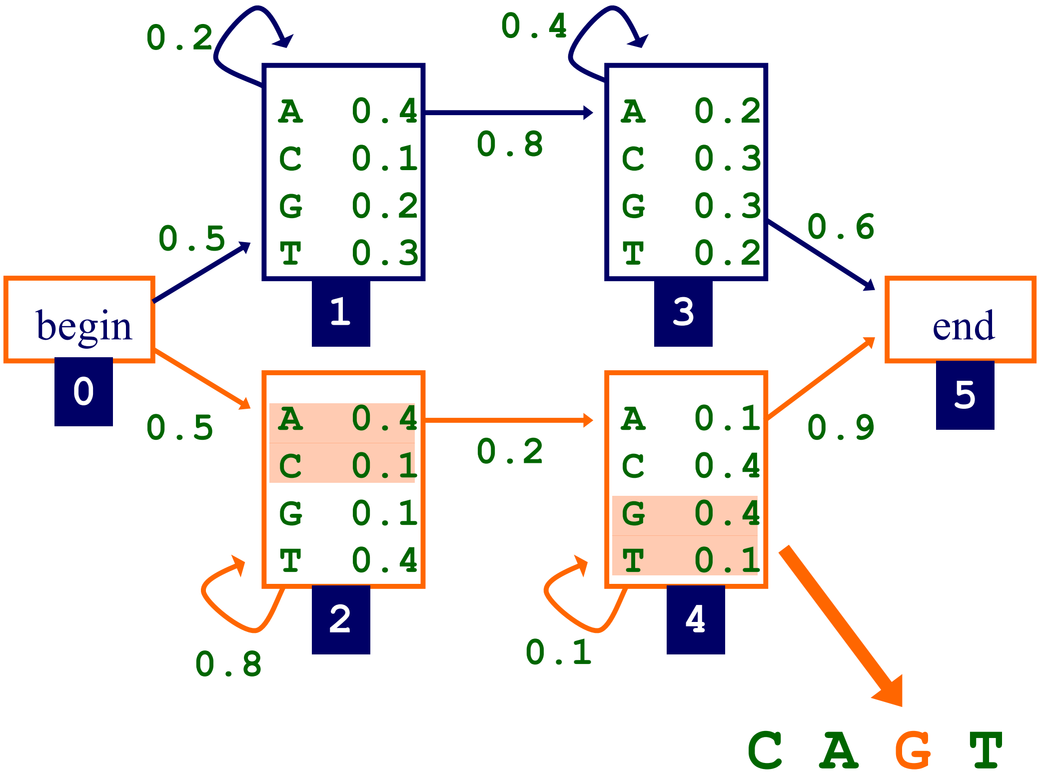
A visual example of the posterior probability calculation

- the *backward algorithm* gives us $b_k(i)$, the probability of observing the rest of x, given that we're in state *k* after *i* characters



A visual example of the posterior probability calculation

- putting forward and backward together, we can compute the probability of producing sequence x with the i th symbol being produced by state k



Posterior decoding

- An alternative to Viterbi (most probable path) decoding for HMMs
- Predict the state at each position that has the highest posterior probability
- Can differ from the state in the Viterbi path
- Posterior decoding predictions are more accurate with respect to some measures

Summary

- The Forward and Backward algorithms provide efficient solutions to the problems of
  - Computing the probability of a sequence
  - Computing the posterior probability of a particular hidden state at particular position in the sequence
- Both are dynamic programming algorithms
  - similar to Viterbi