

Hidden Markov Models

BMI/CS 776

www.biostat.wisc.edu/~craven/776.html

Mark Craven

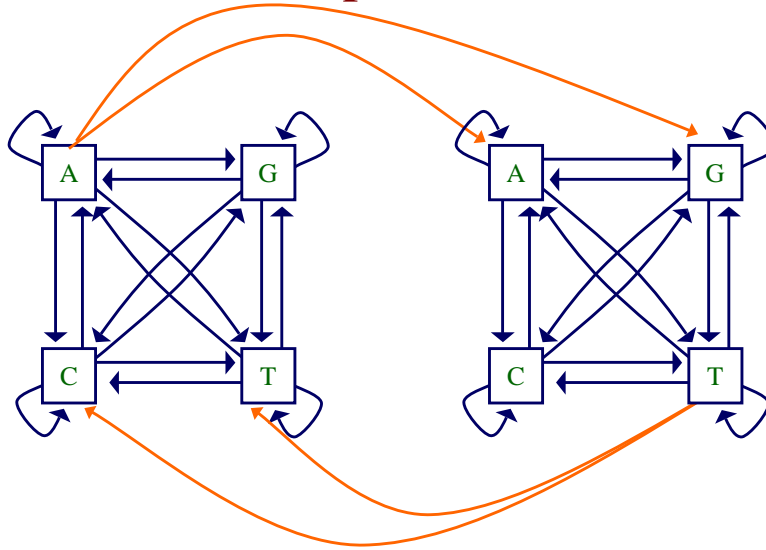
craven@biostat.wisc.edu

March 2002

Announcements

- talks this week
 - *Quantitative Genetics in the age of Genomics*
 - Bruce Walsh, Univ. of Arizona
 - 4pm 3/11, 184 Russell Labs
 - *Genomic Expression Programs in Yeast Cells Responding to Diverse Environmental Changes*
 - Audrey Gasch, Lawrence Berkeley National Lab
 - 10am 3/13, Biotech Center Auditorium
 - *Population-Genetics Models for the Evolution of New Gene Function*
 - Bruce Walsh
 - 4pm 3/13, 1221 Statistics

A Simple HMM



- given say a *T* in our input sequence, which state emitted it?

Hidden State

- we'll distinguish between the *observed* parts of a problem and the *hidden* parts
- in the Markov models we've considered previously, it is clear which state accounts for each part of the observed sequence
- in the model above, there are multiple states that could account for each part of the observed sequence
 - this is the hidden part of the problem

The Parameters of an HMM

- as in Markov chain models, we have transition probabilities

$$a_{kl} = \Pr(\pi_i = l \mid \pi_{i-1} = k)$$

probability of a transition from state k to l

π represents a path (sequence of states) through the model

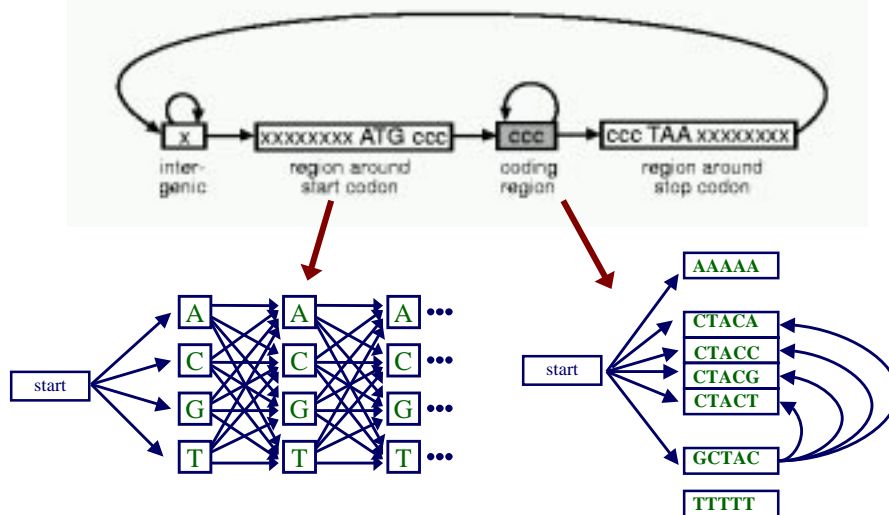
- since we've decoupled states and characters, we might also have emission probabilities

$$e_k(b) = \Pr(x_i = b \mid \pi_i = k)$$

probability of emitting character b in state k

Simple HMM for Gene Finding

Figure from A. Krogh, An Introduction to Hidden Markov Models for Biological Sequences



HMM for Eukaryotic Gene Finding

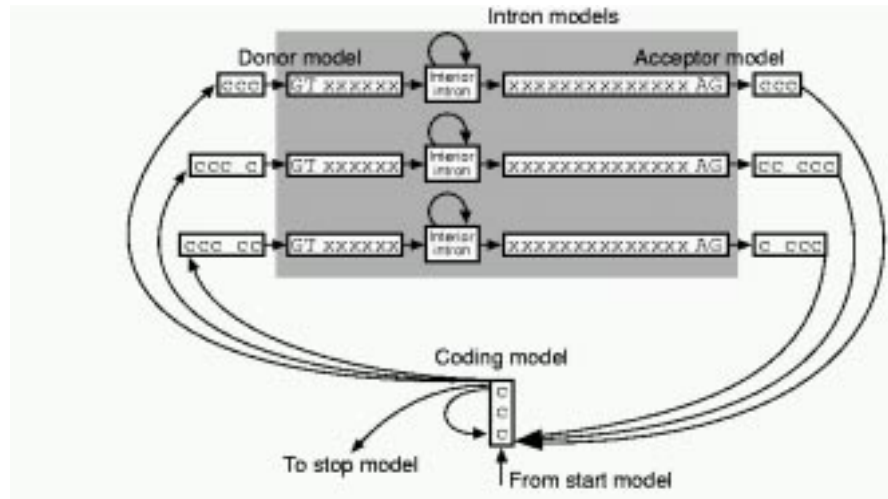


Figure from A. Krogh, An Introduction to Hidden Markov Models for Biological Sequences

Announcements

- reading for next week
 - Chapter 5 (through section 5.5) of Durbin et al.
 - Chapter 6 of Durbin et al.
- talks
 - *Population-Genetics Models for the Evolution of New Gene Function*
 - Bruce Walsh, University of Arizona
 - 4pm 3/13, 1221 Computer Sciences
 - *Information Fusion for Multidocument Summarization*
 - Regina Barzilay, Columbia University
 - 4pm 3/14, 1325 Computer Sciences
- HW #2 ready this afternoon; due 4/3

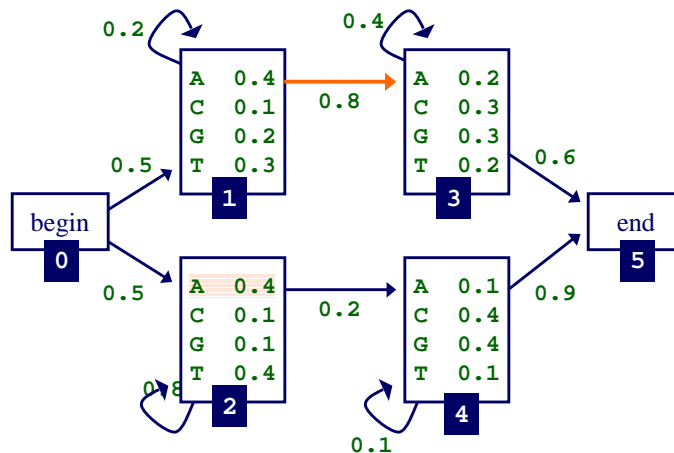
The Course Project

- implement an algorithm or two
- experiment with it on some real data
- milestones
 1. proposal (due 4/8)
 - the algorithm(s) you will be investigating
 - the data set(s) you will be using
 - 1-3 hypotheses
 2. description of your experiments (due 4/15)
 - how you will test your hypotheses
 - data to be used
 - what will be varied
 - methodology
 3. final write-up (due 5/17)
 - ~8 pages similar in format to a CS conference paper
 - prototypical organization: introduction, description of methods, description of experiments, discussion of results

A Simple HMM

a_{13} probability of a transition from state 1 to state 3

$e_2(A)$ probability of emitting character A in state 2



Three Important Questions

- How likely is a given sequence?
the Forward algorithm
- What is the most probable “path” for generating a given sequence?
the Viterbi algorithm
- How can we learn the HMM parameters given a set of sequences?
the Forward-Backward (Baum-Welch) algorithm

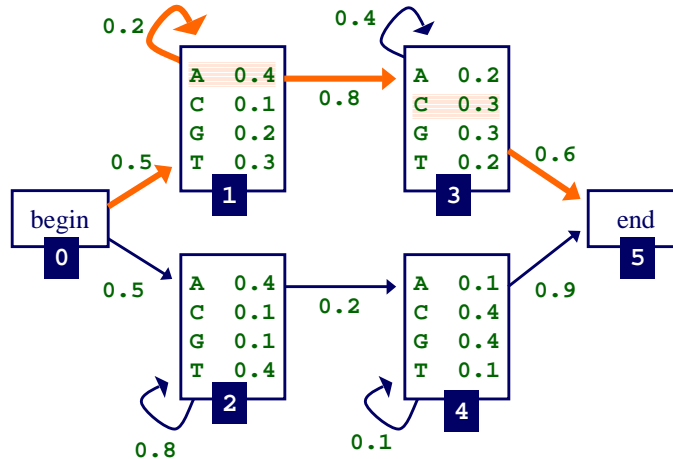
How Likely is a Given Sequence?

- the probability that the path $\pi_0 \dots \pi_N$ is taken and the sequence $x_1 \dots x_L$ is generated:

$$\Pr(x_1 \dots x_L, \pi_0 \dots \pi_N) = a_{0\pi_1} \prod_{i=1}^L e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}$$

(assuming begin/end are the only silent states on path)

How Likely Is A Given Sequence?



$$\begin{aligned} \Pr(\text{AAC}, \pi) &= a_{01} \times e_1(\text{A}) \times a_{11} \times e_1(\text{A}) \times a_{13} \times e_3(\text{C}) \times a_{35} \\ &= 0.5 \times 0.4 \times 0.2 \times 0.4 \times 0.8 \times 0.3 \times 0.6 \end{aligned}$$

How Likely is a Given Sequence?

- the probability over *all* paths is:

$$\Pr(x_1 \dots x_L) = \sum_{\pi} \Pr(x_1 \dots x_L, \underbrace{\pi_0 \dots \pi_N}_{\pi})$$

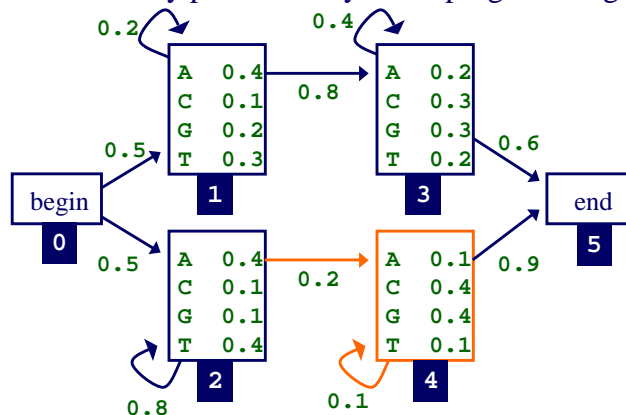
- but the number of paths can be exponential in the length of the sequence...
- the Forward algorithm enables us to compute this efficiently

How Likely is a Given Sequence: The Forward Algorithm

- define $f_k(i)$ to be the probability of being in state k having observed the first i characters of x
- we want to compute $f_N(L)$, the probability of being in the end state having observed all of x
- can define this recursively

The Forward Algorithm

- because of the Markov property, don't have to explicitly enumerate every path – use dynamic programming instead



- e.g. compute $f_4(i)$ using $f_2(i-1)$, $f_4(i-1)$

The Forward Algorithm

- initialization:

$$f_0(0) = 1$$

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

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

The Forward Algorithm

- recursion for emitting states ($i=1\dots L$):

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

- recursion for silent states:

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

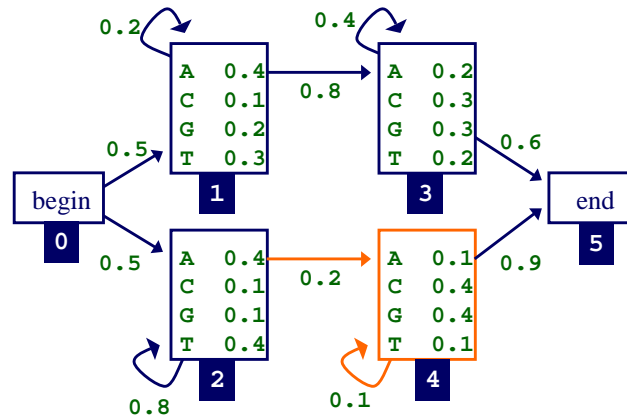
The Forward Algorithm

- termination:

$$\Pr(x) = \Pr(x_1 \dots x_L) = f_N(L) = \sum_k f_k(L) a_{kN}$$

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

Forward Algorithm Example



- given the sequence $x = \mathbf{TAGA}$

Forward Algorithm Example

- given the sequence $x = \text{TAGA}$
- initialization

$$f_0(0) = 1 \quad f_1(0) = 0 \quad f_5(0) = 0$$

- computing other values

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

$$f_2(1) = 0.4 \times (1 \times 0.5 + 0 \times 0.8)$$

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

• • •

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

Three Important Questions

- How likely is a given sequence?
- What is the most probable “path” for generating a given sequence?
- How can we learn the HMM parameters given a set of sequences?

Finding the Most Probable Path: The Viterbi Algorithm

- define $v_k(i)$ to be the probability of the most probable path accounting for the first i characters of x and ending in state k
- we want to compute $v_N(L)$, the probability of the most probable path accounting for all of the sequence and ending in the end state
- can define recursively
- can use DP to find $v_N(L)$ efficiently

Finding the Most Probable Path: The Viterbi Algorithm

- initialization:

$$v_0(0) = 1$$

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

The Viterbi Algorithm

- recursion for emitting states ($i=1 \dots L$):

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

$$\text{ptr}_l(i) = \arg \max_k [v_k(i-1)a_{kl}] \quad \text{keep track of most probable path}$$

- recursion for silent states:

$$v_l(i) = \max_k [v_k(i)a_{kl}]$$

$$\text{ptr}_l(i) = \arg \max_k [v_k(i)a_{kl}]$$

The Viterbi Algorithm

- termination:

$$\Pr(x, \pi) = \max_k (v_k(L)a_{kN})$$

$$\pi_L = \arg \max_k (v_k(L)a_{kN})$$

- traceback: follow pointers back starting at π_L

Three Important Questions

- How likely is a given sequence?
- What is the most probable “path” for generating a given sequence?
- How can we learn the HMM parameters given a set of sequences?

Announcements

- class ends at 1:45 today
- talk today: *Lattice Models, Runs-related Statistics, and Applications to Computational Biology*
Yong Kong, Curagen Corp.
2:00 PM
Biotech Center Auditorium, 425 Henry Mall
- HW #2 due date now 4/8
- project proposal due 4/8

The Learning Task

- given:
 - a model
 - a set of sequences (the training set)
- do:
 - find the most likely parameters to explain the training sequences
- the goal is find a model that *generalizes* well to sequences we haven't seen before

Learning Parameters

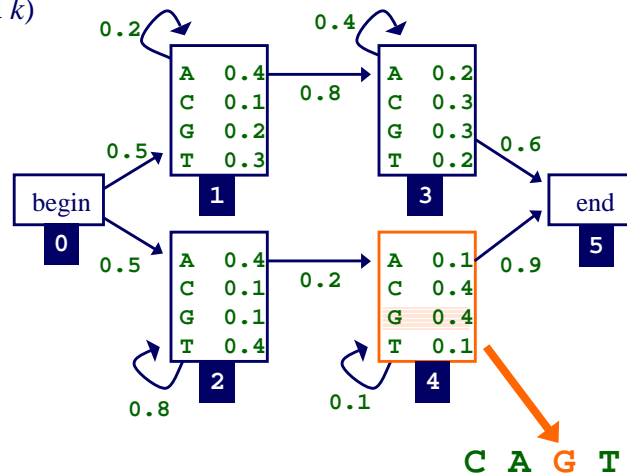
- if we know the state path for each training sequence, learning the model parameters is simple
 - no hidden state during training
 - count how often each parameter is used
 - normalize/smooth to get probabilities
 - process just like it was for Markov chain models
- if we don't know the path for each training sequence, how can we determine the counts?
 - key insight: estimate the counts by considering every path weighted by its probability

Learning Parameters: The Baum-Welch Algorithm

- *a.k.a* the Forward-Backward algorithm
- an EM approach
- algorithm sketch:
 - initialize parameters of model
 - iterate until convergence
 - calculate the *expected* number of times each transition or emission is used
 - adjust the parameters to *maximize* the likelihood of these expected values

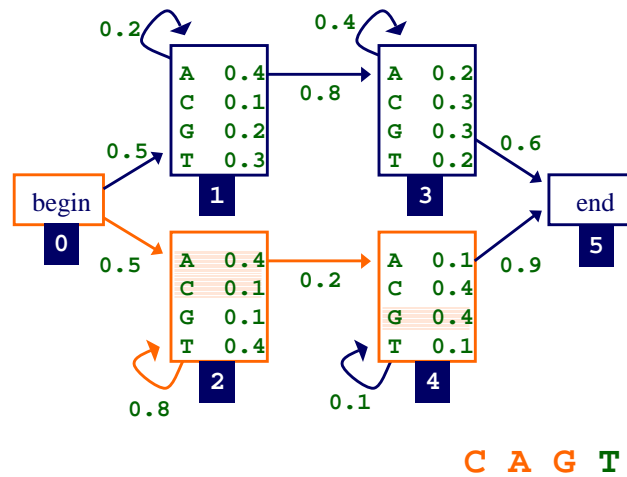
The Expectation Step

- 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)



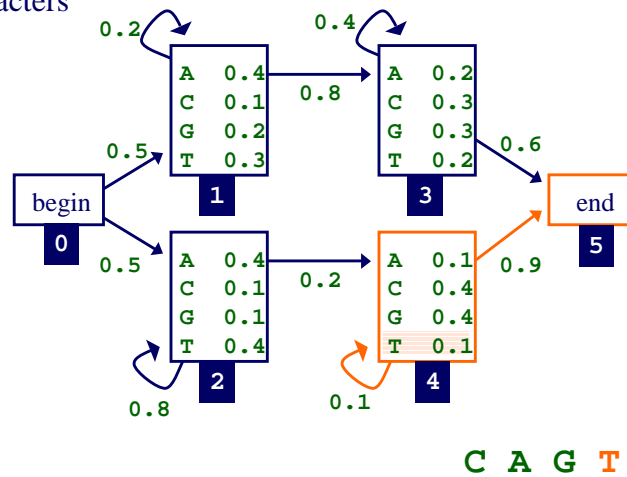
The Expectation Step

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



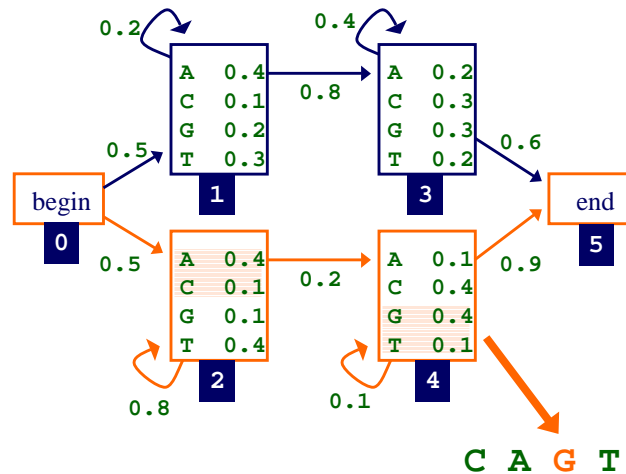
The Expectation Step

- 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



The Expectation Step

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



The Expectation Step

- first, we need to know the probability of the i th symbol being produced by state q , given sequence x

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

- given this we can compute our expected counts for state transitions, character emissions

The Expectation Step

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

$$\Pr(\pi_i = k, x) = \Pr(x_1 \dots x_i, \pi_i = k) \times \Pr(x_{i+1} \dots x_L \mid \pi_i = k)$$

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

The Backward Algorithm

- initialization:

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

for states with a transition to *end* state

The Backward Algorithm

- recursion ($i = L \dots 1$):

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

The Backward Algorithm

- termination:

$$\Pr(x) = \Pr(x_1 \dots x_L) = \sum_l \left\{ \begin{array}{ll} a_{0l} b_l(0), & \text{if } l \text{ is silent state} \\ a_{0l} e_l(x_1) b_l(1), & \text{otherwise} \end{array} \right\}$$

The Expectation Step

- now we can calculate the probability of the i th symbol being produced by state k , given x

$$\Pr(\pi_i = k | x) = \frac{\Pr(\pi_i = k, x)}{\Pr(x)}$$
$$= \frac{f_k(i)b_k(i)}{\Pr(x)}$$

The Expectation Step

- now we can calculate the expected number of times letter c is emitted by state k

$$n_{k,c} = \sum_x \frac{1}{\Pr(x)} \sum_{\{i|x_i=c\}} f_k(i)b_k(i)$$

sum over sequences

sum over positions where c occurs in x

The Expectation Step

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

$$n_{k \rightarrow l} = \sum_x \frac{\sum_i f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{\Pr(x)}$$

- or if l is a silent state

$$n_{k \rightarrow l} = \sum_x \frac{\sum_i f_k(i) a_{kl} b_l(i)}{\Pr(x)}$$

The Maximization Step

- Let $n_{k,c}$ be the expected number of emissions of c from state k for the training set
- estimate new emission parameters by:

$$e_k(c) = \frac{n_{k,c}}{\sum_{c'} n_{k,c'}}$$

- just like in the simple case
- but typically we'll do some "smoothing" (e.g. add pseudocounts)

The Maximization Step

- let $n_{k \rightarrow l}$ be the expected number of transitions from state k to state l for the training set
- estimate new transition parameters by:

$$a_{kl} = \frac{n_{k \rightarrow l}}{\sum_m n_{k \rightarrow m}}$$

The Baum-Welch Algorithm

- initialize parameters of model
- iterate until convergence
 - calculate the *expected* number of times each transition or emission is used
 - adjust the parameters to *maximize* the likelihood of these expected values

Baum-Welch Convergence

- this algorithm will converge to a *local* maximum (in the likelihood of the data given the model)
- usually in a fairly small number of iterations