Introduction to Computational Biology
Lecture # 9: Computations on HMMs

Yael Baran
11/2/08

1 Recap from Last Class

In the last lecture we represented HMMs (Hidden Markov Models) as finite state machines. We built an HMM which generates GC-rich and GC-poor DNA sequences alternately. It had two states, a GC-rich state and a GC-poor state, the first emitting Gs and Cs with higher probabilities than the second. We also had to specify the transition probabilities between the two states. The transition and emission probabilities can be summed in these two matrices:

<table>
<thead>
<tr>
<th></th>
<th>GC-rich</th>
<th>GC-poor</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>GC-rich</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GC-poor</td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>GC-rich</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GC-poor</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: The matrices specifying the HMM parameters: TR(i, j) specifies the probability of moving to state j given the machine is in state i; E(i, j) specifies the probability to emit output j given the machine is in state i.

This lecture we’ll examine another way of representing HMMs.

2 HMMs as Graphical Models

A graphical model for an HMM includes the following:

- Vertices - one for each random variable (\(\forall i, H_i, X_i\))
- Edges - connecting all pairs of vertices between which there is a direct dependency. Specifically: between each pair of subsequent state variables (\(\forall i, H_i, H_{i+1}\)), and between each state variable and its emitted variable (\(\forall i, H_i, X_i\)).

The graph representation can be sketched as in Figure 2. Note that we’ll be drawing directed graphs, since they are more intuitive here, though direct dependencies are symmetric.

Since we assumed Markovity, it follows that

\[ P(H_i|H_1\ldots H_{i-1}) = P(H_i|H_{i-1}) \]  (1)

and in general the conditional probability of a group of vertices \(V\) given another group \(Y\) is the conditional probability of \(V\) given the group \(Y' \subseteq Y\) which separates \(V\) from \(Y\) (by ‘conditional probability of a vertex’ we mean the conditional probability of the random variable it represents).
3 Computation on HMM Graphs

3.1 Definitions of Recursive Variables

As we previously saw it follows from Markovity that

$$P(H, X) = \prod_{i=1}^{n+1} P(H_{i+1} | H_i) \prod_{i=1}^{n} P(X_i | H_i)$$  \hspace{1cm} (2)

We define the forward variable as

$$F_{i+1}(H_{i+1}) = P(X_1 \ldots X_{i+1}, H_{i+1})$$  \hspace{1cm} (3)

and the recursion formula follows from the law of total probability:

$$F_{i+1}(H_{i+1}) = \sum_{H_i} F_i(H_i) P(H_{i+1} | H_i) P(X_{i+1} | H_{i+1})$$

$$= P(X_{i+1} | H_{i+1}) \sum_{H_i} F_i(H_i) P(H_{i+1} | H_i)$$  \hspace{1cm} (4)

Figure 2: The HMM graph: Hidden variables below, corresponding observed variables on top of them.

Figure 3: The forward variable: $F_i(H_i)$ equals the probability of getting the grayed portion of the observed sequence, and $H_i$ as the $i$th hidden state.

Figure 4: Recursive computation of the forward variable: Given $F_i(H_i)$ we compute $F_{i+1}(H_{i+1})$ by summing over $F_i(H_i)$ (bold circle) multiplied by the corresponding transition and emission probabilities (bold arrows).
We define the backward variable as

$$B_i(H_i) = P(X_{i+1} \ldots X_n|H_i)$$  \hspace{1cm} (5)$$

Figure 5: The backward variable: $B_i(H_i)$ equals the probability of getting the grayed portion of the observed sequence given the hidden state $H_i$ (stripped circle)

Figure 6: Recursive computation of the backward variable: Given $B_{i+1}(H_{i+1})$ we compute $B_i(H_i)$ by summing over $B_{i+1}(H_{i+1})$ (bold circle) multiplied by the corresponding transition and emission probabilities (bold arrows).

3.2 Computing Likelihood

We can now use the recursive variables to compute the likelihood of a given observed sequence:

$$P(X_1 \ldots X_n) = P(X_1 \ldots X_n|H_0 = start) = B_o(start)$$ \hspace{1cm} (7)$$

and also

$$P(X_1 \ldots X_n) = \sum_{H_n} F_n(H_n)$$ \hspace{1cm} (8)$$

3.3 Computing Inference

We can also infer the probability of a hidden state given an observed sequence:

$$P(H_i|X_1 \ldots X_n) = \frac{P(X_1 \ldots X_n, H_i)}{P(X_1 \ldots X_n)} = \frac{F_i(H_i)B_i(H_i)}{P(X_1 \ldots X_n)}$$ \hspace{1cm} (9)$$

The last question is actually a very useful one. For example, in the GC case we are given a DNA sequence, and we would like to infer which parts of the sequence are GC-rich and which are not. The equivalent question is: which parts of the sequence were generated by the GC-rich state? Given an observed sequence we can think of two ways to compute the most probable hidden sequence:
3.3.1 Maximizing for each $H_i$ separately

The first option is to calculate the most likely state for each position in the sequence, i.e. $\arg\max_{H_i} P(H_i|X)$, and return them together as the best state sequence. However, what we’re usually looking for is the most probable complete assignment of hidden states, and this is not the same. Consider the following example (We’ll use the state machine representation again):

![State machine diagram](image)

Figure 7: An example for computing most probable hidden states

Let’s say we let the machine run for 3 steps, and got the output string '00'. By simple multiplication we get that

$$
\begin{align*}
P('00' and states = AA) &= 0.0135 \\
P('00' and states = AB) &= 0.021 \\
P('00' and states = BA) &= 0.027 \\
P('00' and states = BB) &= 0.002 \\
\end{align*}
$$

and so

$$
\begin{align*}
P('00' and first state = A) &= 0.0345 \\
P('00' and first state = B) &= 0.029 \\
\end{align*}
$$

In order to get the conditional probabilities we should normalize the results, but right now all that matters is that the most probable state sequence is BA, and yet the most probable first state is A.

The reason for this seemingly abnormal result is, that one of the state sequences which begin with B is highly probable (because of the high transition probability from B to A), and the other is highly improbable (because of the low transition probability from B to B). The two sequences which begin with A are in between, and their disjunction is more probable than the most probable sequence.

3.3.2 Maximizing for all $H_i$

We now describe a dynamic algorithm which finds the most likely hidden state assignment. We define the Viterbi variable as

$$V_i(H_i) = \max_{H_1...H_{i-1}} P(H_1...H_i, X_1...X_i)$$ (10)

The recursion formula is

$$V_{i+1}(H_{i+1}) = \max_{H_1...H_i} P(H_1...H_{i+1}, X_1...X_{i+1})$$
\[ P(X_{i+1}|H_{i+1}, X_1 \ldots X_i) P(H_{i+1}, X_1 \ldots X_i) \]

\[ = \max_{H_1 \ldots H_i} P(X_{i+1}|H_1 \ldots H_{i+1}, X_1 \ldots X_i) P(H_1 \ldots H_{i+1}, X_1 \ldots X_i) \]

\[ = P(X_{i+1}|H_{i+1}) \max_{H_1 \ldots H_i} P(H_1 \ldots H_{i+1}, X_1 \ldots X_i) \]

\[ = P(X_{i+1}|H_{i+1}) \max_{H_i} \max_{H_1 \ldots H_{i-1}} P(H_{i+1}|H_1 \ldots H_i, X_1 \ldots X_i) P(H_1 \ldots H_{i+1}, X_1 \ldots X_i) \]

\[ = P(X_{i+1}|H_{i+1}) \max_{H_i} \left[ P(H_{i+1}|H_i) \max_{H_1 \ldots H_{i-1}} P(H_1 \ldots H_i, X_1 \ldots X_i) \right] \]

\[ = P(X_{i+1}|H_{i+1}) \max_{H_i} P(H_{i+1}|H_i) \]

(11)

The Viterbi variable is similar to the forward variable, only that instead of summing over all possible assignment of \( H_1 \ldots H_{n-1} \) we choose the one that gets the maximal probability.

We can now write the pseudocode for a program which finds the most probable assignment of hidden states: \((H_0\) is the special start state)

\[ \text{VITERBI()} \]

1. \( V_0(H_0) = 1, V_i(H_0) = 0 \)

2. \( V_{i+1}(H_{i+1}) = P(X_{i+1}|H_{i+1}) \max_{H_i} V_i(H_i) P(H_{i+1}|H_i) \)

3. \( \text{Ptr}_{i+1}(H_{i+1}) = \arg \max_{H_i} V_i(H_i) P(X_{i+1}|H_i) \)

At the end we get:

\[ P(X, \hat{H}) = V_{n+1}(\text{end}) \]

and backtracking is used to find the state sequence:

\[ \hat{H}_{n+1} = \text{end} \]

\[ \forall_i \hat{H}_i = \text{Ptr}_{i+1}(\hat{H}_{i+1}) \]