**Computational Biology**

**Lecture 10**

**Hidden Markov Model**

A Hidden Markov Model (HMM) is defined as:

- A set of hidden states
- An alphabet of symbols
- A set of transition probabilities
- Emission probabilities

Markov property:

\[
p(x_i = j | x_{i-1} = k, \ldots, x_0, \pi, a, b) = \frac{a_{k,j} \cdot b(x_i = j | \pi, a, b)}{\sum_j a_{k,j} \cdot b(x_i = j | \pi, a, b)}
\]

\[
p(x_i = j | x_{i-1} = k, \ldots, x_0, \pi, a, b) = \sum_j \frac{a_{k,j} \cdot b(x_i = j | \pi, a, b)}{\sum_j a_{k,j} \cdot b(x_i = j | \pi, a, b)}
\]

where \(j\) represents the state, \(a_{k,j}\) is the transition probability from state \(k\) to state \(j\), and \(b(x_i = j | \pi, a, b)\) is the emission probability of symbol \(j\) given state \(i\).

**HMM for CpG islands**

A HMM for CpG islands is represented by a graph with states and transitions between them. Each state could represent a specific nucleotide or a combination of nucleotides, and transitions between states are likely to occur based on the probability of transitioning from one nucleotide to another.
Questions with HMMs

- **Evaluation**: given $x$, what is the probability $p(x)$ that it was produced by the model?
- **Decoding**: given $x$, what is the most probable path that produces $x$ in the model?
- **Learning**: given $x$, what are the parameters (transitional probabilities and emission probabilities) of the model that maximize $p(x)$.

Viterbi decoding algorithm

- Initialization $v_0(0)=1$, $v_k(0)=0$ for $k > 0$
- Main iteration for $i = 1...n$
  - $v_i(l) = e(x), \text{max}_k (v_k(l-1), a_{i,k})$
  - $p_{tr}(l) = \text{argmax}_k (v_k(l-1), a_{i,k})$
- Termination $p(x;\pi^*) = \text{max}_k (v_n(n), a_{n,k})$

Computing $p(x)$

- Before, $p(x) = a_{x_0} \Pi_{i=2...n} a_{x_i}$
- Now, $p(x) = \Sigma_{\pi} p(x,\pi)$
- Enumerating all $\pi$ is exponential!
- Use Viterbi, same as before, but change $\text{max}_k$ to $\Sigma$
Forward evaluation algorithm

• Let $f(i) = p(x_i, \ldots, x_n, \pi_i = l)$

Then, $f(i) = e(x_i) \sum_{\text{all possible states}} f(i-1) a_{li}$

Derivation

\[
\begin{align*}
\sum_{l} f(i) &= \sum_{l} e(x_i, x_{i-1}, \ldots, x_n, \pi_i = l) \\
&= \sum_{l} e(x_i, x_{i-1}, \ldots, x_n, \pi_i = l) a_{li} \\
&= \sum_{l} e(x_i, x_{i-1}, \ldots, x_n, \pi_i = l) a_{li} f(i-1) \\
&= \sum_{l} e(x_i) a_{li} f(i-1) \\
&= e(x) \sum_{l} a_{li} f(i-1) \\
&= e(x) f(i-1) a_{li}
\end{align*}
\]

Forward evaluation algorithm

• Initialization
  $f_0(0) = 1$, $f_i(0) = 0$ for $k > 0$

• Main iteration
  for $i = 1 \ldots n$
  $f(i) = e(x_i) \sum_{l} f(i-1) a_{li}$

• Termination
  $p(x) = \sum_{l} f(n) a_{il}$
Problem of small numbers

- In Viterbi and Forward algorithm we multiply probabilities \( \rightarrow \) numbers will soon be very small and we lose precision.

- Use log space \( \rightarrow \) addition instead of multiplication.

Log space Viterbi

\[ v(i) = e(x) \max_j (v(i-1), a_{ij}) \]

Let \( V(i) = \log v(i) \)

\[ V(i) = \log [e(x) \max_j (v(i-1), a_{ij})] \]
\[ = \log e(x) + \log \max_j [v(i-1), a_{ij}] \]
\[ = \log e(x) + \max_j \log [(v(i-1), a_{ij})] \]
\[ = \log e(x) + \max_j (V(i-1) + \log a_{ij}) \]

Log space Forward

\[ f(i) = e(x) \Sigma_j (f(i-1), a_{ij}) \]

Let \( F(i) = \log f(i) \)

\[ F(i) = \log [e(x) \Sigma_j (f(i-1), a_{ij})] \]
\[ = \log e(x) + \Sigma_j [f(i-1), a_{ij}] \]
\[ = \log e(x) + \Sigma_j [f(i-1), a_{ij}] \]
\[ = \log e(x) + \Sigma_j e^{F(i) - f(i-1)} \]
\[ = \log e(x) + \Sigma_j e^{F(i) - f(i-1) + \log \alpha_i} \]
Back to the most probable path

- The Viterbi algorithm finds it!
- The most probable path might not be the most appropriate basis for judgment.
- We might want, for instance, the most probable state for an observation \( x_r \).
- More generally, we are interested in \( p(\pi_i = k \mid x) \)

Computing \( p(\pi_i = k \mid x) \)

- \( p(\pi_i = k \mid x) = p(x, \pi_i = k) / p(x) \)
- I know how to compute \( p(x) \): forward alg.

\[
\begin{align*}
p(x, \pi_i = k) &= p(x_1 \ldots x_i, \pi_i = k) \\
&= p(x_i, \pi_i = k) \\
&= f_{i,\pi_i}(k).b_{i-1}(k)
\end{align*}
\]

Backward evaluation algorithm

- Initialization
  \( b_0(n) = a_{i0} \) for all \( k \)
- Main iteration
  for \( i = n - 1 \ldots 1 \)
  \[
  b_i(i) = \sum_{\pi_i} a_{pi}\varphi(x_{i+1})b_{i+1}(i+1)
  \]
- Termination
  \[
  p(x) = \sum_{\pi_0} a_{00}\varphi(x_1)b_1(1)
  \]
Learning (training the HMM)

- Let $\theta$ be the parameters of the HMM (transition probabilities and emission probabilities, the $a$’s and $e$’s)

- Given independent sequences $x^1, \ldots, x^n$, we would like to find $\theta$ that will maximize:

  $$\log p(x^1 \ldots x^n | \theta) = \sum_{j=1}^n \log p(x^j | \theta)$$

  This is called the maximum likelihood parameters.

State sequence is known

- Assume the path for each $x^i$ is known
  - For instance, we have sequences in which CpG islands are already labeled

- Paths are known, let
  - $A_{kl}$ = number of transitions from $k$ to $l$
  - $E_k(b)$ = number of times $b$ emitted in state $k$

- The maximum likelihood parameters are given by:
  - $a_{kl} = A_{kl} / \sum_{l} A_{kl}$
  - $e_k(b) = E_k(b) / \sum_{b} E_k(b)$

Maximum likelihood from counts

- Assume we have a sequence of independent observations $x_1 \ldots x_n$ and that we count $n_i$ occurrences of outcome $i$, $i=1 \ldots k$.

- Let $\theta_i = \text{probability of } i$.

- Then $\theta^{ML} = \{\theta_i = n_i / n, i=1 \ldots k\}$ is the maximum likelihood solution for $\theta$.

- Consider any other $\theta$. We want to show that $p(x | \theta^{ML}) > p(x | \theta)$
Proof

\[
\log \frac{p(x | \theta_{ML})}{p(x | \theta)} = \log \prod \frac{\theta_{ML}}{\theta} = \sum n_i \log \frac{\theta_{ML}}{\theta_i} = n \sum \theta_{ML} \log \frac{\theta_{ML}}{\theta} > 0
\]

The last summation is the relative entropy of \( \theta_{ML} \) and \( \theta \) which is always positive and 0 iff \( \theta_{ML} = \theta \) (from information theory)

Some problems

- Maximum likelihood are vulnerable to overfitting if insufficient data.
- For instance, if a state \( k \) was never used in the set of training sequences, then
  - \( a_{k} = 0 \) for all \( i \)
  - \( \theta_{k} / \theta_{i} = 0 \) for all \( i \)
- To avoid such problem, start with pseudocounts of \( r_{i} \) for \( A_{i} \) and \( r_{j}/(b) \) for \( E_{i}/(b) \).
- Large pseudocount indicates strong prior belief about the probabilities (will require more data to modify)
- Small pseudocount just to avoid zero probability

Example

Dishonest Casino HMM

\[
\begin{align*}
    r_{FF} &= r_{FA} = r_{FO} = r_{II} = 1; & \text{[avoid zero probability]} \\
    r_{FL} &= r_{FL} = r_{FL} = r_{II} = 1; & \text{[avoid zero probability]} \\
    r_{i}(1) &= r_{i}(2) = \ldots = r_{i}(6) = 20 & \text{[strong belief that fair is fair]} \\
    r_{i}(1) &= r_{i}(2) = \ldots = r_{i}(6) = 5 & \text{[wait and see for loaded]}
\end{align*}
\]
New species comes in...

- New species with different distribution of CpG islands.
- We do not have labeled genomic sequences for the new species.
- Need to find maximum likelihood $\theta$ of HMM without knowing the paths!

Baum–Welsh algorithm

start at iteration 0 with some $\theta$, call it $\theta^0$

$L^0 \leftarrow \log \mathbb{P}(x^t | \theta^0)$

$i \leftarrow 0$

repeat

$i \leftarrow i + 1$

$A^i_j \leftarrow E[A^i_j | x^t, \theta^i]$ (expected value)

$E^i(b) \leftarrow E[E^i(b) | x^t, \theta^i]$ (expected value)

calculate $\theta$ using maximum likelihood estimators from counts $A^i_j$ and $E^i(b)$ as before.

$L^i \leftarrow \log \mathbb{P}(x^t | \theta^i)$ (new likelihood)

until $L^{i-1} - L^i < \text{threshold}$

What is the guarantee?

- Baum–Welsh algorithm is a special case of a general algorithm known as Expectation Maximization (EM)

- EM guarantees that $p(X | \theta^{i+1}) \geq p(X | \theta^i)$

- It will therefore converge to a local maximum (not necessarily the maximum)
We need to...

Compute:

\(- E [A_w | x^1 \ldots x^t, \theta]\)

\(- E [E_i(b) | x^1 \ldots x^t, \theta]\)

\[E [A_{kl} | x^1 \ldots x^n, \theta]\]

By linearity of expectation:

\[E [A_{uw} | x^1 \ldots x^t, \theta] = \sum_i E [A_{uw} | x_i, \theta] \]

By linearity of expectation, again:

\[E [A_{uw} | x_i, \theta] = \sum_i E[\# \text{ of } k \rightarrow l \text{ at } x_i | x_i, \theta] \]

\[= \sum_i p(k \rightarrow l \text{ at } x_i | x_i, \theta) \]

\[p(k \rightarrow l \text{ at } x_i | x_i, \theta) = \rho(x_i = k, x_{i+1} = l | x_i, \theta) \]

\[p(x_i = k, x_{i+1} = l | x_i, \theta) = \rho(x_i = k, x_{i+1} = l, x_i / \theta) / \rho(x_i / \theta) \]

\[= f_k(l) a_u \theta(x_{i+1} | b_{i+1} / \theta) / \rho(x_i / \theta) \]

We get:

\[E[A_u | x^1 \ldots x^t, \theta] = \sum_i \frac{1}{p(x_i / \theta)} \sum_{b_{i+1}} f_k(l) a_u \theta(x_{i+1} | b_{i+1}) b_{i+1}(i + 1) \]

\[E[E_i(b) | x^1 \ldots x^t, \theta] = \sum_i \frac{1}{p(x_i / \theta)} \sum_{b_{i+1}} f_k(l) b_{i+1}(i) \]
Viterbi training

start at iteration 0 with some \( \theta \), call it \( \theta_0 \)

\( i \leftarrow 0 \)

repeat

\( i \leftarrow i + 1 \)

\( A_i[j] \leftarrow \text{number of transitions } i \rightarrow j \text{ on the most probable paths } s^1, \ldots, s^m \)

\( E_i[b] \leftarrow \text{number of times } k \text{ emits } b \text{ on the most probable paths } s^1, \ldots, s^m \)

calculate \( \theta \) using maximum likelihood estimators from counts \( A_i \) and \( E_i \) as before.

until none of the optimal paths change

What is the guarantee

- It will converge
- It will not necessarily maximize the true likelihood \( p(x_1 \ldots x^n \mid \theta) \), but \( p(x_1 \ldots x^n \mid \theta, \theta^1, \ldots, \theta^m) \)
- Usually performs less well than Baum–Welsh
- Practical, don’t have to perform Forward and Backward algorithms, only Viterbi!
- Makes sense if we are using only Viterbi decoding