A HMM $(N, \Sigma, \pi, A, B)$ consists of the following elements:

- $N$ is a positive integer specifying the number of states in the model.
- $\Sigma$ is a set of output symbols (note: we are discussing discrete cases now).
Hidden Markov Models

Other parameters $\Theta$:

- $\pi_j$ is the probability of choosing state $j$ as an initial state.
- $a_{jk}$ is the probability of transitioning from state $j$ to state $k$.
- $b_j(o)$ is the probability of emitting symbol $o$ from state $j$.
- In total: $\Theta$ has $N + N^2 + N|\Sigma|$ parameters.

Note that we have the following constraints:

$$\sum_{j=1}^{N} \pi_j = 1.$$  
for all $j$, $\sum_{k=1}^{N} a_{jk} = 1.$  
for all $j$, $\sum_{o \in \Sigma} b_j(o) = 1.$

Hidden Markov Models

An HMM specifies a probability for each possible $(x,y)$ pair, where $x$ is a sequence of symbols drawn from $\Sigma$, and $y$ is a sequence of states drawn from the integers $1, \ldots, N$. The sequences $x$ and $y$ are restricted to have the same length.

For example, say we have an HMM with $N=2$, $\Sigma=\{a, b\}$, and with some choice of the parameters.

Take $x = <a,a,b,b>$ and $y = <1,2,2,1>$. Then in this case,

$$P(x, y \mid \Theta) = \pi_1 a_1 a_2 a_2 a_1 b_1(a) b_2(a) b_2(b) b_1(b)$$
Three Problems in Hidden Markov Models

• Question 1: Evaluation
  What is the probability of the observation sequence $O_1O_2 \ldots O_T$, $P(O_1O_2 \ldots O_T|\lambda)$?

• Question 2: Most Probable Path
  Given $O_1O_2 \ldots O_T$, what is the most probable path that I took?

• Question 3: Learning HMMs:
  Given $O_1O_2 \ldots O_T$, what is the maximum likelihood HMM that could have produced this string of observations?

Starting out with Observable Markov Models

We have state sequences.
How to train?

$A = \{a_{ij}\}$:

$$a_{ij} = \frac{C(i \rightarrow j)}{\sum_{q \in Q} C(i \rightarrow q)}$$
Training Hidden Markov Models

Say we have an HMM with \( N = 2 \), \( K = \{ e, f, g, h \} \).

The **supervised** HMM training problem

*Given paired sequences* \( \{(e/1 \ g/2), (e/1 \ h/2), (f/1 \ h/2), (f/1 \ g/2)\} \), how to choose parameter values for \( \pi, a_{ij}, \) and \( b(o) \)?

The **unsupervised** HMM training problem

*Given output sequences* \( \{(e \ g), (e \ h), (f \ h), (f \ g)\} \), how to choose parameter values for \( \pi, a_{ij}, \) and \( b(o) \)?

How?

**Maximum likelihood estimation!**

What’s the log likelihood function \( L(\Theta) \)?

### MLE for HMMs

We have two sets \( X \) and \( Y \), and a joint distribution \( P(x, y | \Theta) \)

In HMM:

- each \( x \in X \) is an output sequence \( o_1, \ldots, o_T \)
- each \( y \in Y \) is a state sequence \( q_1, \ldots, q_T \)

If we have fully observed data, \( (x, y) \) pairs, then

\[
L(\Theta) = \sum_i \log P(x_i, y_i | \Theta)
\]

If we have partially observed data, \( x_i \) examples, then

\[
L(\Theta) = \sum_i \log P(x_i | \Theta) = \sum_i \log \sum_{y \in Y} P(x_i, y | \Theta)
\]
HMM Training: Caveat

Network structure of HMM is always created by hand
– no algorithm for double-induction of optimal structure and
probabilities has been able to beat simple hand-built
structures.

Question is: How to choose parameter values for \( \pi \), \( a_{ij} \),
and \( b_i(o) \)?

Supervised HMM Training: An Example

We have an HMM with \( N = 2 \), \( K = \{ e, f, g, h \} \)
We see the following paired sequences in training data

\[
\begin{align*}
e/1 & \quad g/2 \\
e/1 & \quad h/2 \\
f/1 & \quad h/2 \\
f/1 & \quad g/2
\end{align*}
\]

Maximum likelihood estimates:
\( \pi_1 = 1.0, \pi_2 = 0.0 \)

for parameters \( a_{ij} \):

<table>
<thead>
<tr>
<th>j=1</th>
<th>j=2</th>
<th>j=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>i=1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>i=2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

for parameters \( b_i(o) \):

\[
\begin{align*}
i=1 & \quad 0.5 \quad 0.5 \quad 0 \quad 0 \\
i=2 & \quad 0 \quad 0 \quad 0.5 \quad 0.5
\end{align*}
\]

Note: state 3 is a dummy final state
Supervised HMM Training: The Likelihood Function

Notation:
Say \((x,y) = \{o_1 \ldots o_T, q_1 \ldots q_T\}\), and
\(f(i,j,x,y)\) = number of times state \(j\) follows state \(i\) in \((x,y)\)
\(f(i,x,y)\) = number of times state \(i\) is the initial state in \((x,y)\)
\(f(i,o,x,y)\) = number of times state \(i\) is paired with observation \(o\)

Then
\[
P(x, y) = \prod_{i \in \{1 \ldots N-1\}} \pi_i^{f(i,x,y)} \prod_{i \in \{1 \ldots N-1\}, j \in \{1 \ldots N\}} a_{ij}^{f(i,j,x,y)} \prod_{i \in \{1 \ldots N-1\}, o \in K} b_i(o)^{f(i,o,x,y)}
\]

Supervised HMM Training: The Likelihood Function

If we have training examples \((x_l, y_l)\) for \(l = 1 \ldots m\),
\[
L(\Theta) = \sum_{l=1}^{m} \log P(x_l, y_l)
\]
\[
= \sum_{l=1}^{m} \left( \sum_{i \in \{1 \ldots N-1\}} f(i, x_l, y_l) \log \pi_i + \sum_{i \in \{1 \ldots N-1\}, j \in \{1 \ldots N\}} f(i, j, x_l, y_l) \log a_{ij} + \sum_{i \in \{1 \ldots N-1\}, o \in K} f(i, o, x_l, y_l) \log b_i(o) \right)
\]
Maximizing the Likelihood Function

Maximizing this function gives MLEs:

\[ \pi_i = \frac{\sum_l f(i, x_l, y_l)}{\sum_l \sum_k f(k, x_l, y_l)} \]

\[ a_{ij} = \frac{\sum_l f(i, j, x_l, y_l)}{\sum_l \sum_k f(i, k, x_l, y_l)} \]  

\[ b_i(o) = \frac{\sum_l f(i, o, x_l, y_l)}{\sum_l \sum_{o' \in K} f(i, o', x_l, y_l)} \]

Just like our intuitive results

What about the Hidden State Case?

For HMM, cannot compute those counts directly from the observed sequences

Use Baum-Welch algorithm. Intuitions:

**Iteratively** estimate the counts.

Start with an estimate for \( a_{ij} \) and \( b_k \), iteratively improve the estimates

Special case of Expectation-maximization (EM)

Uses forward and backward probabilities
Recap: Forward Probabilities

Given an input sequence $x_1 \ldots x_T$:

$\alpha_t(i) = P(x_1 \ldots x_t, q_t = i \mid \Theta)$ \hspace{1cm} forward probabilities

Base case:

$\alpha_t(i) = \pi_i b_i(o_i)$ for all $i$

Recursive case:

$\alpha_t(j) = \sum_i \alpha_{t-1}(i) a_{ij} b_j(o_j)$ for all $j = 1 \ldots N$ and $t = 2 \ldots T$

Forward Procedure

Computation of $\alpha_t(j)$ by summing over all previous values $\alpha_{t-1}(i)$ for all $i$
The Backward Probability

We define the **backward probability** as follows:

\[
\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = i, \Phi)
\]

This is the probability of generating partial observations \(O_{t+1}^T\) (from time \(t+1\) to the end), given that the HMM is in state \(i\) at time \(t\) and of course given model \(\Phi\).

We compute it by induction:

- **Initialization:**
  \[
  \beta_T(i) = 1, \quad 1 \leq i \leq N
  \]

- **Induction:**
  \[
  \beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad t = T-1..1, \quad 1 \leq i \leq N
  \]

Backward Procedure

Computation of \(\beta_t(i)\) by weighted sum of all successive values \(\beta_{t+1}\)
Intuition for Re-estimation of \( a_{ij} \)

We will estimate \( \hat{a}_{ij} \) via this intuition:

\[
\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}
\]

Numerator intuition:

Assume we had some estimate of probability that a given transition \( i \rightarrow j \) was taken at time \( t \) in observation sequence.

If we knew this probability for each time \( t \), we could sum over all \( t \) to get expected value (count) for \( i \rightarrow j \).

Re-estimation of \( a_{ij} \)

Let \( \gamma_t(i, j) \) be the probability of being in state \( i \) at time \( t \) and state \( j \) at time \( t+1 \), given \( O_{1..T} \) and model \( \Phi \):

\[
\gamma_t(i, j) = P(q_t = i, q_{t+1} = j \mid O, \Phi)
\]

\[
= \frac{P(q_t = i, q_{t+1} = j, O \mid \Phi)}{P(O \mid \Phi)}
\]
Computing Numerator for $\gamma_t$

The four components of $P(q_t = i, q_{t+1} = j, O | \Phi) : \alpha, \beta, a_i, b_j$ and $b_j(o_{t+1})$

$\gamma_t(i, j) = P(q_t = i, q_{t+1} = j | O, \Phi)$

$$\gamma_t(i, j) = \frac{P(q_t = i, q_{t+1} = j, O | \Phi)}{P(O | \Phi)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_i \alpha_t(i) \beta_t(i)}$$

$P(O|\Phi)$ can use forward only; backward only; or combination of forward and backward
From $\gamma$ to $a_{ij}$

- $a_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$
- The expected number of transitions from state $i$ to state $j$ is the sum over all $t$ of $\gamma$.
- The total expected number of transitions out of state $i$ is the sum over all transitions out of state $i$.
- Final formula for reestimated $a_{ij}$:
  $$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \gamma_t(i, j)}$$

Re-estimating the Observation Likelihood $b$

- This is the probability of a given symbol $v_k$ from the observation vocabulary $V$, given a state $j$: $\hat{b}_j(v_k)$.
  $$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$
- For this we will need to know the probability of being in state $j$ at time $t$, which we will call $\xi_t(j)$ ($\xi$ for state):
  $$\xi_t(j) = P(q_t = j|O, \Phi)$$
- We compute this by including the observation sequence in the probability and then normalizing:
  $$\xi_t(j) = \frac{P(q_t = j|O, \Phi)}{P(O|\Phi)}$$
Computing $\xi$

Computation of $\xi_j(t)$, the probability of being in state $j$ at time $t$.

Reestimating the Observation Likelihood $b$

$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$

For numerator, sum $\xi_j(t)$ for all $t$ in which $o_t$ is symbol $v_k$.

$\hat{b}_j(v_k) = \frac{\sum_{t=1}^{T} s.t. O_t = v_k \xi_j(t)}{\sum_{t=1}^{T} \xi_j(t)}$
Summary of A and B

\[ \hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \gamma_t(i,j)} \]

The ratio between the expected number of transitions from state \( i \) to \( j \) and the expected number of all transitions from state \( i \).

\[ \hat{b}_j(v_k) = \frac{\sum_{t=1}^{T} \mathbb{1}_{s.t. O_t = v_k} \xi_j(t)}{\sum_{t=1}^{T} \xi_j(t)} \]

The ratio between the expected number of times the observation data emitted from state \( j \) is \( v_k \), and the expected number of times any observation is emitted from state \( j \).

Summary: Forward-Backward Algorithm

- Initialize \( \Phi = (A,B,\pi) \)
- Compute \( \alpha, \beta, \xi \)
- Estimate new \( \Phi' = (A,B,\pi) \)
- Replace \( \Phi \) with \( \Phi' \)
- If not converged, go to 2

That’s for one sequence. For multiple sequences, simply add all of them in the numerators and denominators.
Summary: Forward-Backward Algorithm

Also called Baum-Welch algorithm: learning the A and B matrices of an individual HMM
It doesn’t require training data to be labeled at the state level; all you have to know is that an HMM covers a given sequence of observations, and you can learn the optimal A and B parameters for this data by an iterative process.
Baum-Welch only guaranteed to return to local max, rather than global optimum
It is a special case of EM

MLE and EM for Unsupervised HMM Training

Known states:

\[ L(\Theta) = \sum_i \log P(x_i, y_i | \Theta) \]

Unsupervised, hidden states:
training examples \((x_i)\) for \(i = 1 \ldots m\)

\[ L(\Theta) = \sum_{i=1}^{m} \log \sum_y P(x_i, y | \Theta) \]

The expected complete log likelihood is

\[ Q(\Theta, \Theta^{-1}) = \sum_{i=1}^{m} \sum_y P(y | x_i, \Theta^{-1}) \log P(x_i, y | \Theta) \]
EM for HMM Training: A Brute-Force Approach

\[ Q(\Theta, \Theta^{(t)}) = \sum_{l=1}^{m} \sum_{y} P(y \mid x_l, \Theta^{(t)}) \left( \sum_{i \in [1..N-1]} f(i, x_l, y) \log \pi_i \right. + \] 
\[ \left. \sum_{i \in [1..N-1], \ j \in [1..N]} f(i, j, x_l, y) \log a_{ij} + \sum_{i \in [1..N-1], \ o \in K} f(i, o, x_l, y) \log b_i(o) \right) \] 
\[ = \sum_{l=1}^{m} \sum_{i \in [1..N-1]} g(i, x_l) \log \pi_i + \sum_{i \in [1..N-1], \ j \in [1..N]} g(i, j, x_l) \log a_{ij} + \sum_{i \in [1..N-1], \ o \in K} g(i, o, x_l) \log b_i(o) \]

where each \( g \) is an expected count:

\[ g(i, x_l) = \sum_{y} P(y \mid x_l, \Theta^{(t)}) f(i, x_l, y) \]
\[ g(i, j, x_l) = \sum_{y} P(y \mid x_l, \Theta^{(t)}) f(i, j, x_l, y) \]
\[ g(i, o, x_l) = \sum_{y} P(y \mid x_l, \Theta^{(t)}) f(i, o, x_l, y) \]

Maximizing this log likelihood function gives EM updates:

\[ \pi_i = \frac{\sum_{i} g(i, x_l)}{\sum_{i} \sum_{k} g(k, x_l)} \quad a_{ij} = \frac{\sum_{i} g(i, j, x_l)}{\sum_{i} \sum_{k} g(i, k, x_l)} \quad b_i(o) = \frac{\sum_{i} g(i, o, x_l)}{\sum_{i} \sum_{o \in K} g(i, o', x_l)} \]

Compare this to MLEs in supervised case:

\[ \pi_i = \frac{\sum_{i} f(i, x_l, y_l)}{\sum_{i} \sum_{k} f(k, x_l, y_l)} \quad a_{ij} = \frac{\sum_{i} f(i, j, x_l, y_l)}{\sum_{i} \sum_{k} f(i, k, x_l, y_l)} \quad b_i(o) = \frac{\sum_{i} f(i, o, x_l, y_l)}{\sum_{i} \sum_{o \in K} f(i, o', x_l, y_l)} \]
Unsupervised HMM Training: An Example

We have an HMM with $N = 3$, $K = \{e, f, g, h\}$

We see the following output sequences in training data:

- e g
- e h
- f h
- f g

How would you choose parameter values for $\pi$, $a_{ij}$, and $b_i(o)$?

Four possible state sequences for the first example (e, g):

- $e/1$ g/1  $\pi_1 a_{11} a_{13} b_1(e) b_1(g)$
- $e/1$ g/2  $\pi_1 a_{12} a_{23} b_1(e) b_2(g)$
- $e/2$ g/1  $\pi_2 a_{21} a_{13} b_2(e) b_1(g)$
- $e/2$ g/2  $\pi_2 a_{22} a_{23} b_2(e) b_2(g)$
Unsupervised HMM Training: An Example

To apply EM, we need to initialize our parameters. Say we have the following initial parameter values:

\[ \pi_1 = 0.35, \pi_2 = 0.3, \pi_3 = 0.35 \]

<table>
<thead>
<tr>
<th>( a_{ij} )</th>
<th>( j=1 )</th>
<th>( j=2 )</th>
<th>( j=3 )</th>
<th>( b_i(o) )</th>
<th>( o=e )</th>
<th>( o=f )</th>
<th>( o=g )</th>
<th>( o=h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i=1 )</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
<td>( i=1 )</td>
<td>0.2</td>
<td>0.25</td>
<td>0.3</td>
<td>0.25</td>
</tr>
<tr>
<td>( i=2 )</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
<td>( i=2 )</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

A different probability for each state sequence:

- e/1 g/1 \( \pi_1 a_{11} a_{13} b_1(e) b_1(g) = 0.0021 \)
- e/1 g/2 \( \pi_1 a_{12} a_{23} b_1(e) b_2(g) = 0.00315 \)
- e/2 g/1 \( \pi_2 a_{21} a_{13} b_2(e) b_1(g) = 0.00135 \)
- e/2 g/2 \( \pi_2 a_{22} a_{23} b_2(e) b_2(g) = 0.0009 \)

Each state sequence has a different conditional probability, e.g.:

\[
P(1 \mid e, g, \Theta) = \frac{0.0021}{0.0021 + 0.00315 + 0.00135 + 0.0009} = 0.28
\]

- e/1 g/1 \( P(1 \mid e, g, \Theta) = 0.28 \)
- e/1 g/2 \( P(2 \mid e, g, \Theta) = 0.42 \)
- e/2 g/1 \( P(2 \mid e, g, \Theta) = 0.18 \)
- e/2 g/2 \( P(2 \mid e, g, \Theta) = 0.12 \)
EM for Unsupervised HMM Training (E-Step)

Fill in hidden values for (e g), (e h), (f h), (f g)

<table>
<thead>
<tr>
<th>State 1</th>
<th>State 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>e/1 g/1</td>
<td>( P(1,1</td>
</tr>
<tr>
<td>e/1 g/2</td>
<td>( P(1,2</td>
</tr>
<tr>
<td>e/2 g/1</td>
<td>( P(2,1</td>
</tr>
<tr>
<td>e/2 g/2</td>
<td>( P(2,2</td>
</tr>
<tr>
<td>e/1 h/1</td>
<td>( P(1,1</td>
</tr>
<tr>
<td>e/1 h/2</td>
<td>( P(1,2</td>
</tr>
<tr>
<td>e/2 h/1</td>
<td>( P(2,1</td>
</tr>
<tr>
<td>e/2 h/2</td>
<td>( P(2,2</td>
</tr>
<tr>
<td>f/1 h/1</td>
<td>( P(1,1</td>
</tr>
<tr>
<td>f/1 h/2</td>
<td>( P(1,2</td>
</tr>
<tr>
<td>f/2 h/1</td>
<td>( P(2,1</td>
</tr>
<tr>
<td>f/2 h/2</td>
<td>( P(2,2</td>
</tr>
<tr>
<td>f/1 g/1</td>
<td>( P(1,1</td>
</tr>
<tr>
<td>f/1 g/2</td>
<td>( P(1,2</td>
</tr>
<tr>
<td>f/2 g/1</td>
<td>( P(2,1</td>
</tr>
<tr>
<td>f/2 g/2</td>
<td>( P(2,2</td>
</tr>
</tbody>
</table>

EM for Unsupervised HMM Training (E-Step)

Calculate the expected counts, for \( a_{ij} \pi \):

\[
\begin{align*}
\sum_l g(1,1, x_l) &= 0.28 + 0.211 + 0.181 + 0.237 = 0.910 \\
\sum_l g(1,2, x_l) &= 1.72 \\
\sum_l g(2,1, x_l) &= 0.746 \\
\sum_l g(2,2, x_l) &= 0.626 \\
\sum_l g(1,3, x_l) &= 1.656 \\
\sum_l g(2,3, x_l) &= 2.344 \\
\sum_l g(1, x_l) &= 0.28 + 0.42 + 0.211 + 0.508 + 0.181 + 0.434 + 0.237 + 0.356 = 2.628 \\
\sum_l g(2, x_l) &= 1.372 \\
\sum_l g(3, x_l) &= 0.0
\end{align*}
\]
EM for Unsupervised HMM Training (E-Step)

Calculate the expected counts, for $b_i(o)$:

$$
\sum g(l, e, x_i) = 0.28 + 0.42 + 0.211 + 0.508 = 1.4
$$

$$
\sum g(l, f, x_i) = 1.209
$$

$$
\sum g(l, g, x_i) = 0.941
$$

$$
\sum g(l, h, x_i) = 0.827
$$

$$
\sum g(2, e, x_i) = 0.6
$$

$$
\sum g(2, f, x_i) = 0.385
$$

$$
\sum g(2, g, x_i) = 1.465
$$

$$
\sum g(2, h, x_i) = 1.173
$$

EM for Unsupervised HMM Training: M-Step

Calculate the new estimates:

$$
\pi_i = \frac{\sum g(l, 1, x_i) + \sum g(l, 2, x_i) + \sum g(l, 3, x_i)}{2.628 + 1.372 + 0.91} = 0.657
$$

$$
\alpha_{11} = \frac{\sum g(l, 1, x_i) + \sum g(l, 2, x_i) + \sum g(l, 3, x_i)}{2.628 + 1.372 + 0.91} = 0.21
$$

<table>
<thead>
<tr>
<th>$a_{ij}$</th>
<th>$j=1$</th>
<th>$j=2$</th>
<th>$j=3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i=1$</td>
<td>0.212</td>
<td>0.401</td>
<td>0.387</td>
</tr>
<tr>
<td>$i=2$</td>
<td>0.201</td>
<td>0.169</td>
<td>0.631</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$b(o)$</th>
<th>$o=e$</th>
<th>$o=f$</th>
<th>$o=g$</th>
<th>$o=h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i=1$</td>
<td>0.320</td>
<td>0.276</td>
<td>0.215</td>
<td>0.189</td>
</tr>
<tr>
<td>$i=2$</td>
<td>0.166</td>
<td>0.106</td>
<td>0.404</td>
<td>0.324</td>
</tr>
</tbody>
</table>
EM for Unsupervised HMM Training

Iterate the E-step and M-step 3 times:
\[ \pi_1 = 0.9986, \pi_2 = 0.00138, \pi_3 = 0 \]

<table>
<thead>
<tr>
<th>( a_{ij} )</th>
<th>( j=1 )</th>
<th>( j=2 )</th>
<th>( j=3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i=1 )</td>
<td>0.0054</td>
<td>0.9896</td>
<td>0.00543</td>
</tr>
<tr>
<td>( i=2 )</td>
<td>0.0</td>
<td>0.0013627</td>
<td>0.9986</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( b(o) )</th>
<th>( o=e )</th>
<th>( o=f )</th>
<th>( o=g )</th>
<th>( o=h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i=1 )</td>
<td>0.497</td>
<td>0.497</td>
<td>0.00258</td>
<td>0.00272</td>
</tr>
<tr>
<td>( i=2 )</td>
<td>0.001</td>
<td>0.000189</td>
<td>0.4996</td>
<td>0.4992</td>
</tr>
</tbody>
</table>

Problem with the Brute-Force Approach

Inefficient! When computing the expected counts, we have an exponential number of terms in the summation!

\[
g(i, x_i) = \sum_y P(y \mid x_i, \Theta^{i-1}) f(i, x_i, y)
\]

\[
g(i, j, x_i) = \sum_y P(y \mid x_i, \Theta^{i-1}) f(i, j, x_i, y)
\]

\[
g(i, o, x_i) = \sum_y P(y \mid x_i, \Theta^{i-1}) f(i, o, x_i, y)
\]

Fortunately, there is a way of avoiding this brute force strategy with HMMs, using the dynamic programming algorithm: **Baum-Welch algorithm**.

Aim is to efficiently calculate these expected counts
The Forward and Backward Probabilities

Suppose we could calculate the following quantities, given an input sequence $x_1 \ldots x_T$:

- Forward probabilities:
  \[ \alpha_t(i) = P(x_1 \ldots x_t, s_t = i \mid \Theta) \]
  \[ \beta_t(i) = P(x_{t+1} \ldots x_T \mid s_t = i, \Theta) \]

- Backward probabilities:

The probability of being in state $i$ at time $t$, is

\[ p_t(i) = P(s_t = i \mid x_1 \ldots x_T, \Theta) = \frac{P(s_t = i, x_1 \ldots x_T \mid \Theta)}{P(x_1 \ldots x_T \mid \Theta)} = \frac{\alpha_t(i) \beta_t(i)}{P(x_1 \ldots x_T \mid \Theta)} \]

Also,

\[ P(x_1 \ldots x_T \mid \Theta) = \sum_i \alpha_t(i) \beta_t(i) \text{ for any } t \]

Expected Initial Counts

As before,

\[ g(i, x_1 \ldots x_T) = \text{expected number of times state } i \text{ is state 1} \]

We can calculate this as

\[ g(i, x_1 \ldots x_T) = p_1(i) \]
Expected Emission Counts

As before,
\[ g(i, o, x_1 \ldots x_T) = \text{expected number of times state } i \text{ emits the symbol } o \]

We can calculate this as
\[ g(i, o, x_1 \ldots x_T) = \sum_{x_t = o} p_t(i) \]

Expected Transition Counts

As before,
\[ g(i, j, x_1 \ldots x_T) = \text{expected number of times state } j \text{ follows state } i \]

We can calculate this as
\[ g(i, j, x_1 \ldots x_T) = \sum_{t} \gamma_t(i, j) \]
Viterbi Training

An alternative to Baum-Welch training.

The most probable paths for the training sequences are derived using Viterbi algorithm, and these are used in the re-estimation process.

The process is also iterated when the new parameter values are obtained.

Stopping criteria: None of the paths change. At this point, the parameter estimates will not change either, since they are determined completely by the paths.

Viterbi Training

Difference between Baum-Welch and Viterbi training:

Total possibility vs. best path

Baum-Welch maximizes \( \log P(x_1, x_2, x_3, \ldots, x_n | \Theta) \), while Viterbi maximizes the contribution to the likelihood \( \log P(x_1, x_2, x_3, \ldots, x_n | \Theta, p^*(x_1), \ldots, p^*(x_n)) \) from the most probable paths for all the sequences.

Viterbi training performs less well in general than Baum-Welch.
Three Problems in Hidden Markov Models

• Question 1: Evaluation
  What is the probability of the observation sequence \( O_1, O_2 \ldots O_T \), \( P(O_1, O_2 \ldots O_T | \lambda) \)?

• Question 2: Most Probable Path
  Given \( O_1, O_2 \ldots O_T \), what is the most probable path that I took?

• Question 3: Learning HMMs:
  Given \( O_1, O_2 \ldots O_T \), what is the maximum likelihood HMM that could have produced this string of observations?

Basic Operations in HMMs

For an observation sequence \( O = O_1, \ldots, O_T \), the three basic HMM operations are:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algorithm</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation: Calculating</td>
<td>Forward</td>
<td>( O(TN^2) )</td>
</tr>
<tr>
<td>( P(O_1, O_2 \ldots O_T</td>
<td>\lambda) )</td>
<td>Backward</td>
</tr>
<tr>
<td>Inference: Computing (</td>
<td>Viterbi decoding</td>
<td></td>
</tr>
<tr>
<td>( Q^* = \text{argmax} P(Q</td>
<td>O) )</td>
<td></td>
</tr>
<tr>
<td>Learning: Computing (</td>
<td>Baum-Welch (EM)</td>
<td></td>
</tr>
<tr>
<td>( \lambda^* = \text{argmax} P(O</td>
<td>\lambda) )</td>
<td></td>
</tr>
</tbody>
</table>

\( T = \# \text{ timesteps}, N = \# \text{ states} \)
Summary

• What is an HMM
• Computing (and defining) forward and backward probabilities
• The Viterbi algorithm
• To be happy with the kind of math and analysis needed for HMMs
• Training HMM: understand concepts of forward-backward algorithm