Hidden Markov Models

A hidden Markov model \((N, \Sigma, \Theta)\) consists of the following elements:

- \(N\) is a positive integer specifying the number of states in the model. Without loss of generality, we take the \(N\)-th state to be a special state, the final or stop state.

- \(\Sigma\) is a set of output symbols, for example \(\{a, b\}\)

- \(\Theta\) is a vector of parameters.
Hidden Markov Models

$\Theta$ is a vector of parameters. It contains three types of parameters:

- $\pi_j$ for $j = 1, \ldots, N$ is the probability of choosing state $j$ as an initial state.
- $a_{jk}$ for $j = 1, \ldots, (N-1)$, $k = 1, \ldots, N$, is the probability of transitioning from state $j$ to state $k$.
- $b_j(o)$ for $j = 1, \ldots, (N-1)$, and $o \in \Sigma$, is the probability of emitting symbol $o$ from state $j$.
- Thus it can be seen that $\Theta$ is a vector of $N + (N-1)N + (N-1)|\Sigma|$ parameters.

Note that we have the following constraints:

$$\sum_{j=1}^{N} \pi_j = 1.$$  
for all $j$, $\sum_{k=1}^{N} a_{jk} = 1.$

$$\sum_{o \in \Sigma} b_j(o) = 1.$$  
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 \(1, \ldots, (N-1)\). The sequences \(x\) and \(y\) are restricted to have the same length.

For example, say we have an HMM with \(N = 3\), \(\Sigma = \{a, b\}\), and with some choice of the parameters. Take \(x = <a,a,b,b>\) and \(y = <1,2,2,1>\). Then in this case,

\[
P(x, y \mid \Theta) = \pi_1 a_{12} a_{21} a_{13} b_1(a) b_2(a) b_3(b) b_1(b)
\]

Training an HMM

Say we have an HMM with \(N = 3\), \(K = \{e, f, g, h\}\).

The supervised HMM training problem

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

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 \(b_i(o)\)?

How?

Maximum likelihood estimation!

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

Say we have two sets $X$ and $Y$, and a joint distribution $P(x, y \mid \Theta)$

If we have fully observed data, $(x_i, y_i)$ pairs, then

$$L(\Theta) = \sum_i \log P(x_i, y_i \mid \Theta)$$

If we have partially observed data, $x_i$ examples, then

$$L(\Theta) = \sum_i \log P(x_i \mid \Theta)$$

$$= \sum_i \log \sum_{y \in Y} P(x_i, y \mid \Theta)$$

MLE for HMMs

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

In Hidden Markov Models:
- each $x \in X$ is an output sequence $o_1, \ldots, o_T$
- each $y \in Y$ is a state sequence $q_1, \ldots, q_T$
Supervised HMM Training: An Example

We have an HMM with $N = 3$, $K = \{e, f, g, h\}$
We see the following paired sequences in training data
- e/1 g/2
- e/1 h/2
- f/1 h/2
- f/1 g/2

Maximum likelihood estimates:
\[ \pi_1 = 1.0, \pi_2 = 0.0, \pi_3 = 0.0 \]
for parameters $a_{ij}$:  
\[
\begin{array}{c|ccc}
  j=1 & j=2 & j=3 \\
  i=1 & 0 & 1 & 0 \\
  i=2 & 0 & 0 & 1 \\
\end{array}
\]
for parameters $b(o)$:  
\[
\begin{array}{ccc|cc}
  & o=e & o=f & o=g & o=h \\
  i=1 & 0.5 & 0.5 & 0 & 0 \\
  i=2 & 0 & 0 & 0.5 & 0.5 \\
\end{array}
\]

Supervised HMM Training: The Likelihood Function

Say $(x, y) = \{o_1 \ldots o_T, q_1 \ldots q_T\}$, and
\[ f(i, j, x, y) = \text{number of times state } j \text{ follows state } i \text{ in } (x, y) \]
\[ f(i, x, y) = \text{number of times state } i \text{ is the initial state in } (x, y) \text{ (1 or 0)} \]
\[ f(i, o, x, y) = \text{number of times state } i \text{ is paired with observation } o \]

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

If we have training examples \((x_l, y_l)\) for \(l = 1 \ldots m\),

\[
L(\Theta) = \sum_{l=1}^{m} \log P(x_l, y_l)
\]

\[
= \sum_{l=1}^{m} \left( \sum_{i \in [1..N-1]} f(i, x_l, y_l) \log \pi_i + \right.
\]

\[
\sum_{i \in [1..N-1], \atop j \in [1..N]} f(i, j, x_l, y_l) \log a_{ij} + \right.
\]

\[
\sum_{i \in [1..N-1], \atop o \in K} f(i, o, x_l, y_l) \log b_i(o) \right)
\]

Maximizing the Likelihood function

Maximizing this function gives MLEs:

\[
\pi_i = \frac{\sum_{l=1}^{m} f(i, x_l, y_l)}{\sum_{l=1}^{m} \sum_{k} f(k, x_l, y_l)}
\]

\[
a_{ij} = \frac{\sum_{l=1}^{m} f(i, j, x_l, y_l)}{\sum_{l=1}^{m} \sum_{k} f(i, k, x_l, y_l)}
\]

\[
b_i(o) = \frac{\sum_{l=1}^{m} f(i, o, x_l, y_l)}{\sum_{l=1}^{m} \sum_{o' \in K} f(i, o', x_l, y_l)}
\]
Unsupervised HMM Training: The Likelihood Function

If we have training examples \((x_l)\) for \(l = 1\ldots m\)

\[
L(\Theta) = \sum_{l=1}^{m} \log \sum_{y} P(x_l, y | \Theta)
\]

The expected complete log likelihood is

\[
Q(\Theta, \Theta^{-1}) = \sum_{l=1}^{m} \sum_{y} P(y | x_l, \Theta^{-1}) \log P(x_l, y | \Theta)
\]

EM for HMM Training: A Brute-Force Approach

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

\[
= \sum_{l=1}^{m} \sum_{i \in [1...N-1]} g(i, x_l) \log \pi_i + \sum_{i, j \in [1...N-1]} g(i, j, x_l) \log a_{ij} + \sum_{i \in [1...N-1], o \in K} g(i, o, x_l) \log b_{i(o)}
\]

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

\[
g(i, x_l) = \sum_{y} P(y | x_l, \Theta^{-1}) f(i, x_l, y)
\]

\[
g(i, j, x_l) = \sum_{y} P(y | x_l, \Theta^{-1}) f(i, j, x_l, y)
\]

\[
g(i, o, x_l) = \sum_{y} P(y | x_l, \Theta^{-1}) f(i, o, x_l, y)
\]
EM for HMM Training: A Brute-Force Approach

Maximizing this log likelihood function gives EM updates:

\[
\begin{align*}
\pi_i &= \frac{\sum \sum g(i, x_i)}{\sum \sum \sum g(k, x_i)} \\
a_{ij} &= \frac{\sum g(i, j, x_i)}{\sum \sum \sum g(i, k, x_i)} \\
b_i(o) &= \frac{\sum g(i, o, x_i)}{\sum \sum \sum \sum g(i, o', k, x_i)}
\end{align*}
\]

Compare this to MLEs in supervised case:

\[
\begin{align*}
\pi_i &= \frac{\sum f(i, x_i, y_i)}{\sum f(k, x_i, y_i)} \\
a_{ij} &= \frac{\sum f(i, j, x_i, y_i)}{\sum f(i, k, x_i, y_i)} \\
b_i(o) &= \frac{\sum f(i, o, x_i, y_i)}{\sum f(i, o', x_i, y_i)}
\end{align*}
\]

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:

\[
\begin{align*}
e &\ g \\
e &\ h \\
f &\ h \\
f &\ g
\end{align*}
\]

How would you choose parameter values for \(\pi\), \(a_{ij}\), and \(b_i(o)\)?
Unsupervised HMM Training: An Example

Four possible state sequences for the first example:
- e/1 g/1
- e/1 g/2
- e/2 g/1
- e/2 g/2

Each state sequence has a different probability:
- $e/1 \ g/1$ : $\pi a_{11} a_{13} b_1(e) b_1(g)$
- $e/1 \ g/2$ : $\pi a_{12} a_{23} b_1(e) b_2(g)$
- $e/2 \ g/1$ : $\pi a_{21} a_{13} b_2(e) b_1(g)$
- $e/2 \ g/2$ : $\pi a_{22} a_{23} b_2(e) b_2(g)$

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>$a_i$</th>
<th>$j=1$</th>
<th>$j=2$</th>
<th>$j=3$</th>
<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>0.2</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>

Each state sequence has a different probability:
- $e/1 \ g/1$ : $\pi a_{11} a_{13} b_1(e) b_1(g) = 0.0021$
- $e/1 \ g/2$ : $\pi a_{12} a_{23} b_1(e) b_2(g) = 0.00315$
- $e/2 \ g/1$ : $\pi a_{21} a_{13} b_2(e) b_1(g) = 0.00135$
- $e/2 \ g/2$ : $\pi a_{22} a_{23} b_2(e) b_2(g) = 0.0009$
Unsupervised HMM Training: An Example

Each state sequence has a different probability:

\begin{align*}
e/1 & \quad g/1 \quad \pi_1 a_{11} a_{13} b_1(e) b_1(g) = 0.0021 \\
e/1 & \quad g/2 \quad \pi_1 a_{12} a_{23} b_1(e) b_2(g) = 0.00315 \\
e/2 & \quad g/1 \quad \pi_2 a_{21} a_{13} b_2(e) b_1(g) = 0.00135 \\
e/2 & \quad g/2 \quad \pi_2 a_{22} a_{23} b_2(e) b_2(g) = 0.0009
\end{align*}

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

\[
P(11 | e, g, \Theta) = \frac{0.0021}{0.0021 + 0.00315 + 0.00135 + 0.0009} = 0.28
\]

\begin{align*}
e/1 & \quad g/1 \quad P(11 | e, g, \Theta) = 0.28 \\
e/1 & \quad g/2 \quad P(12 | e, g, \Theta) = 0.42 \\
e/2 & \quad g/1 \quad P(21 | e, g, \Theta) = 0.18 \\
e/2 & \quad g/2 \quad P(22 | e, g, \Theta) = 0.12
\end{align*}

EM for Unsupervised HMM Training (E-Step)

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

\begin{align*}
e/1 & \quad g/1 \quad P(11 | e, g, \Theta) = 0.28 \\
e/1 & \quad g/2 \quad P(12 | e, g, \Theta) = 0.42 \\
e/2 & \quad g/1 \quad P(21 | e, g, \Theta) = 0.18 \\
e/2 & \quad g/2 \quad P(22 | e, g, \Theta) = 0.12 \\
e/1 & \quad h/1 \quad P(11 | e, h, \Theta) = 0.211 \\
e/1 & \quad h/2 \quad P(12 | e, h, \Theta) = 0.508 \\
e/2 & \quad h/1 \quad P(21 | e, h, \Theta) = 0.136 \\
e/2 & \quad h/2 \quad P(22 | e, h, \Theta) = 0.145 \\
f/1 & \quad h/1 \quad P(11 | f, h, \Theta) = 0.181 \\
f/1 & \quad h/2 \quad P(12 | f, h, \Theta) = 0.434 \\
f/2 & \quad h/1 \quad P(21 | f, h, \Theta) = 0.186 \\
f/2 & \quad h/2 \quad P(22 | f, h, \Theta) = 0.198 \\
f/1 & \quad g/1 \quad P(11 | f, g, \Theta) = 0.237 \\
f/1 & \quad g/2 \quad P(12 | f, g, \Theta) = 0.356 \\
f/2 & \quad g/1 \quad P(21 | f, g, \Theta) = 0.244 \\
f/2 & \quad g/2 \quad P(22 | f, g, \Theta) = 0.162
\end{align*}
EM for Unsupervised HMM Training (E-Step)

Calculate the expected counts:
\[ \sum g(1, x_j) = 0.28 + 0.42 + 0.211 + 0.508 + 0.181 + 0.434 + 0.237 + 0.356 = 2.628 \]
\[ \sum g(2, x_j) = 1.372 \]
\[ \sum g(3, x_j) = 0.0 \]
\[ \sum g(1,1, x_j) = 0.28 + 0.211 + 0.181 + 0.237 = 0.910 \]
\[ \sum g(1,2, x_j) = 1.72 \]
\[ \sum g(2,1, x_j) = 0.746 \]
\[ \sum g(2,2, x_j) = 0.626 \]
\[ \sum g(1,3, x_j) = 1.656 \]
\[ \sum g(2,3, x_j) = 2.344 \]
EM for Unsupervised HMM Training: M-Step

Calculate the new estimates:

\[
\pi_i = \frac{\sum_j g(l, x_i)}{\sum_j g(1, x_i) + \sum_j g(2, x_i) + \sum_j g(3, x_i)} = \frac{2.628}{2.628 + 1.372 + 0} = 0.657
\]

\[
a_{i1} = \frac{g(11, x_i)}{\sum_j g(11, x_i) + \sum_j g(12, x_i) + \sum_j g(13, x_i)} = \frac{0.91}{0.91 + 1.72 + 1.656} = 0.21
\]

<table>
<thead>
<tr>
<th>(a_{ij})</th>
<th>(j=1)</th>
<th>(j=2)</th>
<th>(j=3)</th>
<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>0.212</td>
<td>0.401</td>
<td>0.387</td>
<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>0.201</td>
<td>0.169</td>
<td>0.631</td>
<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>
<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>0.0054</td>
<td>0.9896</td>
<td>0.00543</td>
<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>0.0</td>
<td>0.0013627</td>
<td>0.9986</td>
<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!

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

Fortunately, there is a way of avoiding this brute force strategy with HMMs, using a dynamic programming algorithm called the **Baum-Welch algorithm**. Aim is to efficiently calculate these expected counts.

The Forward-Backward Algorithm

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

\[
\alpha_t(i) = P(x_1 \ldots x_{t-1}, s_t = i \mid \Theta) \quad \text{forward probabilities}
\]
\[
\beta_t(i) = P(x_t \ldots x_T \mid s_t = i, \Theta) \quad \text{backward probabilities}
\]

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,

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

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
\[ g(i, o, x_1 \ldots x_T) = \sum_{t, x_t = o} p_i(i) \]
The Baum-Welch Algorithm

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

- $\alpha_t(i) = P(x_1 \ldots x_{t-1}, s_t = i | \Theta)$ \hspace{1cm} forward probabilities
- $\beta_t(i) = P(x_t \ldots x_T | s_t = i, \Theta)$ \hspace{1cm} backward probabilities

The probability of being in state $i$ at time $t$, and in state $j$ at time $t+1$, is

$$p_t(i, j) = P(s_t = i, s_{t+1} = j | x_1 \ldots x_T, \Theta) = \frac{\alpha_t(i) a_{ij} b_t(o_t) \beta_{t+1}(j)}{P(x_1 \ldots x_T | \Theta)}$$

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

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 \[ g(i, j, x_1 \ldots x_T) = \sum_i p_t(i, j) \]
Recursive Definitions for Forward Probabilities

Given an input sequence $x_1 \ldots x_T$:

$$\alpha_i(i) = P(x_i \ldots x_{i-1}, s_i = i | \Theta)$$  \hspace{1cm} \text{forward probabilities}

Base case:

$$\alpha_i(i) = \pi_i \text{ for all } i$$

Recursive case:

$$\alpha_i(j) = \sum_i \alpha_i(i) a_{ij} b_i(o_t) \text{ for all } j = 1 \ldots N \text{ and } t = 2 \ldots T$$

Recursive Definitions for Backward Probabilities

Given an input sequence $x_1 \ldots x_T$:

$$\beta_j(i) = P(x_i \ldots x_T | s_i = i, \Theta)$$  \hspace{1cm} \text{backward probabilities}

Base case:

$$\beta_{T+1}(i) = 1 \text{ for } i = N$$
$$\beta_{T+1}(i) = 0 \text{ for } i \neq N$$

Recursive case:

$$\beta_i(i) = \sum_j a_{ij} b_j(o_t) \beta_{t+1}(j) \text{ for all } j = 1 \ldots N \text{ and } t = 1 \ldots T$$