Topics for today

- Hidden Markov Models
- Forward-Backward algorithm
- Viterbi algorithm
- Hidden Markov Model learning
Hidden Markov Models
Hidden Markov Models

We are going to run through a complete EM for a basic HMM to fortify our understanding of EM.

... and look at some code.
Hidden Markov Models

A family of models on two sets of variables \((h_1, \ldots, h_L, x_1 \ldots x_L)\) that honor a particular set of conditional independencies.

\[
\begin{align*}
    h_i & \perp h_j | h_{i-1}, h_{i+1} & (j \notin \{i-1, i, i+1\}) \\
    h_i & \perp x_j | h_{i-1}, h_{i+1} & (j \notin \{i-1, i, i+1\}) \\
    x_i & \perp h_j | h_i & (j \neq i) \\
    x_i & \perp x_j | h_i & (j \neq i)
\end{align*}
\]

![Diagram of Hidden Markov Models](image-url)
Infer what was said, given a sequence of acoustic features.

- **x** acoustic features
- **h** word identity, for example ‘soup’, ‘tomato’, ‘baseball’
- transition probability $p(h_{i+1}|h_i)$ reflect language model, for example
  
  \[ p(h_{i+1} = \text{baseball}|h_i = \text{tomato}) < p(h_{i+1} = \text{soup}|h_i = \text{tomato}) \]
- emission probability $p(x_i|h_i)$, particular acoustic feature given the current word

For an overview of HMM applications in speech recognition

http://www.nowpublishers.com/article/Details/SIG-004
Infer activity, given a sequence of accelerometer measurements from a phone.

- **x** noisy accelerometer measurements
- **h** type of activity, walking, driving, jumping, sitting, etc.
- transition probability $p(h_{i+1}|h_i)$, for example going straight to running from driving is unlikely
- emission probability $p(x_i|h_i)$, distributions of acceleration while driving, running and walking are different
HMM - conditional probabilities

\[ p(h_1 = m | \pi) = \pi_m \]
\[ p(h_i | h_{i-1}, T) = T(h_i | h_{i-1}) \]
\[ p(x_i | h_i, \nu) = g(x_i ; \nu_{h_i}) \]
Abusing notation

\[ h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow \cdots \rightarrow h_{L-1} \rightarrow h_L \]

\[ \begin{array}{c}
    x_1 \\
    \downarrow \\
    x_2 \\
    \downarrow \\
    x_3 \\
    \downarrow \\
    x_{L-1} \\
    \downarrow \\
    x_L \\
\end{array} \]

This is the correct joint probability

\[
p(x, h) = p(h_1)p(x_1|h_1) \prod_{j=2}^{L} p(h_j|h_{j-1})p(x_j|h_j).
\]

For the sake of compactness, we will write

\[
p(x, h) = \prod_{j=1}^{L} p(h_j|h_{j-1})p(x_j|h_j)
\]

where we assume that \( p(h_1|h_0) = p(h_1) \).
An example of a data instance generated from an HMM

\[ p(h_1) = \pi = [0.5 \ 0.5] \]

\[ p(h_i | h_{i-1}) = T = \begin{bmatrix} 0.975 & 0.025 \\ 0.025 & 0.975 \end{bmatrix} \]

\[ p(x_i | h_i = 1) = f_1(x_i) = \mathcal{N}(-1, 0.25) \]

\[ p(x_i | h_i = 2) = f_2(x_i) = \mathcal{N}(1, 1) \]
An example of a data instance generated from an HMM

\[ p(h_1) = \pi = [0.5 \ 0.5 \ 0] \]

\[ p(h_i|h_{i-1}) = T = \begin{bmatrix} 0.95 & 0.05 & 0 \\ 0 & 0 & 1 \\ 0.1 & 0.9 & 0 \end{bmatrix} \]

\[ p(x_i|h_i = 1) = f_1(x_i) = [0.25 \ 0.25 \ 0.25 \ 0.25] \]

\[ p(x_i|h_i = 2) = f_2(x_i) = [0 \ 0 \ 1 \ 0] \]

\[ p(x_i|h_i = 3) = f_3(x_i) = [0 \ 0 \ 0 \ 1] \]
Inference problems in HMMs

- Inferring marginal distribution of a particular state

\[ p(h_i|x) \propto p(h_i, x) \]

- Inferring marginal distribution a pair of states

\[ p(h_{i+1}, h_i|x) \propto p(h_{i+1}, h_i, x) \]

- Inferring most likely assignment for a single hidden variables

\[ \arg\max_{h_i} p(h_i|x) = \arg\max_{h_i} p(h_i, x) \]

- Inferring most likely assignment for all hidden variables

\[ \arg\max_{h} p(h|x) = \arg\max_{h} p(h, x) \]
Naive approach to inference in HMMs

To compute a marginal distribution $p(h_i|x)$ marginalize all variables except $h_i$

$$p(h_i, x) = \sum_{h_1} \sum_{h_2} \cdots \sum_{h_{i-1}} \sum_{h_{i+1}} \cdots \sum_{h_L} \prod_{j} p(h_j|h_{j-1})p(x_j|h_j)$$

Assume each hidden variable can assume one of two states? What is the complexity of this computation? How many for loops?
Can we do better?

\[ p(h_i, x) = \sum_{h_1} \sum_{h_2} \cdots \sum_{h_{i-1}} \sum_{h_{i+1}} \cdots \sum_{h_L} p(h_1)p(x_1|h_1) \prod_{j=2}^{L} p(h_j|h_{j-1})p(x_j|h_j) \]

\[ = \sum_{h_{i-1}} \sum_{h_{i-2}} \cdots \sum_{h_1} \sum_{h_{i+1}} \cdots \sum_{h_L} p(h_1)p(x_1|h_1) \prod_{j=2}^{L} p(h_j|h_{j-1})p(x_j|h_j) \]

\[ = \sum_{h_{i-1}} \sum_{h_{i-2}} \sum_{h_1} \sum_{h_{i+1}} \cdots \sum_{h_L} p(h_1)p(x_1|h_1) \sum_{h_{i+1}} \cdots \sum_{h_L} \prod_{j=2}^{L} p(h_j|h_{j-1})p(x_j|h_j) \]

The first equality due to the fact reordering nested sum operators does not change the total sum.
The second equality due to the distributive property of multiplication over sums.
Can we do better?

Pushing terms as far left as they would go.

\[ p(h_i, x) = \sum_{h_{i-1}} \sum_{h_{i-2}} \cdots \sum_{h_1} p(h_1)p(x_1|h_1) \sum_{h_{i+1}} \cdots \sum_{h_L} \prod_{j=2}^{L} p(h_j|h_{j-1})p(x_j|h_j) \]

\[ = \sum_{h_{i-1}} p(x_{i-1}|h_{i-1}) \sum_{h_{i-2}} p(x_{i-2}|h_{i-2})p(h_{i-1}|h_{i-2}) \cdots \sum_{h_1} p(h_2|h_1)p(h_1)p(x_1|h_1) \]

\[ \times p(h_i|h_{i-1})p(x_i|h_i) \]

\[ \times \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1}) \cdots \sum_{h_L} p(h_L|h_{L-1})p(x_L|h_L) \]

We used distributive property of multiplication over sum to move terms around.
Can we do better?

\[ p(h_i, x) = \sum_{h_{i-1}} p(x_{i-1} | h_{i-1}) \sum_{h_{i-2}} p(x_{i-2} | h_{i-2}) p(h_{i-2} | h_{i-3}) \cdots \sum_{h_1} p(h_2 | h_1) p(h_1) p(x_1 | h_1) \]

\[ \times p(h_i | h_{i-1}) p(x_i | h_i) \]

\[ \sum_{h_{i+1}} p(h_{i+1} | h_i) p(x_{i+1} | h_{i+1}) \cdots \sum_{h_L} p(h_L | h_{L-1}) p(x_L | h_L) \]
Spot the recursion

\[ p(h_i, x) = \sum_{h_i-1} p(x_{i-1}|h_{i-1}) \sum_{h_{i-2}} \cdots \sum_{h_2} p(h_3|h_2) p(x_2|h_2) \sum_{h_1} p(h_2|h_1) p(h_1)p(x_1|h_1) \]

\[ \times p(h_i|h_{i-1})p(x_i|h_i) \]

\[ \times \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1}) \cdots \sum_{h_L} p(h_L|h_{L-1})p(x_L|h_L) \]

Note that

\[ \alpha(h_2) = p(x_2|h_2) \sum_{h_1} p(h_2|h_1)\alpha(h_1) \]

and more generally

\[ \alpha(h_i) = p(x_i|h_i) \sum_{h_{i-1}} p(h_i|h_{i-1})\alpha(h_{i-1}) \]
Computing and storing alpha tables

Start with $\alpha(h_1)$ and use the recursion forward

$$\alpha(h_1) = p(h_1)p(x_1|h_1)$$
$$\alpha(h_i) = p(x_i|h_i) \sum_{h_{i-1}} p(h_i|h_{i-1}) \alpha(h_{i-1})$$

Assuming that each $h_i$ ranges of $K$ possible states, $\alpha$ table is of size $K \times L$

<table>
<thead>
<tr>
<th>$\alpha(h_1 = 1)$</th>
<th>$\alpha(h_2 = 1)$</th>
<th>...</th>
<th>$\alpha(h_L = 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha(h_1 = 2)$</td>
<td>$\alpha(h_2 = 2)$</td>
<td>...</td>
<td>$\alpha(h_L = 2)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$\alpha(h_1 = K)$</td>
<td>$\alpha(h_2 = K)$</td>
<td>...</td>
<td>$\alpha(h_L = K)$</td>
</tr>
</tbody>
</table>
Interpreting alpha tables

\[
\alpha(h_1) = p(h_1)p(x_1|h_1) = p(x_1, h_1)
\]

\[
\alpha(h_2) = p(x_2|h_2) \sum_{h_1} p(h_2|h_1)\alpha(h_1) = p(x_2|h_2) \sum_{h_1} p(h_2|h_1)p(x_1, h_1)
\]

\[
= p(x_2|h_2) \sum_{h_1} p(h_2, x_1, h_1) = p(x_1, x_2, h_2)
\]

\[
\alpha(h_i) = p(x_1, \ldots, x_i, h_i)
\]
Some useful equalities

Probability of a part of sequence prefix $x_1, \ldots, x_i$

$$p(x_1, \ldots, x_i) = \sum_{h_i} \alpha(h_i)$$

Probability of the full sequence

$$p(x_1, \ldots, x_L) = \sum_{h_L} \alpha(h_L)$$

Probability of hidden state at index $i$ given prefix

$$p(h_i|x_1, \ldots, x_i) = \frac{\alpha(h_i)}{\sum_{h_i} \alpha(h_i)}$$
Spot the recursion

\[ p(h_i, x) = \sum_{h_{i-1}} p(x_{i-1}|h_{i-1}) \sum_{h_{i-2}} \cdots \sum_{h_2} p(h_3|h_2) p(h_2|h_1)p(x_2|h_2) \sum_{h_1} p(h_1)p(x_1|h_1) \]

\[ \times p(h_i|h_{i-1})p(x_i|h_i) \]

\[ \times \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1}) \cdots \sum_{h_L} p(h_L|h_{L-1})p(x_L|h_L) \]

\[ \times \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1})\beta(h_{i+1}) \]

\[ \beta(h_i) = \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1})\beta(h_{i+1}) \]
Computing and storing beta tables

Start with $\beta(h_L)$ and use the recursion **backward**

$$
\beta(h_L) = 1, \text{ for all values of } h_L \\
\beta(h_i) = \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_i)\beta(h_{i+1})
$$

Assuming that each $h_i$ ranges of K possible states, $\beta$ table is of size $K \times L$

<table>
<thead>
<tr>
<th>$\beta(h_1 = 1)$</th>
<th>$\beta(h_2 = 1)$</th>
<th>...</th>
<th>$\beta(h_L = 1) = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta(h_1 = 2)$</td>
<td>$\beta(h_2 = 2)$</td>
<td>...</td>
<td>$\beta(h_L = 2) = 1$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$\beta(h_1 = K)$</td>
<td>$\beta(h_2 = K)$</td>
<td>...</td>
<td>$\beta(h_L = K) = 1$</td>
</tr>
</tbody>
</table>
Interpreting beta tables

\[ \beta(h_{L-1}) = \sum_{h_L} p(h_L|h_{L-1}) p(x_L|h_L) \beta(h_L) = p(x_L|h_{L-1}) \]

\[ \beta(h_i) = p(x_{i+1}, \ldots, x_L|h_i) \]

Betas store probability of sequence suffix given a latent variable.

\[ \beta(h_i) = p(x_{i+1}, \ldots, x_L|h_i) \]

Alphas store probability of a sequence prefix and a latent variable

\[ \alpha(h_i) = p(x_1, \ldots, x_i, h_i) \]
Some useful equalities

\[ p(x_1, \ldots, x_L, h_i) = \alpha(h_i)\beta(h_i) \]

Probability of hidden state at index \( i \) given full sequence

\[ p(h_i|x_1, \ldots, x_L) = \frac{\alpha(h_i)\beta(h_i)}{\sum_{h_i} \alpha(h_i)\beta(h_i)} \]

Probability of the full sequence

\[ p(x_1, \ldots, X_L) = \sum_{h_1} \alpha(h_1)\beta(h_1) = \sum_{h_i} \alpha(h_i)\beta(h_i) = \sum_{h_L} \alpha(h_L)\beta(h_L) \]
Forward-backward algorithm

\[ \alpha(h_1) = p(h_1)p(x_1|h_1) = p(x_1, h_1) \]
\[ \alpha(h_i) = p(x_i|h_i) \sum_{h_{i-1}} \alpha(h_{i-1})p(h_i|h_{i-1}) \]
\[ \beta(h_L) = 1 \]
\[ \beta(h_i) = \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1})\beta(h_{i+1}) \]

Marginals

\[ p(h_i, x) = \alpha(h_i)\beta(h_i) \]
\[ p(h_i, h_{i+1}, x) = \alpha(h_i)p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1})\beta(h_{i+1}) \]
Next we will use the same approach as we did in deriving forward-backward to obtain another algorithm – Viterbi algorithm.

The key property we used in our derivation – simplification really – was distributive law.
Distributive law

We used distributive law to move from:

\[ p(h_i, x) = \sum_{h_1} \sum_{h_2} \cdots \sum_{h_{i-1}} \sum_{h_{i+1}} \cdots \sum_{h_L} \prod_j p(h_j|h_{j-1})p(x_j|h_j) \]

to

\[ p(h_i, x) = \sum_{h_{i-1}} p(x_{i-1}|h_{i-1}) \sum_{h_{i-2}} \cdots \sum_{h_2} \sum_{h_3} p(h_3|h_2)p(h_2|h_1)p(x_2|h_2) \sum_{h_1} p(h_1)p(x_1|h_1) \]

\[ \times p(h_i|h_{i-1})p(x_i|h_i) \]

\[ \times \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1}) \cdots \sum_{h_L} p(h_L|h_{L-1})p(x_L|h_L) \]

\[ \times \beta(h_{L-1}) \]

\[ \beta(h_i) \]

\[ \alpha(h_{i-1}) \]

\[ \alpha(h_2) \]

\[ \alpha(h_1) \]
Distributive law

Application of distributive law gave rise to a recursion. Resulting algorithm is one of the \textit{sum-product} algorithms. max is another operation that distributes over addition

\[ \max_{x,y} f(x)g(y) = \max_x f(x) \max_y g(y) \]

This will lead to \textit{max-product} algorithms
A different inference task

Suppose we wanted to solve

$$\max_h p(x, h) = \max_h \prod_j p(h_j|h_{j-1}) p(x_j|h_j)$$

$$\max h p(h, x) = \max_{h_1} \max_{h_2} \ldots \max_{h_L} \prod_j p(h_j|h_{j-1}) p(x_j|h_j)$$

Note: there are no sums, because $p(x, h)$ is the probability of the full configuration – nothing to marginalize.
Distributing product over sum

\[ p(h, x) = \max_{h_1} \max_{h_2} \ldots \max_{h_i} \max_{h_{i+1}} \max_{h_{i+2}} \ldots \max_{h_L} p(h_1) p(x_1|h_1) \prod_{j=2}^{L} p(h_j|h_{j-1}) p(x_j|h_j) \]

\[ = \max_{h_i} \max_{h_{i-1}} \max_{h_1} \max_{h_{i+1}} \max_{h_L} p(h_1) p(x_1|h_1) \prod_{j=2}^{L} p(h_j|h_{j-1}) p(x_j|h_j) \]

\[ = \max_{h_i} \max_{h_{i-1}} \max_{h_1} \max_{h_{i+1}} \prod_{j=2}^{L} p(h_j|h_{j-1}) p(x_j|h_j) \]
Max-product algorithm

\[
\max_h p(x, h) = \max_{h_i} p(x_i | h_i) \max_{h_{i-1}} \ldots \max_{h_3} p(h_3 | h_2) p(h_2 | h_1) p(x_2 | h_2) \max_{h_1} p(h_1) p(x_1 | h_1) \\
\times \max_{h_{i+1}} p(h_{i+1} | h_i) p(x_{i+1} | h_{i+1}) \ldots \max_{h_L} p(h_L | h_{L-1}) p(x_L | h_L)
\]
Max-product recursion

\[
\alpha(h_1) = p(h_1)p(x_1|h_1) = p(x_1, h_1)
\]

\[
\alpha(h_i) = p(x_i|h_i) \max_{h_{i-1}} \alpha(h_{i-1})p(h_i|h_{i-1})
\]

\[
\beta(h_L) = 1
\]

\[
\beta(h_i) = \max_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}|h_{i+1})\beta(h_{i+1})
\]

The most likely sequence is obtained by

\[
h_i^* = \arg\max_{h_i} \alpha(h_i)\beta(h_i)
\]
Viterbi algorithm

Viterbi algorithm is a specific example of max-product:

**Forward pass:**

\[
\alpha(h_1) = p(h_1)p(x_1|h_1) = p(x_1, h_1)
\]
\[
\alpha(h_i) = p(x_i|h_i) \max_{h_{i-1}} \alpha(h_{i-1}) p(h_i|h_{i-1})
\]
\[
\text{ptr}(h_i) = \arg\max_{h_{i-1}} \alpha(h_{i-1}) p(h_i|h_{i-1})
\]

**Backward pass:**

\[
h_L^* = \arg\max_{h_L} \alpha(h_L)
\]
\[
h_i^* = \text{ptr}(h_{i+1})
\]
Typical inference tasks in HMMs

Typical tasks:

\[ p(h_i|x) \] Marginal posterior distribution of single latent variable

\[ p(h_i, h_{i-1}|x) \] Marginal posterior distribution of a latent variable pair

\[ \arg\max_{h_i} p(h_i|x) \] Most-likely marginal assignment (Posterior decoding)

\[ \arg\max_h p(h|x) \] Most-likely joint assignment (Viterbi decoding)

Posterior decoding and Viterbi decoding are not guaranteed to yield the same solutions.
Max prod solution $\neq$ max of sum product

Let $\mathbf{x}^*$ be the solution computed by max product algorithm

$$
\mathbf{x}^* = \arg\max_{\mathbf{v}} p(\mathbf{x} = \mathbf{v})
$$

and hence the $\mathbf{x}^*$ is an assignment\(^1\) that achieves maximum probability.

Let $\hat{\mathbf{x}}$ be defined as

$$
\hat{x}_k = \arg\max_{x_k} p(x_k) = \arg\max_{x_k} \sum_{x_{1}, x_{2}, \ldots, x_{k-1}, x_{k+1}, \ldots} p(\mathbf{x})
$$

so each variable in $\hat{\mathbf{x}}$ is set to the maximum of its marginal.

By definition

$$
p(\mathbf{x}^*) \geq p(\hat{\mathbf{x}})
$$

\(^1\)there can be more than one
Suboptimality of max of marginals

<table>
<thead>
<tr>
<th>$p(x_1, x_2)$</th>
<th>$x_1 = 0$</th>
<th>$x_1 = 1$</th>
<th>$p(x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2 = 0$</td>
<td>0.35</td>
<td>0.3</td>
<td>0.65</td>
</tr>
<tr>
<td>$x_2 = 1$</td>
<td>0.05</td>
<td>0.3</td>
<td>0.35</td>
</tr>
<tr>
<td>$p(x_1)$</td>
<td>0.4</td>
<td>0.6</td>
<td></td>
</tr>
</tbody>
</table>

Most likely state is $(x_1 = 0, x_2 = 0)$ with prob 0.35.

If we went by the maximum of marginals we would obtain $(x_1 = 1, x_2 = 0)$ and that state has prob $0.3 < 0.35$.

Max product finds the most likely assignment.

Sum product computes the marginals, don’t mix and match.
When do I use which algorithm then?

Suppose ground truth latent sequence is \( z \) and your reconstruction is \( h \).

Use **Posterior decoding** \((h_i = \arg \max p(h_i, x))\) if you are penalized for each mismatch in the latent variables
\[
\sum_i [z_i \neq h_i].
\]

Use **Viterbi decoding** \((h = \arg \max_h p(h, x))\) if you are penalized equally for *any* mismatch
\[
1 - \prod_i [z_i = h_i].
\]
Assignments to labels for two different models on the same data instance

Color is based on MAP assignments of class labels (ground truth parameters were used).

MoG gets 8.2% misclassification error and HMM on MoG 0.3%. 

MAP from HMM on MoG

MAP from Mixture of Gaussians

\[ h_l = 1 \]
\[ h_l = 2 \]
Learning parameters of HMM

We will use exact EM:

\[ E : q^{\text{new}} = \arg \max_q \sum_t \sum_{h^t} q(h^t) \log p(x^t, h^t | \theta) - \sum_h q(h^t) \log q(h^t) \]

\[ M : \theta^{\text{new}} = \arg \max_\theta \sum_t \sum_{h^t} q^{\text{new}}(h^t) \log p(x^t, h^t | \theta) \]
Clarifications – expectations and marginals

Suppose we need to compute an expectation of some function $f$ with respect to some distribution $q(h_1, \ldots, h_L)$.

Let us assume that $f$ works on $h_A$ where $A \subset \{1, \ldots, L\}$

I want to compute

$$E_q[f(h_A)] = \sum_{h} q(h)f(h_A)$$

and my claim is that

$$E_q[f(h_A)] = \sum_{h} q(h)f(h_A) = \sum_{h_A} q(h_A)f(h_A)$$

and hence I only need marginal $q(h_A)$ to compute this expectation.
Computing expectations

My goal: \( \sum_h q(h)f(h_A) = \sum_{h_A} q(h_A)f(h_A) \)

Introduce \( B \) such that \( A \cup B = \{1, \ldots, L\} \) and \( A \cap B = \emptyset \).

\[
E_q[f(h_A)] = \sum_h q(h)f(h_A)
\]
\[
= \sum_{h_A} \sum_{h_B} q(h_A, h_B)f(h_A)
\]
\[
= \sum_{h_A} \sum_{h_B} q(h_B|h_A)q(h_A)f(h_A)
\]
\[
= \sum_{h_A} q(h_A)f(h_A) \sum_{h_B} q(h_B|h_A)
\]
\[
= \sum_{h_A} q(h_A)f(h_A) \sum_{h_B} q(h_B|h_A) = 1
\]
\[
= \sum_{h_A} q(h_A)f(h_A)
\]
Which parameters are we learning

\[
p(h_1 = m|\pi) = \pi_m \\
p(h_i|h_{i-1}, T) = T(h_i, |h_{i-1}) \\
p(x_i|h_i, \nu) = g(x_i; \nu_{h_i})
\]

and in the case of MoG \( \nu_k = (\mu_k, \Sigma_k) \) mean and covariance matrix of the \( k^{th} \) class.

So the M-step

\[
M : \theta^{\text{new}} = \arg\max_{\theta} \sum_t \sum_{h^t} q^{\text{new}}(h^t) \log p(x^t, h^t|\theta)
\]

operates on \( \theta = \{\pi, T, \nu_1, \ldots, \nu_K\} \).
M-step derivation

\[ T^{new} = \arg\max_T \sum_t \sum_{h^t} q(h^t) \log \left\{ p(h^t_1)p(x^t_1|h^t_1) \prod_{l=2}^L T(h^t_l|h^t_{l-1})p(x^t_l|h^t_l) \right\} \]

Let us simplify the expression under argmax

\[ \arg\max_T \sum_t \sum_{h^t} q(h^t) \log \left\{ p(h^t_1)p(x^t_1|h^t_1) \prod_{l=2}^L T(h^t_l|h^t_{l-1})p(x^t_l|h^t_l) \right\} = \]

\[ \arg\max_T \sum_t \sum_{h^t} q(h^t) \left( \log \left\{ \prod_{l=2}^L T(h^t_l|h^t_{l-1}) \right\} + \log \left\{ p(h^t_1)p(x^t_1|h^t_1) \prod_{l=2}^L p(x^t_l|h^t_l) \right\} \right) \]

\[ \text{no occurrence of } T \]

\[ \arg\max_T \sum_t \sum_{l=2}^{L} \sum_{h^t} q(h^t) \log \left\{ T(h^t_l|h^t_{l-1}) \right\} = \]

\[ \arg\max_T \sum_t \sum_{l=2}^{L} \sum_{h^t, h^t_{l-1}} q(h^t_l, h^t_{l-1}) \log \left\{ T(h^t_l|h^t_{l-1}) \right\} \]
M-step derivation

Again as with mixing proportions we are learning a matrix of multinomials

\[ \sum_a T(a|b) = 1 \]

So we need to solve a constrained optimization problem

\[
\text{maximize } \sum_{T} \sum_{t} \sum_{l=2}^{L} \sum_{h_t, h_{t-1}} q(h_t, h_{t-1}) \log \left\{ T(h_t|h_{t-1}) \right\}
\]

subject to \( \sum_a T(a|b) = 1, \forall b \)

and the Lagrangian for this problem is

\[
L(T, \lambda) = \sum_{T} \sum_{t} \sum_{l=2}^{L} \sum_{h_t, h_{t-1}} q(h_t, h_{t-1}) \log \left\{ T(h_t|h_{t-1}) \right\} + \sum_b \lambda_b \left( \sum_a T(a|b) - 1 \right)
\]
M-step derivation

The following first order conditions have to hold for an optimum

\[
\frac{\partial L(T, \lambda)}{\partial T(a|b)} L(T^*, \lambda^*) = 0
\]

\[
\frac{\partial L(T, \lambda)}{\partial \lambda_b} L(T^*, \lambda^*) = 0
\]

and more explicitly

\[
\sum_t \sum_{l=2}^L q(h_t^t = a, h_{t-1}^t = b) \frac{1}{T(a|b)} + \lambda_b = 0
\]

\[
\sum_a T(a|b) - 1 = 0
\]

this last part you can push through yourselves to get

\[
T^{\text{new}}(a|b) = \frac{\sum_t \sum_{l=2}^L q(h_t^t = a, h_{t-1}^t = b)}{\sum_t \sum_{l=2}^L q(h_{t-1}^t = b)}
\]
M-step derivation

Similar gymnastics lead to

$$\pi^\text{new}_m = \frac{\sum_t q(h^t_1 = m)}{\sum_t \sum h^t_1 q(h^t_1)} = \frac{\sum_t q(h^t_1 = m)}{N}$$
M-step derivation

Which marginals of $q(h^t)$ do we need for the update of $\nu$

$$\nu^{new} = \operatorname{argmax}_\nu \sum_t \sum_{h^t} q(h^t) \ log \left\{ p(h^t_1) p(x^t_1|h^t_1) \prod_{l=2}^L T(h^t_l|h^t_{l-1}) p(x^t_l|h^t_l) \right\}$$

and we can (and you should!) push through simplification of the update to obtain

$$\nu^{new} = \operatorname{argmax}_\nu \sum_t \sum_l \sum_{h^t_l} q(h^t_l) \ log \left\{ p(x^t_l|h^t_l) \right\}$$

$$= \operatorname{argmax}_\nu \sum_t \sum_l \sum_{h^t_l} q(h^t_l) \ log \left\{ g(x^t_l|\nu h^t_l) \right\}$$

Equating the derivative of the expression under argmax with respect to $\nu$ and to zero yields the updates.
M-step derivation

In the case of the gaussian distribution specified by $\mu_k$ and $\Sigma_k$

$$\mu_k^{\text{new}} = \frac{\sum_t \sum_{i=1}^L q(h^t_i = k)x^t_i}{\sum_t \sum_{i=1}^L q(h^t_i = k)}$$

$$\Sigma_k^{\text{new}} = \frac{\sum_t \sum_{i=1}^L q(h^t_i = k)x^t_i(x^t_i)'}{\sum_t \sum_{i=1}^L q(h^t_i = k)} - \mu_k^{\text{new}}(\mu_k^{\text{new}})'$$
Recall that E-step is

\[ E : q^{\text{new}} = \arg\max_q \sum_t \sum_{h^t} q(h^t) \log p(x^t, h^t|\theta) - \sum_h q(h^t) \log q(h^t) \]

alternatively

\[ \arg\min_q \text{KL}(q(h^t)||p(x^t, h^t|\theta)) \]

so

\[ q(h^t) \propto p(x^t, h^t|\theta) \]

Marginals
EM for HMM: E step

1: for \( t = 1 \) to \( N \), iterating over samples do
2: \( \alpha(h_1) = p(h_1)p(x_1^t|h_1) \)
3: for \( i = 2 \) to \( L \) do
4: \( \alpha(h_i) = p(x_i^t|h_i) \sum_{h_{i-1}} \alpha(h_{i-1})p(h_i|h_{i-1}) \)
5: end for
6: \( \beta(h_L) = 1 \)
7: for \( i = L - 1 \) to \( 1 \) do
8: \( \beta(h_i) = \sum_{h_{i+1}} p(h_{i+1}|h_i)p(x_{i+1}^t|h_{i+1})\beta(h_{i+1}) \)
9: end for
10: for \( i = 1 \) to \( L \) do
11: \( q_t(h_i) = \frac{\alpha(h_i)\beta(h_i)}{\sum_{h_i} \alpha(h_i)\beta(h_i)} \)
12: end for
13: for \( i = 1 \) to \( L - 1 \) do
14: \( q_t(h_i, h_{i+1}) = \alpha(h_i)p(h_{i+1}|h_i)p(x_{i+1}^t|h_{i+1})\beta(h_{i+1}) \)
15: end for
16: end for
EM for HMM: M step

1. \[ T_{\text{new}}(a|b) = \frac{\sum_t \sum_{l=2}^L q(h_t^l=a, h_{l-1}=b)}{\sum_t \sum_{l=2}^L q(h_t^{l-1}=b)} \]
2. \[ \pi_{\text{new}}^{m} = \frac{\sum_t q(h_1^t=m)}{\sum_t \sum_{h_1^t} q(h_1^t)} = \frac{\sum_t q(h_1^t=m)}{N} \]
3. \textbf{for} \ k = 1 \ \textbf{to} \ K \ \textbf{do} \]
4. \[ \nu_{\text{new}} = \arg\max_{\nu} \sum_t \sum_{h^t} q(h^t) \log \left\{ p(h_1^t) p(x_1^t|h_1^t) \prod_{l=2}^L T(h_l^t|h_{l-1}^t) p(x_l^t|h_l^t) \right\} \]
5. \textbf{end for}
We did ...

- Forward-Backward algorithm
- Viterbi algorithm
- Hidden Markov Models (inference, learning)