Introduction to Probabilistic Graphical Models

Christoph Lampert

IST Austria (Institute of Science and Technology Austria)
Inference in Hidden Markov Models
Hidden Markov Models

Reminder: a hidden Markov model (HMM) consists of

▶ a discrete Markov chain of hidden (or ‘latent’) variables $h_1:T$
▶ one observable (continuous or discrete) variable $v_i$ for each hidden variable $h_i$

\[
p(h_1:T, v_1:T) = p(v_1|h_1)p(h_1) \prod_{t=2}^{T} p(v_t|h_t)p(h_t|h_{t-1})
\]

We call the HMM stationary if

▶ the transition distribution $p(h_{t+1} = i'|h_t = i)$ and the emission distribution $p(v_t = j|h_t = i)$ do not depend on the position $t$, but only on the values $i, i'$ and $j$
HMM parameters

**Transition Distribution**

For a stationary HMM the transition distribution $p(h_{t+1}|h_t)$ is defined by the $H \times H$ transition matrix

$$A_{i',i} = p(h_{t+1} = i' | h_t = i)$$

and an initial distribution

$$a_i = p(h_1 = i).$$

**Emission Distribution**

For a stationary HMM and emission distribution $p(v_t|h_t)$ with discrete states $v_t \in \{1, \ldots, V\}$, we define a $V \times H$ emission matrix

$$B_{i,j} = p(v_t = i | h_t = j)$$

For continuous outputs, $h_t$ selects one of $H$ possible output distributions $p(v_t|h_t)$, $h_t \in \{1, \ldots, H\}$. 
The classical inference problems

<table>
<thead>
<tr>
<th>Inference Type</th>
<th>Description</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Filtering</strong></td>
<td>(Inferring the present)</td>
<td>$p(h_t</td>
</tr>
<tr>
<td><strong>Prediction</strong></td>
<td>(Inferring the future)</td>
<td>$p(h_t</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sometimes also $p(v_t</td>
</tr>
<tr>
<td><strong>Smoothing</strong></td>
<td>(Inferring the past)</td>
<td>$p(h_t</td>
</tr>
<tr>
<td><strong>Likelihood</strong></td>
<td></td>
<td>$p(v_{1:T})$</td>
</tr>
<tr>
<td><strong>Most likely Hidden path</strong></td>
<td>(Viterbi alignment)</td>
<td>argmax $h_{1:T} p(h_{1:T}</td>
</tr>
<tr>
<td><strong>Learning</strong></td>
<td>(Parameter estimation)</td>
<td>$\mathcal{D} \rightarrow A_{i,i'}, a_i, B_{i,j}$</td>
</tr>
</tbody>
</table>
The Burglar Scenario

You’re asleep upstairs in your house and awoken by noises from downstairs. You realise that a burglar is on the ground floor and attempt to understand where he his from listening to his movements.
The Burglar Scenario

You’re asleep upstairs in your house and awoken by noises from downstairs. You realise that a burglar is on the ground floor and attempt to understand where he is from listening to his movements.

The HMM view

- You mentally partition the ground floor into a $5 \times 5$ grid.
- For each grid position you know the probability that if someone is in that position the floorboard will creak.
- Similarly you know for each position the probability that someone will bump into something in the dark.
- The floorboard creaking and bumping into objects can occur independently.
- In addition you assume that the burglar will move only one grid square – forwards, backwards, left or right in a single timestep.

Can you infer the burglar’s position from the sounds?
The Burglar Scenario: Example

- Latent variable $h_t \in \{1, \ldots, 25\}$ denotes the positions on a $5 \times 5$ grid.
  - Dark squares mean probability 0.9, light means probability 0.1.
- Observed variables:
  - $v_t = (c_t, b_t) \in \{(n, n), (n, y), (y, n), (y, y)\}$
- Observed probability factorizes $p(v|h) = p(c|h)p(b|h)$
Burglar

**Localising the burglar through time for 10 time steps**

|   | Creaks and Bumps $v_t$ | Filtering $p(h_t|v_{1:t})$ | Smoothing $p(h_t|v_{1:T})$ | Viterbi argmax$_{h_{1:T}} p(h_{1:T}|v_{1:T})$ | True Burglar position $h_{1:T}^*$ |
|---|------------------------|----------------------------|-----------------------------|-----------------------------------------------|----------------------------------|
| (a) | ![Diagram](a.png)        | ![Diagram](b.png)            | ![Diagram](c.png)           | ![Diagram](d.png)                             | ![Diagram](e.png)               |

Note:
- (b) is computed on-the-fly in every time step
- (c) and (d) are computed offline after all observations are available
Real-world example

https://www.youtube.com/watch?v=4Z3shNP0dQA
Filtering $p(h_t | \nu_{1:t})$
Filtering $p(h_t | v_{1:t})$

\[
p(h_t, v_{1:t}) = \sum_{h_{t-1}} p(h_t, h_{t-1}, v_{1:t-1}, v_t)
\]

\[
= \sum_{h_{t-1}} p(v_t | v_{1:t-1}, h_t, h_{t-1}) p(h_t | v_{1:t-1}, h_{t-1}) p(v_{1:t-1}, h_{t-1})
\]

\[
= \sum_{h_{t-1}} p(v_t | h_t) p(h_t | h_{t-1}) p(h_{t-1}, v_{1:t-1})
\]
Filtering $p(h_t|v_{1:t})$

\[
p(h_t, v_{1:t}) = \sum_{h_{t-1}} p(h_t, h_{t-1}, v_{1:t-1}, v_t)
\]

\[
= \sum_{h_{t-1}} p(v_t|h_{1:t-1}, h_t, h_{t-1})p(h_t|v_{1:t-1}, h_{t-1})p(v_{1:t-1}, h_{t-1})
\]

\[
= \sum_{h_{t-1}} p(v_t|h_t)p(h_t|h_{t-1})p(h_{t-1}, v_{1:t-1})
\]

Hence if we define $\alpha(h_t) \equiv p(h_t, v_{1:t})$ the above gives the $\alpha$-recursion

\[
\alpha(h_t) = p(v_t|h_t)\sum_{h_{t-1}} p(h_t|h_{t-1})\alpha(h_{t-1}), \quad \text{with} \quad \alpha(h_1) = p(h_1, v_1) = p(v_1|h_1)p(h_1)
\]
Filtering $p(h_t | v_{1:t})$

$$p(h_t, v_{1:t}) = \sum_{h_{t-1}} p(h_t, h_{t-1}, v_{1:t-1}, v_t)$$

$$= \sum_{h_{t-1}} p(v_t | v_{1:t-1}, h_t, h_{t-1}) p(h_t | v_{1:t-1}, h_{t-1}) p(v_{1:t-1}, h_{t-1})$$

$$= \sum_{h_{t-1}} p(v_t | h_t) p(h_t | h_{t-1}) p(h_{t-1}, v_{1:t-1})$$

Hence if we define $\alpha(h_t) \equiv p(h_t, v_{1:t})$ the above gives the $\alpha$-recursion

$$\alpha(h_t) = p(v_t | h_t) \sum_{h_{t-1}} p(h_t | h_{t-1}) \alpha(h_{t-1}), \quad \text{with} \quad \alpha(h_1) = p(h_1, v_1) = p(v_1 | h_1) p(h_1)$$

Filtered posterior follows by normalization: $p(h_t | v_{1:t}) = \frac{p(h_t, v_{1:t})}{\sum_{\tilde{h}_t} p(\tilde{h}_t, v_{1:t})} = \frac{\alpha(h_t)}{\sum_{\tilde{h}_t} \alpha(\tilde{h}_t)}$
Inference in Hidden Markov Models

Likelihood $p(v_{1:T})$
Likelihood $p(v_{1:T})$ \[ p(v_{1:T}) = \sum_{h_T} p(h_T, v_{1:T}) = \sum_{h_T} \alpha(h_T) \]
Smoothing $p(h_t | v_1:T)$

To compute the smoothed quantity we consider how $h_t$ partitions the series into the past and future:

$$p(h_t, v_1:T) = p(h_t, v_1:t, v_{t+1:T}) = p(h_t, v_1:t) \cdot \alpha(h_t) \cdot \beta(h_t)$$

**Forward.** The term $\alpha(h_t)$ is obtained from the 'forward' $\alpha$ recursion.

**Backward.** The term $\beta(h_t)$ we will obtain using a 'backward' $\beta$ recursion as we show next.

The forward and backward recursions are independent and may therefore be run in parallel, with their results combined to obtain the smoothed posterior.

$$p(h_t | v_1:T) \equiv \gamma(h_t) = \alpha(h_t) \cdot \beta(h_t) \sum \bar{h}_t \alpha(\bar{h}_t) \cdot \beta(\bar{h}_t)$$

"Parallel Smoothing"
Smoothing $p(h_t | v_{1:T})$

To compute the smoothed quantity we consider how $h_t$ partitions the series into the past and future:

$$p(h_t, v_{1:T}) = p(h_t, v_{1:t}, v_{t+1:T})$$

$$= p(h_t, v_{1:t}) p(v_{t+1:T} | h_t, v_{1:t}) = \alpha(h_t) \beta(h_t)$$
Smoothing $p(h_t | v_{1:T})$

To compute the smoothed quantity we consider how $h_t$ partitions the series into the past and future:

$$p(h_t, v_{1:T}) = p(h_t, v_{1:t}, v_{t+1:T})$$

$$= \underbrace{p(h_t, v_{1:t})}_{\text{past}} \underbrace{p(v_{t+1:T} | h_t, v_{1:t})}_{\text{future}} = \alpha(h_t) \beta(h_t)$$

**Forward.** The term $\alpha(h_t)$ is obtained from the ‘forward’ $\alpha$ recursion.

**Backward.** The term $\beta(h_t)$ we will obtain using a ‘backward’ $\beta$ recursion as we show next.
Smoothing $p(h_t|v_{1:T})$

To compute the smoothed quantity we consider how $h_t$ partitions the series into the past and future:

$$p(h_t, v_{1:T}) = p(h_t, v_{1:t}, v_{t+1:T})$$

$$= p(h_t, v_{1:t}) p(v_{t+1:T}|h_t, v_{1:t}) = \alpha(h_t) \beta(h_t)$$

**Forward.** The term $\alpha(h_t)$ is obtained from the ‘forward’ $\alpha$ recursion.

**Backward.** The term $\beta(h_t)$ we will obtain using a ‘backward’ $\beta$ recursion as we show next.

The forward and backward recursions are independent and may therefore be run in parallel, with their results combined to obtain the smoothed posterior.

$$p(h_t|v_{1:T}) \equiv \gamma(h_t) = \frac{\alpha(h_t)\beta(h_t)}{\sum_{\bar{h}_t} \alpha(\bar{h}_t)\beta(\bar{h}_t)}$$

"Parallel Smoothing"
The $\beta$ recursion

\[
p(v_t:T | h_{t-1}) = \sum_{h_t} p(v_t, v_{t+1:T}, h_t | h_{t-1})
= \sum_{h_t} p(v_t | v_{t+1:T}, h_t, h_{t-1}) p(v_{t+1:T}, h_t | h_{t-1})
= \sum_{h_t} p(v_t | h_t) p(v_{t+1:T} | h_t, h_{t-1}) p(h_t | h_{t-1})
\]

Defining $\beta(h_t) \equiv p(v_{t+1:T} | h_t)$ gives the $\beta$-recursion

\[
\beta(h_{t-1}) = \sum_{h_t} p(v_t | h_t) p(h_t | h_{t-1}) \beta(h_t), \quad \text{for } 2 \leq t \leq T \quad \text{and} \quad \beta(h_T) = 1.
\]

Together the $\alpha - \beta$ recursions are called the Forward-Backward algorithm.
Smoothing $p(h_t|v_{1:T})$

"Correction Smoothing":

$$p(h_t|v_{1:T}) = \sum_{h_{t+1}} p(h_t, h_{t+1}|v_{1:T}) = \sum_{h_{t+1}} p(h_t|h_{t+1}, v_{1:t}, v_{t+1:T})p(h_{t+1}|v_{1:T})$$

This gives a recursion for $\gamma(h_t) \equiv p(h_t|v_{1:T})$:

$$\gamma(h_t) = \sum_{h_{t+1}} p(h_t|h_{t+1}, v_{1:t})\gamma(h_{t+1})$$

with $\gamma(h_T) \propto \alpha(h_T)$. The term $p(h_t|h_{t+1}, v_{1:t})$ may be computed using the filtered results $p(h_t|v_{1:t})$:

$$p(h_t|h_{t+1}, v_{1:t}) \propto p(h_{t+1}, h_t|v_{1:t}) \propto p(h_{t+1}|h_t)p(h_t|v_{1:t})$$

where the proportionality constant is found by normalisation. This is sequential since we need to first complete the $\alpha$ recursions, after which the $\gamma$ recursion may begin. This ‘corrects’ the filtered result. Interestingly, once filtering has been carried out, the evidential states $v_{1:T}$ are not needed during the subsequent $\gamma$ recursion.
Computing the pairwise marginal $p(h_t, h_{t+1} | v_{1:T})$

To implement the EM algorithm for learning, we require terms such as $p(h_t, h_{t+1} | v_{1:T})$.

\[
p(h_t, h_{t+1} | v_{1:T}) \propto p(v_{1:t}, v_{t+1}, v_{t+2:T}, h_{t+1}, h_t) \\
= p(v_{t+2:T} | v_{1:t}, v_{t+1}, h_t, h_{t+1}) p(v_{1:t}, v_{t+1}, h_{t+1}, h_t) \\
= p(v_{t+2:T} | h_{t+1}) p(v_{t+1} | v_{1:t}, h_t, h_{t+1}) p(v_{1:t}, h_{t+1}, h_t) \\
= p(v_{t+2:T} | h_{t+1}) p(v_{t+1} | h_{t+1}) p(h_{t+1} | v_{1:t}, h_t) p(v_{1:t}, h_t)
\]

After rearranging:

\[
p(h_t, h_{t+1} | v_{1:T}) \propto \alpha(h_t) p(v_{t+1} | h_{t+1}) p(h_{t+1} | h_t) \beta(h_{t+1})
\]
Prediction

Predicting the future hidden variable:

\[ p(h_{t+1}|v_{1:t}) = \]
Prediction

Predicting the future hidden variable:

\[ p(h_{t+1}|v_{1:t}) = \sum_{h_t} p(h_{t+1}|h_t) p(h_t|v_{1:t}) \]

(filtering)
Prediction

Predicting the future hidden variable:

\[
p(h_{t+1}|v_{1:t}) = \sum_{h_t} p(h_{t+1}|h_t) p(h_t|v_{1:t})
\]

Predicting the future observation The one-step ahead predictive distribution is given by

\[
p(v_{t+1}|v_{1:t}) =
\]
Prediction

Predicting the future hidden variable:

\[ p(h_{t+1} \mid v_{1:t}) = \sum_{h_t} p(h_{t+1} \mid h_t) p(h_t \mid v_{1:t}) \]

Predicting the future observation The one-step ahead predictive distribution is given by

\[ p(v_{t+1} \mid v_{1:t}) = \sum_{h_{t+1}} p(v_{t+1} \mid h_{t+1}) p(h_{t+1} \mid v_{1:t}) \]
Most likely joint state

The most likely path $h_{1:T}$ of $p(h_{1:T} | v_{1:T})$ is the same as the most likely state of

$$p(h_{1:T}, v_{1:T}) = \prod_t p(v_t | h_t)p(h_t | h_{t-1}) \quad \text{with } h_0 = \emptyset$$

Consider

$$\max_{h_T} \prod_{t=1}^T p(v_t | h_t)p(h_t | h_{t-1})$$

$$= \left\{ \prod_{t=1}^{T-1} p(v_t | h_t)p(h_t | h_{t-1}) \right\} \max_{h_T} p(v_T | h_T)p(h_T | h_{T-1})$$

$$\mu(h_{T-1})$$

The ”message” $\mu(h_{T-1})$ conveys information from the end of the chain to the penultimate timestep.
Most likely joint state

We can continue in this manner, defining the recursion

$$\mu(h_{t-1}) = \max_{h_t} p(v_t|h_t)p(h_t|h_{t-1})\mu(h_t), \quad \text{for } 2 \leq t \leq T \quad \text{and} \quad \mu(h_T) = 1.$$  

The effect of maximising over $h_2, \ldots, h_T$ is compressed into a message $\mu(h_1) \rightarrow$ the first entry most likely state, $h_1^*$, is given by

$$h_1^* = \arg\max_{h_1} p(v_1|h_1)p(h_1)\mu(h_1)$$

Once computed, backtracking gives the remaining entries:

$$h_t^* = \arg\max_{h_t} p(v_t|h_t)p(h_t|h_t^*_{t-1})\mu(h_t)$$
Learning Hidden Markov Models
Learning HMMs

Setting:

- given: data \( \mathcal{V} = \{v^1, \ldots, v^N\} \) of \( N \) sequences,
each sequence \( v^N = v_{1:T}^N \) is of length \( T_n \)
- goal: maximum-likelihood of HMM parameters \( \theta = (A, B, a) \), where
  - \( A \) is the HMM transition matrix, \( p(h_{t+1} | h_t) \)
  - \( B \) is the emission matrix, \( p(v_t | h_t) \)
  - \( a \) is the vector of initial state probabilities, \( p(h_1) \).
- assumption: the sequences are i.i.d. (within sequences, data are still dependent, of course)
- assumption: the number of hidden states \( H \) and observable states \( V \) is known and finite
Learning HMMs

Setting:

▶ given: data \( V = \{ v^1, \ldots, v^N \} \) of \( N \) sequences, each sequence \( v^N = v^N_{1:T_n} \) is of length \( T_n \)

▶ goal: maximum-likelihood of HMM parameters \( \theta = (A, B, a) \), where
  ▶ \( A \) is the HMM transition matrix, \( p(h_{t+1}|h_t) \)
  ▶ \( B \) is the emission matrix, \( p(v_t|h_t) \)
  ▶ \( a \) is the vector of initial state probabilities, \( p(h_1) \).

▶ assumption: the sequences are i.i.d. (within sequences, data are still dependent, of course)

▶ assumption: the number of hidden states \( H \) and observable states \( V \) is known and finite

Find \( \theta \) that maximizes

\[
p(v^1, \ldots, v^N; \theta) = \sum_{h^1, \ldots, h^N} p(v^1, \ldots, v^N, h^1, \ldots, h^N; \theta) = \prod_{n=1}^{N} \sum_{h^n} p(v^n, h^n; \theta)
\]
Learning HMMs

Setting:

- given: data \( \mathcal{V} = \{\mathbf{v}^1, \ldots, \mathbf{v}^N\} \) of \( N \) sequences,
  each sequence \( \mathbf{v}^N = \mathbf{v}^N_{1:T_N} \) is of length \( T_n \)
- goal: maximum-likelihood of HMM parameters \( \theta = (\mathbf{A}, \mathbf{B}, \mathbf{a}) \), where
  - \( \mathbf{A} \) is the HMM transition matrix, \( p(h_{t+1}|h_t) \)
  - \( \mathbf{B} \) is the emission matrix, \( p(\mathbf{v}_t|h_t) \)
  - \( \mathbf{a} \) is the vector of initial state probabilities, \( p(h_1) \).
- assumption: the sequences are i.i.d. (within sequences, data are still dependent, of course)
- assumption: the number of hidden states \( H \) and observable states \( V \) is known and finite

Find \( \theta \) that maximizes

\[
p(\mathbf{v}^1, \ldots, \mathbf{v}^N; \theta) = \sum_{\mathbf{h}^1, \ldots, \mathbf{h}^N} p(\mathbf{v}^1, \ldots, \mathbf{v}^N, \mathbf{h}^1, \ldots, \mathbf{h}^N; \theta) = \prod_{n=1}^{N} \sum_{\mathbf{h}^n} p(\mathbf{v}^n, \mathbf{h}^n; \theta)
\]

How?
Learning HMMs

Setting:

- given: data $\mathcal{V} = \{v_1, \ldots, v^N\}$ of $N$ sequences, each sequence $v^N = v_{1:T}^N$ is of length $T_n$
- goal: maximum-likelihood of HMM parameters $\theta = (A, B, a)$, where
  - $A$ is the HMM transition matrix, $p(h_{t+1}|h_t)$
  - $B$ is the emission matrix, $p(v_t|h_t)$
  - $a$ is the vector of initial state probabilities, $p(h_1)$.
- assumption: the sequences are i.i.d. (within sequences, data are still dependent, of course)
- assumption: the number of hidden states $H$ and observable states $V$ is known and finite

Find $\theta$ that maximizes

$$p(v^1, \ldots, v^N; \theta) = \sum_{h^1, \ldots, h^N} p(v^1, \ldots, v^N, h^1, \ldots, h^N; \theta)$$

How? EM-algorithm (for HMMs called Baum-Welch algorithm for historic reasons)
Learning HMMs

Like for GMM, construct a lower bound using a distribution $q(h^1, \ldots, h^N)$

$$
\log p(v^1, \ldots, v^N; \theta) = \log \sum_{h^1, \ldots, h^N} p(v^1, \ldots, v^N, h^1, \ldots, h^N; \theta)
\leq \mathbb{E}_{(h^1, \ldots, h^N) \sim q} \log p(v^1, \ldots, v^N, h^1, \ldots, h^N; \theta) - \mathbb{E}_{(h^1, \ldots, h^N) \sim q} \log q(h^1, \ldots, h^N)
$$

**EM algorithm:**

initialize $\theta^0$

for $t = 1, 2, \ldots$, until convergence do

$q^t \leftarrow \text{argmax}_q G(\theta^{t-1}, q) \quad // \text{E-step}$

$\theta^t \leftarrow \text{argmax}_\theta G(\theta, q^t) \quad // \text{M-step}$

end for
Learning HMMs

Like for GMM, construct a lower bound using a distribution $q(h^1, \ldots, h^N)$

$$\log p(v^1, \ldots, v^N; \theta) = \log \sum_{h^1, \ldots, h^N} p(v^1, \ldots, v^N, h^1, \ldots, h^N; \theta) \leq \mathbb{E}_{(h^1, \ldots, h^N) \sim q} \log p(v^1, \ldots, v^N, h^1, \ldots, h^N; \theta) - \mathbb{E}_{(h^1, \ldots, h^N) \sim q} \log q(h^1, \ldots, h^N) =: G(\theta, q)$$

**EM algorithm:**

initialize $\theta^0$

for $t = 1, 2, \ldots$, until convergence do

$q^t \leftarrow \arg\max_q G(\theta^{t-1}, q)$  // E-step

$\theta^t \leftarrow \arg\max_{\theta} G(\theta, q^t)$  // M-step

end for
E-step, Part 1

\[ q \leftarrow \arg\max_q G(\theta^{t-1}, q) \]

- as for GMMs:

\[ q^t \leftarrow p(h^1, \ldots, h^N | v^1, \ldots, v^N; \theta^{t-1}) \overset{i.i.d.}{=} \prod_{n=1}^N p(h^n | v^n; \theta^{t-1}) \]
E-step, Part 1

\[ q \leftarrow \arg\max_q G(\theta^{t-1}, q) \]

- as for GMMs:

\[ q^t \leftarrow p(h^1, \ldots, h^N|v^1, \ldots, v^N; \theta^{t-1}) \overset{i.i.d.}{=} \prod_{n=1}^{N} p(h^n|v^n; \theta^{t-1}) =: q^n(h^n) \]
E-step, Part 1

\[ q \leftarrow \text{argmax}_q G(\theta^{t-1}, q) \]

- as for GMMs:

\[ q^t \leftarrow p(h^1, \ldots, h^N | v^1, \ldots, v^N; \theta^{t-1}) \overset{i.i.d.}{=} \prod_{n=1}^{N} p(h^n | v^n; \theta^{t-1}) = \prod_{n=1}^{N} q^n(h^n) =: q^n(h^n) \]

later more...
M-step
M-step

\[
\mathbb{E}_{(h^1, \ldots, h^N) \sim q} \log p(v^1, \ldots, v^N, h^1, \ldots, h^N; \theta)
\]

\[
i.i.d. \quad \mathbb{E}_{(h^1, \ldots, h^n \sim q)} \sum_{n=1}^{N} \log p(v^n, h^n; \theta) = \sum_{n=1}^{N} \mathbb{E}_{h \sim q^n} \log p(v^n_{1:T^n}, h_{1:T^n}; \theta)
\]

\[
\text{HMM graph} \quad \sum_{n=1}^{N} \mathbb{E}_{h \sim q^n} \log \left[ p(h_1; a) \prod_{t=2}^{T^n} p(h_t|h_{t-1}; A) \prod_{t=1}^{T^n} p(v^n_t|h_t; B) \right]
\]

\[
= \sum_{n=1}^{N} \mathbb{E}_{h \sim q^n} \log p(h_1; a) + \sum_{n=1}^{N} \sum_{t=2}^{T^n} \mathbb{E}_{h \sim q^n} \log p(h_t|h_{t-1}; A) + \sum_{n=1}^{N} \sum_{t=1}^{T^n} \mathbb{E}_{q^n} \log p(v^n_t|h_t; B)
\]

\[
\text{sum of independent terms} \rightarrow \text{we can optimize for } a, A \text{ and } B \text{ separately}
\]
\[
\mathcal{L}_{\text{initial}}(a) = \sum_{n=1}^{N} \mathbb{E}_{h_1:T_n \sim q^n} \log p(h_1; a) = \sum_{n=1}^{N} \mathbb{E}_{h_1 \sim q^n} \log a_{h_1}
\]

\(a\) is a discrete probability distribution over \(H\) states, i.e. \(\sum_i a_i = 1\). Use Langragian:

\[
\mathcal{L}(a, \lambda) = \mathcal{L}_{\text{initial}}(a) - \lambda \left( \sum_i a_i - 1 \right)
\]

\[
\frac{d\mathcal{L}_{\text{initial}}(a)}{da_i}(a) = \frac{d}{da_i} \sum_{n=1}^{N} \mathbb{E}_{h_1 \sim q^n} \sum_{i'=1}^{H} \mathbb{I}[h_1 = i'] \log a_{i'} = \sum_{n=1}^{N} \mathbb{E}_{h_1 \sim q^n} \mathbb{I}[h_1 = i] \frac{1}{a_i} = \frac{1}{a_i} \sum_{n=1}^{N} q^n(h_1)
\]

\[
0 = \frac{d\mathcal{L}(a, \lambda)}{da_i}(\hat{a}, \hat{\lambda}) = \frac{1}{\hat{a}_i} \sum_{n=1}^{N} q^n(h_1 = i) - \lambda \quad \Rightarrow \quad \hat{a}_i = \frac{1}{\hat{\lambda}} \sum_{n=1}^{N} q^n(h_1)
\]

\[
0 = \frac{d\mathcal{L}(a, \lambda)}{d\lambda}(\hat{a}, \hat{\lambda}) = -1 + \sum_{i=1}^{H} \frac{1}{\hat{\lambda}} \sum_{n=1}^{N} q^n(h_1 = i) = -1 + \sum_{i=1}^{H} \frac{1}{\hat{\lambda}} \quad \Rightarrow \quad \hat{\lambda} = n
\]
Inference in Hidden Markov Models

\[ \mathcal{L}_{\text{transition}}(A) = \sum_{n=1}^{N} \sum_{t=2}^{T_n} \mathbb{E}_{h \sim q^n} \log p(h_t \mid h_{t-1}; A) \]

\[ = \sum_{n=1}^{N} \sum_{t=2}^{T_n} \mathbb{E}_{h_1:t^n \sim q^n} \sum_{i,i'=1}^{H} [h_t = i \land h_{t-1} = i'] \log A_{i,i'} \]

Each column of \( A \) is a (conditional) distribution over the rows, i.e. \( \sum_i A_{i,i'} = 1 \) for any \( i' \in \{1, \ldots, H\} \). We can optimize for any fixed \( i' \) independently:

\[ \mathcal{L}(A, \lambda) = \mathcal{L}_{\text{transition}}(A) - \lambda \left( \sum_i A_{i,i'} - 1 \right) \]

\[ \hat{A}_{i,i'} \propto \sum_{n=1}^{N} \sum_{t=2}^{T_n} q^n(h_t = i, h_{t-1} = i') \] with normalization to make \( \hat{A}_{i,i'} = 1 \) for each \( i' \)
\[ L_{\text{emission}}(A) = \sum_{n=1}^{N} \sum_{t=1}^{T^n} \mathbb{E}_{h \sim q^n} \log p(v^n_t | h_t; B) = \sum_{n=1}^{N} \sum_{t=1}^{T^n} \sum_{j=1}^{V} [v^n_t = j] \mathbb{E}_{h_1:T^n \sim q^n} \sum_{i=1}^{H} [h_t = i] \log B_{j,i} \]

Each column of \( B \) is a (conditional) distribution over the rows, i.e. \( \sum_j B_{j,i} = 1 \) for any \( j \in \{1, \ldots, V\} \). We can optimize for any fixed \( i \) independently:

\[ L(B, \lambda) = L_{\text{emission}}(B) - \lambda \left( \sum_i B_{j,i} - 1 \right) \]

\[ \hat{B}_{j,i} \propto \sum_{n=1}^{n} \sum_{t=1}^{T_n} [v^n_t = j] q^n(h_t = i) \quad \text{with normalization to make } \hat{B}_{j,i} = 1 \text{ for each } i \]
E-step, Part 2

For the M-step we compute:

\[
\hat{a}_i \propto \sum_{n=1}^{N} q^n(h_1) \quad \hat{A}_{i,i'} \propto \sum_{n=1}^{n} \sum_{t=2}^{T_n} q^n(h_t = i, h_{t-1} = i') \quad \hat{B}_{j,i} \propto \sum_{n=1}^{n} \sum_{t=1}^{T_n} [v^n_t = j] q^n(h_t = i)
\]

Of \( q^n(h) = p(h|v^n; \theta) \) we really only need:

- \( q^n(h_1) = p(h_1|v^n_{1:T}; \theta) \) for \( a \)
- \( q^n(h_t, h_{t-1}) = p(h_t, h_{t-1}|v^n_{1:T}; \theta) \) for \( A \)
- \( q^n(h_t) = p(h_t|v^n_{1:T}; \theta) \) for \( B \)

For computing all of these we have derived efficient ways in the previous section.
Inference in Hidden Markov Models

Learning Hidden Markov Models

EM for HMMs: Initialization

EM algorithm:

initialize $\theta^0$

for $t = 1, 2, \ldots$, until convergence do

$\phi^t \leftarrow \arg\max_q G(\theta^{t-1}, q)$  // E-step

$\theta^t \leftarrow \arg\max_{\theta} G(\theta, \phi^t)$  // M-step

end for

Parameter initialisation

- EM algorithm converges to a local maximum of the likelihood,
- in general, there is no guarantee that the algorithm will find the global maximum
- often, the initialization determined how good the found solution is
- practical strategy:
  - first, train non-temporal mixture model for $p(v) = \sum_h p(v|h)p(h)$
  - initialize $a$ and $B$ from this, and assume independence for $A$
For an HMM with continuous observation $v_t$, we need a model of $p(v_t|h_t)$, i.e. a continuous distribution for each state of $h_t$.

**Inference**
- filtering, smoothing, etc. remain largely unchanged, as everything is conditioned on $v_{1:T}$

**Learning**
- learning requires computing normalization constants w.r.t. $v$
- depending on the model, this might or might not be tractable