Learning for Hidden Markov Models

Michael U. Gutmann

Probabilistic Modelling and Reasoning (INFR11134)
School of Informatics, The University of Edinburgh

Spring Semester 2022

Recap

- Variational principle of performing inference via optimisation.
- Maximising the evidence lower bound (ELBO) with respect to the variational distribution allows us to (approximately) compute the marginal and the conditional from the joint.
- Overview of how to use the variational principle to solve inference and learning tasks.
- We studied in detail the case of latent variable models and autoencoders.
- For parameter estimation in presence of unobserved variables: Coordinate ascent on the ELBO leads to the (variational) EM algorithm.

Program

1. HMM parametrisation and the learning problem
2. Options for learning the parameters
3. Learning the parameters by EM

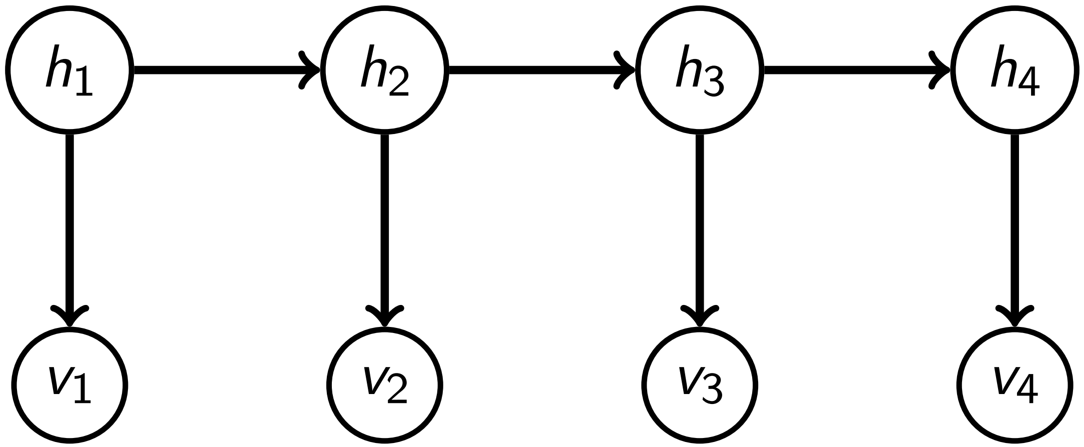
Program

1. HMM parametrisation and the learning problem
   - Assumptions: discrete case and stationarity
   - Constraints on the parameters
2. Options for learning the parameters
3. Learning the parameters by EM

Hidden Markov model

Specified by

- DAG (representing the independence assumptions)



- Transition distribution $p(h_i|h_{i-1})$
- Emission distribution $p(v_i|h_i)$
- Initial state distribution $p(h_1)$

The classical inference problems

- Classical inference problems:
  - Filtering: $p(h_t|v_{1:t})$
  - Smoothing: $p(h_t|v_{1:u})$ where $t < u$
  - Prediction: $p(h_t|v_{1:u})$ and/or $p(v_t|v_{1:u})$ where $t > u$
  - Most likely hidden path (Viterbi alignment): $\operatorname{argmax}_{h_{1:t}} p(h_{1:t}|v_{1:t})$
  - Posterior sampling (forward filtering, backward sampling): $h_{1:t} \sim p(h_{1:t}|v_{1:t})$
- Inference problems can be solved by message passing.
- Requires that the transition, emission, and initial state distributions are known.

Learning problem

- Data: $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$, where each $\mathcal{D}_j$ is a sequence of visibles of length d_j, i.e.

$$\mathcal{D}_j = (v_1^{(j)}, \ldots, v_{d_j}^{(j)})$$

- Assumptions:
  - All variables are discrete: $h_i \in \{1, \ldots K\}$, $v_i \in \{1, \ldots, M\}$.
  - Stationarity
- Parametrisation:
  - Transition distribution is parametrised by the matrix $\mathbf{A}$

$$p(h_i = k | h_{i-1} = k'; \mathbf{A}) = A_{k,k'} \qquad (A_{k',k} \text{ convention is also used})$$

  - Emission distribution is parametrised by the matrix $\mathbf{B}$

$$p(v_i = m | h_i = k; \mathbf{B}) = B_{m,k} \qquad (B_{k,m} \text{ convention is also used})$$

  - Initial state distribution is parametrised by the vector $\mathbf{a}$

$$p(h_1 = k; \mathbf{a}) = a_k$$

- Task: Use the data $\mathcal{D}$ to learn $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{a}$

Learning problem

- Since $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{a}$ represent (conditional) distributions, the parameters are constrained to be non-negative and to satisfy

$$\sum_{k=1}^{K} p(h_i = k | h_{i-1} = k') = \sum_{k=1}^{K} A_{k,k'} = 1 \qquad \text{for all } k'$$

$$\sum_{m=1}^{M} p(v_i = m | h_i = k) = \sum_{m=1}^{M} B_{m,k} = 1 \qquad \text{for all } k$$

$$\sum_{k=1}^{k} p(h_1 = k) = \sum_{k=1}^{K} a_k = 1$$

- Note: Much of what follows holds more generally for HMMs and does not use the stationarity assumption or that the h_i and v_i are discrete random variables.
- The parameters together will be denoted by $\boldsymbol{\theta}$.

Program

1. HMM parametrisation and the learning problem
   - Assumptions: discrete case and stationarity
   - Constraints on the parameters
2. Options for learning the parameters
3. Learning the parameters by EM

Program

1. HMM parametrisation and the learning problem
2. Options for learning the parameters
   - Learning by gradient ascent on the log-likelihood or by EM
   - Comparison
3. Learning the parameters by EM

Options for learning the parameters

- The model $p(\mathbf{h}, \mathbf{v}; \boldsymbol{\theta})$ is normalised but we have unobserved variables.
- Option 1: Gradient ascent on the log-likelihood

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} + \epsilon \sum_{j=1}^{n} \mathbb{E}_{p(\mathbf{h}|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} \left[\nabla_{\boldsymbol{\theta}} \log p(\mathbf{h}, \mathcal{D}_j; \boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}_{\text{old}}} \right]$$

  see slides *Intractable Likelihood Functions*
- Option 2: EM algorithm

$$\boldsymbol{\theta}_{\text{new}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{j=1}^{n} \mathbb{E}_{p(\mathbf{h}|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} \left[\log p(\mathbf{h}, \mathcal{D}_j; \boldsymbol{\theta}) \right]$$

  see slides *Variational Inference and Learning I*
- For HMMs, both are possible since the required posteriors can be computed with sum-product message passing.

Options for learning the parameters

Option 1: $\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} + \epsilon \sum_{j=1}^{n} \mathbb{E}_{p(\mathbf{h}|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} \left[\nabla_{\boldsymbol{\theta}} \log p(\mathbf{h}, \mathcal{D}_j; \boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}_{\text{old}}} \right]$

Option 2: $\boldsymbol{\theta}_{\text{new}} = \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{j=1}^{n} \mathbb{E}_{p(\mathbf{h}|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} [\log p(\mathbf{h}, \mathcal{D}_j; \boldsymbol{\theta})]$

- Similarities:
  - Both require computation of the posterior expectation.
  - In opt 2, assume the "M" step is performed by gradient ascent,

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \epsilon \sum_{j=1}^{n} \mathbb{E}_{p(\mathbf{h}|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} [\nabla_{\boldsymbol{\theta}} \log p(\mathbf{h}, \mathcal{D}_j; \boldsymbol{\theta})]$$

    where $\boldsymbol{\theta}$ is initialised with $\boldsymbol{\theta}_{\text{old}}$, and the final $\boldsymbol{\theta}'$ gives $\boldsymbol{\theta}_{\text{new}}$.
    If only one gradient step is taken, option 2 becomes option 1.
- Differences:
  - Unlike option 2, option 1 requires re-computation of the posterior after each ϵ update of $\boldsymbol{\theta}$, which may be costly.
  - In some cases (including HMMs), the "M"/argmax step can be performed analytically in closed form.

Program

1. HMM parametrisation and the learning problem
2. Options for learning the parameters
3. Learning the parameters by EM
    - E-step
    - M-step
    - EM (Baum-Welch) algorithm

The EM objective function

- Denote the objective in the EM algorithm by $J(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}})$,

$$J(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}) = \sum_{j=1}^{n} \mathbb{E}_{p(\mathbf{h}|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} \left[\log p(\mathbf{h}, \mathcal{D}_j; \boldsymbol{\theta})\right]$$

- Expected log-likelihood after filling-in the missing data
- We show next that for the HMM model in general, the full posteriors $p(\mathbf{h}|\mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})$ are not needed but just

$$p(h_i, h_{i-1} \mid \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) \qquad p(h_i \mid \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}).$$

  They can be obtained with the alpha-beta recursion (sum-product algorithm).
- Posteriors need to be computed for each observed sequence $\mathcal{D}_j$, and need to be re-computed after updating $\boldsymbol{\theta}$.

The EM objective function

- The HMM model factorises as

$$p(\mathbf{h}, \mathbf{v}; \boldsymbol{\theta}) = p(h_1; \mathbf{a}) p(v_1 | h_1; \mathbf{B}) \prod_{i=2}^{d} p(h_i | h_{i-1}; \mathbf{A}) p(v_i | h_i; \mathbf{B})$$

- For sequence $\mathcal{D}_j$, we have

$$\begin{aligned} \log p(\mathbf{h}, \mathcal{D}_j; \boldsymbol{\theta}) = & \log p(h_1; \mathbf{a}) + \log p(v_1^{(j)} | h_1; \mathbf{B}) + \\ & \sum_{i=2}^{d_j} \log p(h_i | h_{i-1}; \mathbf{A}) + \log p(v_i^{(j)} | h_i; \mathbf{B}) \end{aligned}$$

- Since

$$\begin{aligned} \mathbb{E}_{p(\mathbf{h} | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})} [\log p(h_1; \mathbf{a})] &= \mathbb{E}_{p(h_1 | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})} [\log p(h_1; \mathbf{a})] \\ \mathbb{E}_{p(\mathbf{h} | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})} [\log p(h_i | h_{i-1}; \mathbf{A})] &= \mathbb{E}_{p(h_i, h_{i-1} | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})} [\log p(h_i | h_{i-1}; \mathbf{A})] \\ \mathbb{E}_{p(\mathbf{h} | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})} \left[\log p(v_i^{(j)} | h_i; \mathbf{B})\right] &= \mathbb{E}_{p(h_i | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})} \left[\log p(v_i^{(j)} | h_i; \mathbf{B})\right] \end{aligned}$$

we do not need the full posterior but only the marginal posteriors and the joint of the neighbouring variables.

The EM objective function

With the factorisation (independencies) in the HMM model, the objective function thus becomes

$$
\begin{aligned}
J(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}) &= \sum_{j=1}^{n} \mathbb{E}_{p(\mathbf{h}|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} \left[\log p(\mathbf{h}, \mathcal{D}_j; \boldsymbol{\theta})\right] \\
&= \sum_{j=1}^{n} \mathbb{E}_{p(h_1|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} \left[\log p(h_1; \mathbf{a})\right] + \\
&\quad \sum_{j=1}^{n} \sum_{i=2}^{d_j} \mathbb{E}_{p(h_i, h_{i-1}|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} \left[\log p(h_i | h_{i-1}; \mathbf{A})\right] + \\
&\quad \sum_{j=1}^{n} \sum_{i=1}^{d_j} \mathbb{E}_{p(h_i|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} \left[\log p(v_i^{(j)} | h_i; \mathbf{B})\right]
\end{aligned}
$$

In the derivation so far we have not yet used the assumed parametrisation of the model. We insert these assumptions next.

The term for the initial state distribution

- We have assumed that

$$p(h_1 = k; \mathbf{a}) = a_k \qquad k = 1, \ldots, K$$

which we can write as

$$p(h_1; \mathbf{a}) = \prod_k a_k^{\mathbb{1}(h_1 = k)}$$

(like for the Bernoulli model, see slides *Basics of Model-Based Learning*)

- The log pmf is thus

$$\log p(h_1; \mathbf{a}) = \sum_k \mathbb{1}(h_1 = k) \log a_k$$

- Hence

$$\mathbb{E}_{p(h_1|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} [\log p(h_1; \mathbf{a})] = \sum_k \mathbb{E}_{p(h_1|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})} [\mathbb{1}(h_1 = k)] \log a_k$$

$$= \sum_k p(h_1 = k|\mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) \log a_k$$

The term for the transition distribution

- We have assumed that

$$p(h_i = k | h_{i-1} = k'; \mathbf{A}) = A_{k,k'} \qquad k, k' = 1, \dots K$$

which we can write as

$$p(h_i | h_{i-1}; \mathbf{A}) = \prod_{k,k'} A_{k,k'}^{\mathbb{1}(h_i = k, h_{i-1} = k')}$$

(see slides *Basics of Model-Based Learning*)

- Further:

$$\log p(h_i | h_{i-1}; \mathbf{A}) = \sum_{k,k'} \mathbb{1}(h_i = k, h_{i-1} = k') \log A_{k,k'}$$

- Hence $\mathbb{E}_{p(h_i, h_{i-1} | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})} [\log p(h_i | h_{i-1}; \mathbf{A})]$ equals

$$\sum_{k,k'} \mathbb{E}_{p(h_i, h_{i-1} | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})} \left[\mathbb{1}(h_i = k, h_{i-1} = k') \right] \log A_{k,k'}$$

$$= \sum_{k,k'} p(h_i = k, h_{i-1} = k' | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) \log A_{k,k'}$$

The term for the emission distribution

We can do the same for the emission distribution.

With

$$p(v_i|h_i;\mathbf{B}) = \prod_{m,k} B_{m,k}^{\mathbb{1}(v_i=m,h_i=k)} = \prod_{m,k} B_{m,k}^{\mathbb{1}(v_i=m)\mathbb{1}(h_i=k)}$$

we have

$$\mathbb{E}_{p(h_i|\mathcal{D}_j;\boldsymbol{\theta}_{\text{old}})}\left[\log p(v_i^{(j)}|h_i;\mathbf{B})\right] = \sum_{m,k} \mathbb{1}(v_i^{(j)} = m)p(h_i = k|\mathcal{D}_j,\boldsymbol{\theta}_{\text{old}})\log B_{m,k}$$

E-step for discrete-valued HMM

- Putting all together, we obtain the EM objective function for the HMM with discrete visibles and hiddens.

$$
\begin{aligned}
J(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}) = & \sum_{j=1}^{n} \sum_{k} p(h_1 = k | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) \log a_k + \\
& \sum_{j=1}^{n} \sum_{i=2}^{d_j} \sum_{k,k'} p(h_i = k, h_{i-1} = k' | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) \log A_{k,k'} + \\
& \sum_{j=1}^{n} \sum_{i=1}^{d_j} \sum_{m,k} \mathbb{1}(v_i^{(j)} = m) p(h_i = k | \mathcal{D}_j, \boldsymbol{\theta}_{\text{old}}) \log B_{m,k}
\end{aligned}
$$

- The objectives for **a**, and the columns of **A** and **B** decouple.
- Does not decouple in separate objectives for all parameters because of the constraint that the elements of **a** have to sum to one, and that the columns of **A** and **B** have to sum to one.

M-step

- We discuss the details for the maximisation with respect to $\mathbf{a}$. The other cases are done equivalently.
- Optimisation problem:

$$\max_{\mathbf{a}} \sum_{j=1}^{n} \sum_{k} p(h_1 = k | \mathcal{D}_j ; \boldsymbol{\theta}_{\text{old}}) \log a_k$$

$$\text{subject to } a_k \geq 0 \qquad \sum_k a_k = 1$$

- The non-negativity constraint could be handled by re-parametrisation, but the constraint is here not active (the objective is not defined for $a_k \leq 0$) and can be dropped.
- The normalisation constraint can be handled by using the methods of Lagrange multipliers (see e.g. Barber Appendix A.6).

M-step

- Lagrangian: $\sum_{j=1}^{n} \sum_k p(h_1 = k | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) \log a_k - \lambda(\sum_k a_k - 1)$
- The derivative with respect to a specific a_i is

$$\sum_{j=1}^{n} p(h_1 = i | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) \frac{1}{a_i} - \lambda$$

- Gives the necessary condition for optimality

$$a_i = \frac{1}{\lambda} \sum_{j=1}^{n} p(h_1 = i | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})$$

- The derivative with respect to λ gives back the constraint

$$\sum_i a_i = 1$$

- Set $\lambda = \sum_i \sum_{j=1}^{n} p(h_1 = i | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})$ to satisfy the constraint.
- The Hessian of the Lagrangian is negative definite, which shows that we have found a maximum.

M-step

- Since $\sum_i p(h_1 = i | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) = 1$, we obtain $\lambda = n$ so that

$$a_k = \frac{1}{n} \sum_{j=1}^{n} p(h_1 = k | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})$$

  Average of all posteriors of h_1 obtained by message passing.

- Equivalent calculations give

average joint proba

$$A_{k,k'} = \frac{\sum_{j=1}^{n} \sum_{i=2}^{d_j} p(h_i = k, h_{i-1} = k' | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})}{\sum_k \sum_{j=1}^{n} \sum_{i=2}^{d_j} p(h_i = k, h_{i-1} = k' | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})}$$

normalise: converts joint to conditional

  and

$$B_{m,k} = \frac{\sum_{j=1}^{n} \sum_{i=1}^{d_j} \mathbb{1}(v_i^{(j)} = m) p(h_i = k | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})}{\sum_m \sum_{j=1}^{n} \sum_{i=1}^{d_j} \mathbb{1}(v_i^{(j)} = m) p(h_i = k | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})}$$

  Inferred posteriors obtained by message passing are averaged over different sequences $\mathcal{D}_j$ and across each sequence (stationarity).

EM for discrete-valued HMM (Baum-Welch algorithm)

Given parameters $\boldsymbol{\theta}_{\text{old}}$

1. For each sequence $\mathcal{D}_j$ compute the posteriors

$$p(h_i, h_{i-1} \mid \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}}) \qquad p(h_i \mid \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})$$

   using the alpha-beta recursion (sum-product algorithm)

2. Update the parameters

$$a_k = \frac{1}{n} \sum_{j=1}^{n} p(h_1 = k | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})$$

$$A_{k,k'} = \frac{\sum_{j=1}^{n} \sum_{i=2}^{d_j} p(h_i = k, h_{i-1} = k' | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})}{\sum_k \sum_{j=1}^{n} \sum_{i=2}^{d_j} p(h_i = k, h_{i-1} = k' | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})}$$

$$B_{m,k} = \frac{\sum_{j=1}^{n} \sum_{i=1}^{d_j} \mathbb{1}(v_i^{(j)} = m) p(h_i = k | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})}{\sum_m \sum_{j=1}^{n} \sum_{i=1}^{d_j} \mathbb{1}(v_i^{(j)} = m) p(h_i = k | \mathcal{D}_j; \boldsymbol{\theta}_{\text{old}})}$$

Repeat step 1 and 2 using the new parameters for $\boldsymbol{\theta}_{\text{old}}$. Stop if change in likelihood or parameters is less than a threshold.

Program recap

1. HMM parametrisation and the learning problem
   - Assumptions: discrete case and stationarity
   - Constraints on the parameters
2. Options for learning the parameters
   - Learning by gradient ascent on the log-likelihood or by EM
   - Comparison
3. Learning the parameters by EM
   - E-step
   - M-step
   - EM (Baum-Welch) algorithm