COMS30035, Machine learning: Sequential Data 3: EM for HMMs

Edwin Simpson
edwin.simpson@bristol.ac.uk

Department of Computer Science, SCEEM
University of Bristol

November 6, 2023

Agenda

- Markov Models
- Hidden Markov Models
- EM for HMMs
- Linear Dynamical Systems

Hidden Markov Models (HMMs)

- We want to use maximum likelihood to estimate the HMM parameters:
  1. $\boldsymbol{A}$ - transition matrix
  2. $\boldsymbol{\pi}$ - initial state probabilities
  3. ϕ - parameters of the emission distributions
- We examine the *unsupervised* case where the sequence of states $\boldsymbol{Z}$ is not observed.
- $\ln p(\boldsymbol{X}|\boldsymbol{A},\pi,\phi) =$
  $\ln \sum_{\boldsymbol{Z}} \left\{ p(\boldsymbol{z}_1|\boldsymbol{\pi}) \prod_{n=2}^{N} p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1},\boldsymbol{A}) \prod_{n=1}^{N} p(\boldsymbol{x}_n|\phi,\boldsymbol{z}_n) \right\}$

Edwin Simpson
edwin.simpson@bristol.ac.uk

Likelihood for an HMM

- As with GMMs, there is no closed-form solution to the MLE, so we turn to EM
- Unlike GMM, the likelihood doesn't factorise over the data points:
    1. $\ln p(\boldsymbol{X}|\boldsymbol{A}, \boldsymbol{\pi}, \phi) = \ln \sum_{\boldsymbol{Z}} \left\{ p(\boldsymbol{z}_1|\boldsymbol{\pi}) \prod_{n=2}^{N} p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}, \boldsymbol{A}) \prod_{n=1}^{N} p(\boldsymbol{x}_n|\phi, \boldsymbol{z}_n) \right\}$
    2. The distribution of $\boldsymbol{z}_n$ depends on $\boldsymbol{z}_{n-1}$, which also depends on $\boldsymbol{z}_{n-2}$...
    3. Can't just sum over the values of z_n independently for each data point.
    4. So we have to sum over all K^N possible sequences $\boldsymbol{Z}$!

Expectation Maximisation (EM)

- Goal: maximise the expected log likelihood
- First, we define $Q(\theta|\theta^{old}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{X}, \theta^{old}) \ln p(\boldsymbol{X}, \boldsymbol{Z}|\theta)$.

1. Initialise the parameters with a random guess: $\theta^{old} = \{\boldsymbol{A}, \pi, \phi\}$.
2. **E-step**: use θ^{old} to compute expectations over $\boldsymbol{Z}$ required to compute $Q(\theta|\theta^{old})$.
3. **M-step**: choose the values of $\theta = \{\boldsymbol{A}, \pi, \phi\}$ that maximise $Q(\theta|\theta^{old})$.
4. Set $\theta^{old} = \theta$.
5. Repeat steps 2-4 until convergence.

E step

- We need to compute expectations of the latent states and pairs of latent states:
- Responsibilities: $\gamma(z_{nk}) = p\left(z_n = k | \boldsymbol{X}, \boldsymbol{\theta}^{(old)}\right)$
- State pairs: $\xi(z_{n-1,j}, z_{nk}) = p\left(z_{n-1} = j, z_n = k | \boldsymbol{X}, \boldsymbol{\theta}^{(old)}\right)$
- To compute these efficiently, we need the *forward-backward* algorithm (coming up in a few slides...)

M step

- $\pi_k = \gamma(z_{1k})$
- $A_{jk} = \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk}) / \sum_{n=2}^{N} \gamma(z_{n-1,j})$
- ϕ_k: parameters of posterior emission distributions, with observations weighted by responsibilities, $\gamma(z_{nk})$
  - If we have Gaussian emissions, the equations are the same as for GMM.
  - Discrete observations with value i:

$$\phi_{ki} = p(x_n = i | z_n = k) = \frac{\sum_{n=1}^{N} \gamma(z_{nk})[x_n = i]}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{1}$$

Forward-backward Algorithm

- A specific example of the *sum-product algorithm* used in the E-step
- Forward pass computes for each time-step n and state value k:

$$\begin{aligned}\alpha(z_{nk}) &= p(\boldsymbol{x}_1, ..., \boldsymbol{x}_n, z_n = k|\boldsymbol{\pi}, \boldsymbol{A}, \phi) \\ &= p(\boldsymbol{x}_n|z_n = k, \phi_k) \sum_{l=1}^{K} A_{lk}\alpha(z_{n-1,l})\end{aligned} \tag{2}$$

- Backward pass computes:

$$\begin{aligned}\beta(z_{nk}) &= p(\boldsymbol{x}_{n+1}, ..., \boldsymbol{x}_N|z_n = k, \boldsymbol{A}, \phi) \\ &= \sum_{l=1}^{K} A_{kl} p(\boldsymbol{x}_{n+1}|z_{n+1} = l, \phi_l)\beta(z_{n+1,l})\end{aligned} \tag{3}$$

Forward-backward Algorithm

- Use the computed α and β terms to compute our expectations over $\boldsymbol{z}$:

$$
\begin{aligned}
\tilde{\xi}(z_{n-1,l}, z_{nk}) =& p(\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}, z_{n-1} = l | \boldsymbol{A}, \pi, \phi) && \text{before} \\
& p(z_n = k | z_{n-1} = l, \boldsymbol{A}) p(\boldsymbol{x}_n | z_n = k, \phi) && \text{current} \\
& p(\boldsymbol{x}_{n+1}, ..., x_N | z_n = k, \boldsymbol{A}, \phi) && \text{after} \\
=& \alpha(z_{n-1,l}) A_{kl} p(\boldsymbol{x}_n | z_n = k, \phi) \beta(z_{nk})
\end{aligned} \tag{4}
$$

- $\xi(z_{n-1,l}, z_{nk}) = \tilde{\xi}(z_{n-1,l}, z_{nk}) / \sum_{l=1}^{K} \sum_{k=1}^{K} \tilde{\xi}(z_{n-1,l}, z_{nk})$
- $\gamma(z_{nk}) = \sum_{l=1}^{K} \xi(z_{n-1,l}, z_{nk})$

Putting It All Together...

1. Initialise the parameters with a random guess: $\boldsymbol{\theta}^{old} = \{\boldsymbol{A}, \boldsymbol{\pi}, \phi\}$.
2. **E-step** using $\boldsymbol{\theta}^{old}$:
   2.1 Run forward pass to compute $\alpha(z_{nk})$
   2.2 Run backward pass to compute $\beta(z_{nk})$
   2.3 Use $\alpha(z_{n-1,l})$ and $\beta(z_{nk})$ to compute $\xi(z_{n-1,l}, z_{nk})$ and $\gamma(z_{nk})$.
3. **M-step** using $\xi(z_{n-1,l}, z_{nk})$ and $\gamma(z_{nk})$, update $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{A}, \phi\}$.
4. Set $\boldsymbol{\theta}^{old} = \boldsymbol{\theta}$.
5. Repeat steps 2-4 until convergence.

By summing inside each forward and backward computations, we now have an algorithm that is linear ($\mathcal{O}(N)$) rather than exponential ($\mathcal{O}(K^N)$) in the sequence length 😎.

Viterbi Algorithm

- Given our estimated model parameters $\theta = \{\boldsymbol{\pi}, \boldsymbol{A}, \phi\}$, how can we predict a sequence of hidden states $\boldsymbol{Z}$?
- Most probable labels (given by the values of $\gamma(z_{nk})$) are not the same as the most probable *sequence*!
- We apply a *max-sum* algorithm called *viterbi* to "decode" the sequence with $\mathcal{O}(N)$ computational cost.

Viterbi Algorithm

- Forward pass: compute the probability of the most likely sequence that leads to each possible state at time n.
- Backward pass: starting with the most likely final state and recursing backwards, choose the previous state $n-1$ that makes the chosen state at n most likely.

Viterbi Algorithm

- Forward pass:
  1. $\omega(z_{1k}) = \ln \pi_k + \ln p(\boldsymbol{x}_1 | z_1 = k)$
  2. For $n = 2$ to N compute for each state value k:
     - 2.1 $\omega(z_{nk}) = \max_l \left\{ \omega(z_{n-1,l}) + \ln p(z_n = k | z_{n-1} = l) \right\} + \ln p(\boldsymbol{x}_n | z_n = k)$.
     - 2.2 $\psi(z_{nk}) = \operatorname{argmax}_l \left\{ \omega(z_{n-1,l}) + \ln p(z_n = k | z_{n-1} = l) \right\} + \ln p(\boldsymbol{x}_n | z_n = k)$.
     - 2.3 Passes messages from the start of the sequence to the end.
- Backward pass:
  1. Most likely final state: $\hat{z}_N = \operatorname{argmax}_k \omega(z_{Nk})$.
  2. For $n = N - 1$ to 1: $\hat{z}_n = \psi(z_{n+1, \hat{z}_{n+1}})$.
- There are multiple paths leading to each possible state at each step n. We keep only the path with the highest probability, so we don't have to compute the likelihood of every complete path from 1 to N.

Summary

By computing sums and maximums at each timestep we can perform inference over an exponential number of sequences. We use the...

- Forward-backward algorithm, an instance of the more general *sum-product* algorithm to marginalise over sequences of hidden states.
- Viterbi algorithm, an instance of the more general *max-sum* algorithm to find the most likely sequence of hidden states.

Now do the quiz!

Please do the quiz for this lecture on Blackboard.
Next up: linear dynamical systems for modelling continuous states.

Edwin Simpson
edwin.simpson@bristol.ac.uk