Hidden Markov Models

Peng Chen

School of Management
University of Science and Technology of China

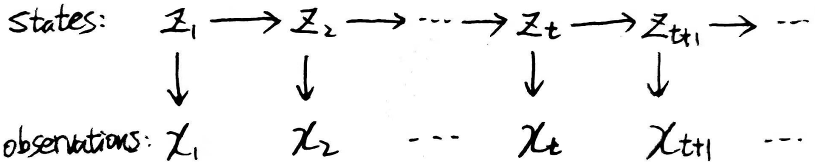
December 10, 2020

Table of Contents

Table of Contents

Introduction

- A hidden Markov model or HMM consists of a discrete-time, discrete-state Markov chain, with hidden states $z_t \in \{1, \cdots, K\}$, plus an observation model $p(x_t|z_t)$.



Notation

- State Space $\{1, \cdots, K\}$, Observation Space.
- State Sequence: z_t, Observation Sequence: x_t, $t = 1, \cdots, T$.
- Parameter: $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$, where $\pi(i) = p(z_1 = i)$ is the initial state distribution, $A(i,j) = p(z_t = j | z_{t-1} = i)$ is the transition matrix, and $\boldsymbol{B}$ are the parameters of the class-conditional densities $p(x_t | z_t = j)$.

Example (Boxes and Balls Model)

- There are 4 boxes with each containing 10 balls (red and white).
- We pick a box randomly with equal probability, and then pick a ball from this box randomly.
- State Space: $\{1, 2, 3, 4\}$, Observation Space: $\{R, W\}$.
- Suppose we have the observation sequence: $x_{1:5} = \{R, R, W, W, R\}$.

Box	Red	White
1	5	5
2	3	7
3	6	4
4	8	2

Example (Boxes and Balls Model)

- Initial state distribution $\boldsymbol{\pi} = (0.25, 0.25, 0.25, 0.25)^T$.
- The transition rule can be denoted by the transition matrix $\boldsymbol{A}$:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.4 & 0 & 0.6 & 0 \\ 0 & 0.4 & 0 & 0.6 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

- The class-conditional distribution matrix $\boldsymbol{B}$ is

$$\begin{bmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \\ 0.6 & 0.4 \\ 0.8 & 0.2 \end{bmatrix}$$

Model Assumptions

- The observations are conditionally independent given the states:

$$P_{z_{t+1}|z_{1:t},x_{1:t}} = P_{z_{t+1}|z_t}.$$

- The state process is Markov:

$$P_{z_{t+1}|z_{1:t},x_{1:t}} = P_{z_{t+1}|z_t}.$$

- (Properties)
  The joint process is Markov:

$$P_{x_{t+1:T},z_{t+1:T}|x_{1:t},z_{1:t}} = P_{x_{t+1:T},z_{t+1:T}|x_t,z_t}.$$

  Given z_t, x_t is conditionally independent of everything else:

$$P_{x_t|x_{1:t-1},x_{t+1:T},z_{1:T}} = P_{x_t|z_t}.$$

HMM

- The corresponding joint distribution has the form

$$\begin{aligned} p(z_{1:T}, x_{1:T}) =& p(z_{1:T}) p(x_{1:T}|z_{1:T}) \\ =& \left[p(z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}) \right] \left[\prod_{t=1}^{T} p(x_t|z_t) \right]. \end{aligned}$$

- $p(z_{1:T}) = p(z_1) \prod_{t=2}^{T} p(z_t|z_{t-1})$ (Bayes rule & The state is Markov)
- $p(x_{1:T}|z_{1:T}) = \prod_{t=1}^{T} p(x_t|z_t)$
  (The observations are conditionally independent given the states)

Applications

- HMMs have the advantage over Markov models in that they can represent long-range dependencies between observations, mediated via the latent variables.
- Two common scenarios:
  Online scenario: compute $p(z_t|x_{1:t})$.
  Offline scenario: compute $p(z_t|x_{1:T})$.

Examples of Applications

- Automatic speech recognition:
  Here x_t represents features extracted from the speech signal, and z_t represents the word that is being spoken. The transition model $p(z_t|z_{t1})$ represents the language model, and the observation model $p(x_t|z_t)$ represents the acoustic model.
- Activity recognition
  Here x_t represents features extracted from a video frame, and z_t is the class of activity.

Examples of Applications

- Part of speech tagging
  Here x_t represents a word, and z_t represents its part of speech (noun, verb, adjective, etc.)
- Gene finding
  Here x_t represents the DNA nucleotides (A,C,G,T), and z_t represents whether we are inside a gene-coding region or not.
- Protein sequence alignment
  Here x_t represents an amino acid, and z_t represents whether this matches the latent consensus sequence at this location. This model is called a profile HMM.

Table of Contents

Types of inference problems

- We discuss how to infer the hidden state sequence of an HMM, assuming the parameters are known.
- Consider an example called the occasionally dishonest casino.
- In this model, $x_t \in \{1, 2, \cdots, 6\}$ represents which dice face shows up, and $z_t \in \{1(\text{fair}), 2(\text{loaded})\}$ represents the identity of the dice that is being used.

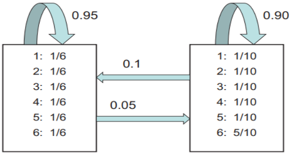


Figure 17.9 An HMM for the occasionally dishonest casino. The blue arrows visualize the state transition diagram **A**. Based on (Durbin et al. 1998, p54).

Types of inference problems

- We can just see the rolls and want to infer which dice is being used. Following are different kinds of inference.
- Filtering means to compute the belief state $p(z_t|x_{1:t})$ online, or recursively, as the data streams in.
- Smoothing means to compute $p(z_t|x_{1:T})$ offline, given all evidence.

Listing 17.1 Example output of `casinoDemo`

```
Rolls:    664153216162115234653214356634261655234232315142464156663246
Die:      LLLLLLLLLLLLLLFFFFFFLLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFLLLLLLLL
```

Types of inference problems

- Fixed lag smoothing is an compromise between online and offline estimation for its computation of $p(z_{t-l}|x_{1:t})$, where $l > 0$ is called the lag.
- Prediction We might want to predict the future given the past, i.e., to compute $p(z_{t+h}|x_{1:t})$, where $h > 0$ is called the prediction horizon.
- MAP estimation This means computing $\text{argmax}_{z_{1:T}} p(z_{1:T}|x_{1:T})$, which is a most probable state sequence (Viterbi decoding).

Types of inference problems

- Posterior samples We can obtain more information from the sample paths sampled from the posterior, $z_{1:T} \sim p(z_{1:T}|x_{1:T})$, than the sequence of marginals computed by smoothing.
- Probability of the evidence We can compute the probability of the evidence, $p(x_{1:T})$, by summing up over all hidden paths, $p(x_{1:T}) = \sum_{z_{1:T}} p(z_{1:T}, x_{1:T})$.

Types of inference problems

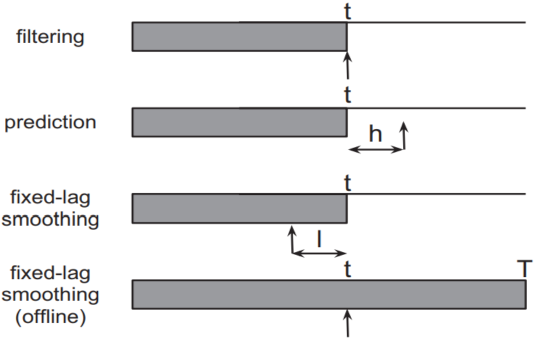


Figure 17.11 The main kinds of inference for state-space models. The shaded region is the interval for which we have data. The arrow represents the time step at which we want to perform inference. t is the current time, T is the sequence length, ℓ is the lag and h is the prediction horizon. See text for details.

The forwards algorithm

- To compute the filtered marginals, $\alpha_t = p(z_t|x_{1:t})$ in an HMM.
- The algorithm has two steps: the prediction step and the update step.
- Prediction step We compute the one-step-ahead predictive density,

$$p(z_t = j|x_{1:t-1}) = \sum_i p(z_t = j|z_{t-1} = i)p(z_{t-1} = i|x_{1:t-1}),$$

which acts as the new prior for time t.

The forwards algorithm

- Update step We absorb the observed data from time t using Bayes rule,

$$\begin{aligned}
\alpha_t(j) &\triangleq p(z_t = j|x_{1:t}) = p(z_t = j|x_t, x_{1:t-1}) \\
&= \frac{p(x_t|z_t = j, x_{1:t-1})p(z_t = j|x_{1:t-1})}{p(x_t|x_{1:t-1})} \\
&= \frac{p(x_t|z_t = j)p(z_t = j|x_{1:t-1})}{Z_t} \\
&\propto p(x_t|z_t = j)p(z_t = j|x_{1:t-1}),
\end{aligned}$$

where

$$\begin{aligned}
Z_t &= \sum_j p(x_t|z_t = j)p(z_t = j|x_{1:t-1}) \\
&= \sum_j p(x_t, z_t = j|x_{1:t-1}) = p(x_t|x_{1:t-1}).
\end{aligned}$$

The forwards algorithm

- The distribution $p(z_t|x_{1:t})$ is called the (filtered) belief state at time t, and is a vector of K numers, denoted by $\boldsymbol{\alpha}_t$.
- We can rewrite the update in the matrix-vector form:

$$\boldsymbol{\alpha}_t \propto \boldsymbol{\phi}_t \odot (\boldsymbol{\Psi}^T \boldsymbol{\alpha}_{t-1}),$$

  where $\phi_t(j) = p(x_t|z_t = j)$ is the local evidence at time t, $\Psi(i,j) = p(z_t = j|z_{t-1} = i)$ is the transition matrix, and $\odot$ is the Hadamard product.

The forwards algorithm

- In addition to computing the hidden states, we can use this algorithm to compute the log probability of the evidence:

$$\log p(x_{1:T}|\boldsymbol{\theta}) = \sum_{t=1}^{T} \log p(x_t|x_{1:t-1}) = \sum_{t=1}^{T} \log Z_t.$$

Algorithm 17.1: Forwards algorithm

Input: Transition matrices $\psi(i,j) = p(z_t = j|z_{t-1} = i)$, local evidence vectors $\psi_t(j) = p(\mathbf{x}_t|z_t = j)$, initial state distribution $\pi(j) = p(z_1 = j)$;

$[\alpha_1, Z_1] = \text{normalize}(\boldsymbol{\psi}_1 \odot \boldsymbol{\pi})$;

for $t = 2 : T$ **do**

    $[\alpha_t, Z_t] = \text{normalize}(\boldsymbol{\psi}_t \odot (\boldsymbol{\Psi}^T \boldsymbol{\alpha}_{t-1}))$;

Return $\boldsymbol{\alpha}_{1:T}$ and $\log p(\mathbf{y}_{1:T}) = \sum_t \log Z_t$;

Subroutine: $[\mathbf{v}, Z] = \text{normalize}(\mathbf{u}) : Z = \sum_j u_j; \quad v_j = u_j/Z$;

Examples

- Consider the Boxes and Balls Model with observation sequence $\{R, R, W, W, R\}$ and parameters in page 7, and then we have

$$
\begin{aligned}
\boldsymbol{\alpha}_1 &= (0.227, 0.136, 0.273, 0.364) \\
\boldsymbol{\alpha}_2 &= (0.121, 0.136, 0.291, 0.452) \\
\boldsymbol{\alpha}_3 &= (0.158, 0.364, 0.304, 0.173) \\
\boldsymbol{\alpha}_4 &= (0.363, 0.343, 0.199, 0.095) \\
\boldsymbol{\alpha}_5 &= (0.353, 0.164, 0.240, 0.243) \\
p(x_{1:5}|\boldsymbol{\theta}) &= 0.032
\end{aligned}
$$

The forwards-backwards algorithm

- To compute the smoothed marginals $p(z_t = j|x_{1:T})$.
- We break the chain into two parts, the past and the future, by conditioning on z_t:

$$p(z_t = j|x_{1:T}) \propto p(z_t = j, x_{t+1:T}|x_{1:t}) \propto p(z_t = j|x_{1:t})p(x_{t+1:T}|z_t = j).$$

- Define $\beta_t(j) \triangleq p(x_{t+1:T}|z_t = j)$ as the conditional likelihood of future evidence given that the hidden state at time t is j.
- Define $\gamma_t(j) \triangleq p(z_t = j|x_{1:T})$ as the desired smoothed posterior marginal. Thus, $\gamma_t(j) \propto \alpha_t(j)\beta_t(j)$.

The forwards-backwards algorithm

- We now describe how to recursively compute the β's in a right-to-left fashion.
- If we have already computed β_t, we can compute β_{t-1} as follows:

$$
\begin{aligned}
\beta_{t-1}(i) &= p(x_{t:T}|z_{t-1}=i) \\
&= \sum_j p(z_t=j, x_t, x_{t+1:T}|z_{t-1}=i) \\
&= \sum_j p(x_{t+1:T}|z_t=j, z_{t-1}=i, x_t)p(z_t=j, x_t|z_{t-1}=i) \\
&= \sum_j p(x_{t+1:T}|z_t=j)p(x_t|z_t=j, z_{t-1}=i)p(z_t=j|z_{t-1}=i) \\
&= \sum_j \beta_t(j)\phi_t(j)\Psi(i,j).
\end{aligned}
$$

The forwards-backwards algorithm

- We rewrite the equation in matrix-vector form as

$$\boldsymbol{\beta}_{t-1} = \boldsymbol{\Psi}(\boldsymbol{\phi}_t \odot \boldsymbol{\beta}_t).$$

- The base case is

$$\beta_T(i) = p(x_{T+1:T}|z_T = i) = p(\emptyset|z_T = i) = 1,$$

which is the probability of a non-event.

Two-slice smoothed marginals

- In order to estimate the parameters of the transition matrix using EM, we need to compute:

$$N_{ij} = \sum_{t=1}^{T-1} E[I(z_t = i, z_{t+1} = j)|x_{1:T}] = \sum_{t=1}^{T-1} p(z_t = i, z_{t+1} = j|x_{1:T}).$$

Two-slice smoothed marginals

- The term $p(z_t = i, z_{t+1} = j|x_{1:T})$ is called a (smoothed) two-slice marginal, and can be computed as follows:

$$\begin{aligned}
\xi_{t,t+1}(i,j) &\triangleq p(z_t = i, z_{t+1} = j|x_{1:T}) \\
&= p(z_t|x_{1:T})p(z_{t+1}|z_t, x_{1:T}) \\
&\propto p(z_t|x_{1:t})p(x_{t+1:T}|z_t)p(z_{t+1}|z_t, x_{1:T}) \\
&= p(z_t|x_{1:t})p(x_{t+1:T}|z_t)p(z_{t+1}|z_t, x_{t+1:T}) \\
&= p(z_t|x_{1:t})p(x_{t+1:T}|z_t, z_{t+1})p(z_{t+1}|z_t) \\
&\propto p(z_t|x_{1:t})p(x_{t+1}|z_{t+1})p(x_{t+2:T}|z_{t+1})p(z_{t+1}|z_t) \\
&= \alpha_t(i)\phi_{t+1}(j)\beta_{t+1}(j)\psi(i,j)
\end{aligned}$$

- In matrix-vector form, we have

$$\boldsymbol{\xi}_{t,t+1} \propto \boldsymbol{\Psi} \odot (\boldsymbol{\alpha}_t(\boldsymbol{\phi}_{t+1} \odot \boldsymbol{\beta}_{t+1})^T).$$

Time and space complexity

- A straightforward implementation of FB takes $O(K^2T)$ time, since we must perform a $K \times K$ matrix multiplication at each step.
- In some cases, the bottleneck is memory, not time. It is possible to devise a simple divide-and-conquer algorithm that reduces the space complexity from $O(KT)$ to $O(K\log T)$ at the cost of increasing the running time from $O(K^2T)$ to $O(K^2T\log T)$.

The Viterbi algorithm

- To compute $z^* = \text{argmax}_{z_{1:T}} p(z_{1:T}|x_{1:T})$.
- It is equivalent to computing a shortest path through the trellis diagram in Figure 17.12 with the weight of a path $z_1, z_2, ..., z_T$ given by the log probability $\log p(z_{1:T}, x_{1:T})$

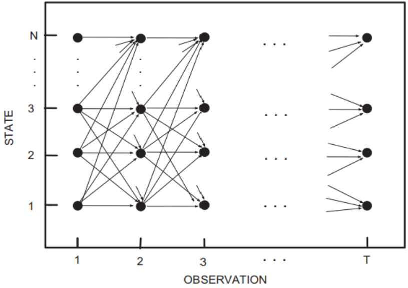


Figure 17.12 The trellis of states vs time for a Markov chain. Based on (Rabiner 1989).

MAP vs MPE

- The jointly most probable sequence of states is not necessarily the same as the sequence of marginally most probable states.
- The former is what Viterbi computes.
- The latter is given by the maximizer of the posterior marginals or MPM:

$$\hat{z} = (\text{argmax}_{z_1} p(z_1|x_{1:T}), \cdots, \text{argmax}_{z_T} p(z_T|x_{1:T})).$$

	$X_1 = 0$	$X_1 = 1$	
$X_2 = 0$	0.04	0.3	0.34
$X_2 = 1$	0.36	0.3	0.66
	0.4	0.6	

The Viterbi algorithm

- Define the probability of ending up in state j at time t, given that we take the most probable path, as

$$\delta_t(j) \triangleq \max_{z_1,\cdots,z_{t-1}} p(z_{1:t-1}, z_t = j | x_{1:t}).$$

- The most probable path to state j at time t must consist of the most probable path to some other state i at time $t-1$, followed by a transition from i to j. Hence,

$$\delta_t(j) = \max_i \delta_{t-1}(i)\psi(i,j)\phi_t(j).$$

The Viterbi algorithm

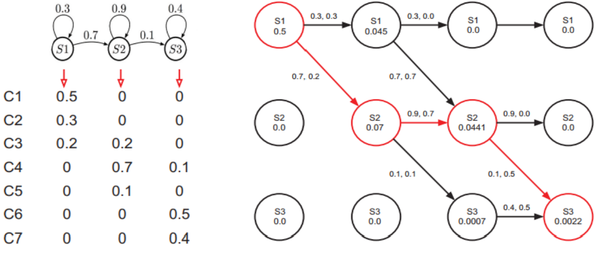
- Let
$$a_t(j) = \underset{i}{argmax}\, \delta_{t-1}(i)\psi(i,j)\phi_t(j),$$
and it is the most likely previous state on the most probable path to $z_t = j$.
- We initialize by setting $\delta_1(j) = \pi_j \phi_1(j)$ and terminate by computing the most probable final state
$$z_T^* = \underset{i}{argmax}\, \delta_T(i).$$
- We can then compute the most probable sequence of states using traceback:
$$z_t^* = a_{t+1}(z_{t+1}^*).$$

The Viterbi algorithm

- Inorder to avoid numerical underflow, we can normalize the $\boldsymbol{\delta}_t$ terms at each step.
- Unlike the forwards-backwards case, we can easily work in the log domain ($\log \max = \max \log, \log \sum \neq \sum \log$), which can result in a significant speedup in the case of Gaussian observation models.

Example

- Consider a simple HMM with observation space $\{C_1, C_2, \cdots, C_7\}$, and its transition matrix and class-conditional probability is expressed in the following figure.
- Suppose we observe the sequence of observations $\{C_1, C_3, C_4, C_6\}$.
- The model starts in state S1.



(a) (b)

Example

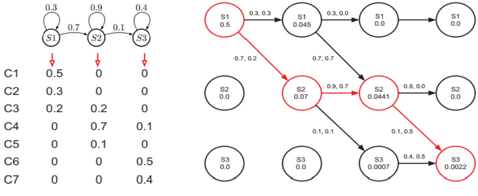
$$
\begin{aligned}
t=1,\quad & \delta_1(1)=0.5,\\
& \delta_1(2)=0,\\
& \delta_1(3)=0,\\
t=2,\quad & \delta_2(1)=\delta_1(1)\psi(1,1)\phi_1(C_3)=0.5\times 0.3\times 0.3=0.045,\\
& \delta_2(2)=\delta_1(1)\psi(1,2)\phi_2(C_3)=0.5\times 0.7\times 0.2=0.07,\\
& \delta_2(3)=0;\\
t=3,\quad & \delta_3(1)=0,\\
& \delta_3(2)=\max\{\delta_2(1)\psi(1,2)\phi_2(C_4),\delta_2(2)\psi(2,2)\phi_2(C_4)\}\\
& \qquad\quad =\max\{0.02205,0.0441\}=0.0441,\\
& \delta_3(3)=\delta_2(1)\psi(2,3)\phi_3(C_4)=0.07\times 0.1\times 0.1=0.0007;
\end{aligned}
$$

Example

- (continue)

$$
\begin{aligned}
t = 4, \quad \delta_4(1) &= 0, \\
\delta_4(2) &= 0, \\
\delta_4(3) &= \max\{\delta_3(2)\psi(2,3)\phi_3(C_6), \delta_3(3)\psi(3,3)\phi_3(C_6)\} \\
&= \max\{0.0022, 0.0001\} = 0.0022;
\end{aligned}
$$

- So, we have $z_4^* = S_3$ and then we can calculat that $z_3^* = S_2, z_2^* = S_2, z_1^* = S_1$ using traceback.



(a) (b)

The Viterbi algorithm

- The time complexity of Viterbi is clearly $O(K^2T)$ in general, and the space complexity is $O(KT)$, both the same as forwards-backwards.
- The Viterbi algorithm can be extended to return the top N paths, which is called the N-best list.

Forwards filtering, backwards sampling

- To sample paths from the posterior: $z^s_{1:T} \sim p(z_{1:T}|x_{1:T})$.
- One way is to do as follow:
  run forwards backwards, to compute the two-slice smoothed posteriors, $p(z_{t-1,t}|x_{1:T})$; next compute the conditionals $p(z_t|z_{t-1}, x_{1:T})$ by normalizing; sample from the initial pair of states, $z^*_{1,2} \sim p(z_{1,2}|x_{1:T})$; finally, recursively sample $z^*_t \sim p(z_t|z^*_{t-1}, x_{1:T})$.
  (a forwards-backwards pass and an additional forwards sampling pass.)

Forwards filtering, backwards sampling

- An alternative is to do the forwards pass, and then perform sampling in the backwards pass.
- We write the joint from right to left :

$$p(z_{1:T}|x_{1:T}) = p(z_T|x_{1:T}) \prod_{t=T-1}^{1} p(z_t|z_{t+1}, x_{1:T}).$$

- We can then sample z_t given future sampled states using

$$z_t^s \sim p(z_t|z_{t+1:T}, x_{1:T}) = p(z_t|z_{t+1}, z_{t+2:T}, x_{1:t}, x_{t+1:T}) = p(z_t|z_{t+1}^s, x_{1:t})$$

Forwards filtering, backwards sampling

- The sampling distribution is given by

$$
\begin{aligned}
p(z_t = i | z_{t+1} = j, x_{1:t}) &= p(z_t | z_{t+1}, x_{1:t}, x_{t+1}) \\
&= \frac{p(z_{t+1}, z_t | x_{1:t+1})}{p(z_{t+1} | x_{1:t+1})} \\
&\propto \frac{p(x_{t+1} | z_{t+1}, z_t, x_{1:t}) p(z_{t+1}, z_t | x_{1:t})}{p(z_{t+1} | x_{1:t+1})} \\
&= \frac{p(x_{t+1} | z_{t+1}) p(z_{t+1} | z_t, x_{1:t}) p(z_t | x_{1:t})}{p(z_{t+1} | x_{1:t+1})} \\
&= \frac{\phi_{t+1}(j) \psi(i, j) \alpha_t(i)}{\alpha_{t+1}(j)}
\end{aligned}
$$

- The base case is $z_T^s \sim p(z_T = i | x_{1:T}) = \alpha_T(i)$.

Table of Contents

Learning for HMMs

- To estimate the parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$, where $\pi(i) = p(z_1 = i)$ is the initial state distribution, $A(i, j) = p(z_t = j | z_{t-1} = i)$ is the transition matrix, and $\boldsymbol{B}$ are the parameters of the class-conditional densities $p(x_t | z_t = j)$.
- Case1: $z_{1:T}$ is observed We can easily compute the MLEs for $\boldsymbol{\theta}$.
- Case2: $z_{1:T}$ is hidden We can estimate $\boldsymbol{\theta}$ with the EM (Baum-Welch) algorithm.

E step

- Given the sample $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$, the expected complete data log likelihood is

$$
\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = & \sum_{k=1}^{K} E[N_k^1] \log \pi_k + \sum_{j=1}^{K} \sum_{k=1}^{K} E[N_{jk}] \log A_{jk} \\
& + \sum_{i=1}^{N} \sum_{t=1}^{T_i} \sum_{k=1}^{K} p(z_t = k | \boldsymbol{x}_i, \boldsymbol{\theta}^{old}) \log p(x_{i,t|\phi_k}),
\end{aligned}
$$

E step

- $E[N_k^1] = \sum_{i=1}^{N} p(z_{i1} = k|\boldsymbol{x}_i, \boldsymbol{\theta}^{old}) = \sum_{i=1}^{N} \gamma_{i,k}(1)$
  (the number in state k at time 1)
- $E[N_{jk}] = \sum_{i=1}^{N} \sum_{t=2}^{T_i} p(z_{i,t-1} = j, z_{i,t} = k|\boldsymbol{x}_i, \boldsymbol{\theta}^{old}) = \sum_{i=1}^{N} \sum_{t=2}^{T_i} \xi_{i,j,k}(t)$
  (the number of transitions from state j to state k)
- $E[N_j] = \sum_{i=1}^{N} \sum_{t=1}^{T_i} p(z_{i,t} = j|\boldsymbol{x}_i, \boldsymbol{\theta}^{old}) = \sum_{i=1}^{N} \sum_{t=1}^{T_i} \gamma_{i,t}(j)$
  (the total number of times in state j)

M step

- We have that the M step for $\boldsymbol{A}$ and $\boldsymbol{\pi}$ is to just normalize the expected number:

$$\hat{A}_{jk} = \frac{E[N_{jk}]}{\sum_{k'} E[N_{jk'}]}, \hat{\pi}_k \frac{E[N_k^1]}{N}.$$

- $\hat{B}_{jl} = \frac{E[M_{jl}]}{E[N_j]}$, where $E[M_{jl}] = \sum_{i=1}^{N} \sum_{t=1}^{T_i} \gamma_{i,t}(j) I(x_{i,t} = l)$.

Initialization

- Use some fully labeled data to initialize the parameters.
- Initially ignore the Markov dependencies, and estimate the observation parameters using the standard mixture model estimation methods, such as K-means or EM.
- Randomly initialize the parameters, use multiple restarts, and pick the best solution.

Model selection

- Two main issues: how many states, and what topology to use for the state transition diagram.
- Choosing the number of hidden states
  (i)Use grid-search over a range of K's, using as an objective function cross-validated likelihood, the BIC score, or a variational lower bound to the log-marginal likelihood.
  (ii)Use reversible jump MCMC.
  (iii)Use variational Bayes to "extinguish" unwanted components.
  (iv)Use an "infinite HMM", which is based on the hierarchical Dirichlet process.

Model selection

- Structure learning
- To learn the structure of the state transition diagram, not the structure of the graphical model (which is fixed).
- Alternatively, one can pose the problem as MAP estimation using a minimum entropy prior.