STAT 516: Stochastic Modeling of Scientific Data **Autumn 2024**

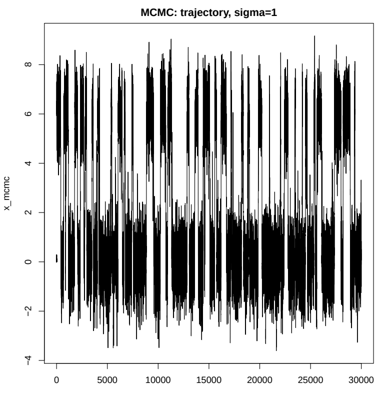
Lecture 9: Hidden Markov Model

Instructor: Yen-Chi Chen

These notes are partially based on those of Mathias Drton.

9.1 Introduction

Hidden Markov Model (HMM) is a powerful tool to model a time series with multiple *patterns*. For a concrete example, consider the trace plot of an MCMC for a 2-Gaussian mixture model:



There seems to be two patterns in the plot–one centered at 0 and the other centered at 6. The trajectory is oscillating between the two centers. The HMM is a powerful tool to model a data like this.

The HMM consists of two variables, the observed variable Y_t and a hidden state variable X_t. The variables $\{Y_1, \cdots, Y_n\}$ are what we observed whereas the hidden variables $\{X_1, \cdots, X_n\}$ are unobserved states at each time point. In an HMM, the joint distribution of the observations $Y_1, \ldots, Y_n$ and the hidden states $X_1, \ldots, X_n$ factors according to the graph:

$$\begin{tikzcd}
X_1 \arrow[r, no head] \arrow[d, no head] & X_2 \arrow[r, no head] \arrow[d, no head] & X_3 \arrow[r, no head] \arrow[d, no head] & X_4 \arrow[r, dashed, no head] \arrow[d, no head] & X_n \arrow[d, no head] \\
Y_1 & Y_2 & Y_3 & Y_4 & Y_n
\end{tikzcd}$$

Namely, the observations Y_t's are dependent on each other because of the hidden variables. This implies that conditioned on the hidden states, observed variables Y_t's are independent. Moreover, the hidden states form a homogeneous Markov chain. Here we will assume that the number of states $S = \{1, 2, \cdots, s\}$ is finite and we will also assume that the observed variables are categorical/discrete $Y_t \in M = \{1, \cdots, m\}$.

The HMM consists of 3 sets of parameters:

- *Initial distribution* (of hidden state): $\nu = (\nu(1), \cdots, \nu(s))$.

- *Transition probability*: $\mathbf{P} = \{p_{ij}\}, i, j = 1, \cdots, s$.
- *Emission probability*: $\mathbf{E} = \{e(k|i)\}, i = 1, \cdots, s$ and $k = 1, \cdots, m$.

The first two parameters are the typical parameters of a Markov chain and the last one, the emission probability, describes how a hidden state is associated with the observed variables.

Let $\mathbf{y}_{-t} = (y_1, \cdots, y_{t-1}, y_{t+1}, \cdots, y_n)$ and $\mathbf{y}_{a:b} = (y_a, y_{a+1}, \cdots, y_b)$ and $\mathbf{y} = (y_1, \cdots, y_n)$. According to the graphical model, the variables of an HMM is characterizes by the parameters as

$$\begin{aligned} P(x_1, \cdots, x_n) &= \nu(x_1) \prod_{t=2}^{n} P(x_t|x_{t-1}) = \nu(x_1) \prod_{t=2}^{n} p_{x_{t-1}, x_t} \\ P(y_t|\mathbf{y}_{-t}, \mathbf{x}) &= P(y_t|x_t) = e(y_t|x_t). \end{aligned}$$

Beware, although the hidden variables $X_1, \cdots, X_n$ are Markov chain, the observed variables $Y_1, \cdots, Y_n$ may not be.

There are four common goals of an HMM:

- *Likelihood Evaluation:* we want to rapidly evaluate the likelihood value $L(\nu, \mathbf{P}, \mathbf{E}|\mathbf{y}) = P(\mathbf{y})$.
- *Parameter Estimation:* we want to estimate the underlying parameters $\nu, \mathbf{P}, \mathbf{E}$.
- *Hidden State Inference:* given $\mathbf{y}$ and parameters $\nu, \mathbf{P}, \mathbf{E}$, we want to reconstruct the corresponding hidden states $\mathbf{x}$.
- *Forecasting:* we want to predict the future outcomes.

9.2 Likelihood Evaluation: Forward and Backward Algorithm

9.2.1 Backward algorithm

Assume that we observed $\mathbf{Y} = \mathbf{y}$. The likelihood function is the joint probability of $\mathbf{y}$, so

$$\begin{aligned} L(\nu, \mathbf{P}, \mathbf{E}|\mathbf{y}) = P(\mathbf{y}) &= \sum_{\mathbf{x}} P(\mathbf{y}, \mathbf{x}) \\ &= \sum_{\mathbf{x}} P(\mathbf{y}|\mathbf{x}) P(\mathbf{x}) \\ &= \sum_{\mathbf{x}} \prod_{t=1}^{n} P(y_t|\mathbf{y}_{1:(t-1)}, \mathbf{x}) \nu(x_1) \prod_{r=2}^{n} P(x_r|\mathbf{x}_{1:(r-1)}) \\ &= \sum_{\mathbf{x}} \nu(x_1) \prod_{t=1}^{n} e(y_t|x_t) \prod_{r=2}^{n} p(x_r|x_{r-1}) \\ &= \sum_{\mathbf{x}} \nu(x_1) e(y_1|x_1) \prod_{t=2}^{n} e(y_t|x_t) p(x_t|x_{t-1}). \end{aligned}$$

Note that because each x_t has s possible states, computing the summation here requires an evaluation of s^n elements, which is very large. Thus, we need to find a smart way to bypass this summation problem.

Here we introduce the backward algorithm, which will drastically reduce the computational cost. To illustrate the idea, we consider the case $n = 3$. The likelihood function will be

$$
\begin{aligned}
L(\nu, \mathbf{P}, \mathbf{E}|\mathbf{y}) &= \sum_{x_1,x_2,x_3} \nu(x_1)e(y_1|x_1)e(y_2|x_2)p(x_2|x_1)e(y_3|x_3)p(x_3|x_2) \\
&= \underbrace{\sum_{x_1} \nu(x_1)e(y_1|x_1) \underbrace{\left\{ \sum_{x_2} e(y_2|x_2)p(x_2|x_1) \underbrace{\left\{ \sum_{x_3} e(y_3|x_3)p(x_3|x_2) \right\}}_{b_2(x_2)} \right\}}_{b_1(x_1)}}_{b_0}
\end{aligned}
$$

What do we gain by doing this? Naively summing over all states requires an $O(ns^n)$ operations. But recursively evaluating $b_t(1), \cdots, b_t(s)$ is a lot cheaper using dynamic programming; the total cost is $O(ns^2)$.

Now we take a close look at each $b_t(x_t)$.

$$
b_2(x_2) = \sum_{x_3} e(y_3|x_3)p(x_3|x_2) = \sum_{x_3} P(y_3|x_3)P(x_3|x_2) = P(y_3|x_2).
$$

And

$$
\begin{aligned}
b_1(x_1) &= \sum_{x_2} e(y_2|x_2)p(x_2|x_1)b_2(x_2) \\
&= \sum_{x_2} P(y_2|x_2)P(x_2|x_1)P(y_3|x_2) \\
&= \sum_{x_2} P(y_3, y_2, x_2|x_1) \\
&= P(y_3, y_2|x_1).
\end{aligned}
$$

You can generalize all the above derivation to more variable case. The general form of $b_t(i)$ is

$$
b_t(i) = \sum_{j=1}^{s} p(j|i)e(y_{t+1}|j)b_{t+1}(j). \tag{9.1}
$$

and it represents the **backward probability**:

$$
b_t(i) = P(\mathbf{y}_{(t+1):n}|X_t = i) \tag{9.2}
$$

and $b_0 = P(\mathbf{y}) = L(\nu, \mathbf{E}, \mathbf{P}|\mathbf{y})$ is the likelihood value.

The **Backward algorithm** works as follows:

1. Let $b_n(i) = 1$ for all $i \in S$.
2. For $t = n-1, \cdots, 1$, compute
$$
b_t(i) = \sum_{j=1}^{s} p(j|i)e(y_{t+1}|j)b_{t+1}(j)
$$
for $i \in S$.

3. Finally, compute

$$b_0 = \sum_{j=1}^{s} \nu(j)e(y_1|j)b_1(j) = L(\nu, \mathbf{E}, \mathbf{P}|\mathbf{y}).$$

It is called the backward algorithm becomes we are taking a backward sweep from $t = n$ to $t = 0$.

Here is derivation of the formula in equation (9.1) from equation (9.2). We assume that $b_t(i) = P(\mathbf{y}_{(t+1):n}|X_t = i)$. Then

$$\begin{aligned}
b_t(i) &= P(\mathbf{y}_{(t+1):n}|X_t = i)\\
&= \sum_j P(y_{t+1}, X_{t+1} = j, \mathbf{y}_{(t+2):n}|X_t = i)\\
&= \sum_j P(y_{t+1}|X_{t+1} = j, \mathbf{y}_{(t+2):n}, X_t = i)P(\mathbf{y}_{(t+2):n}|X_{t+1} = j, X_t = i)P(X_{t+1} = j|X_t = i)\\
&= \text{using the graphical model}\\
&= \sum_j P(y_{t+1}|X_{t+1} = j)P(\mathbf{y}_{(t+2):n}|X_{t+1} = j)P(X_{t+1} = j|X_t = i)\\
&= \sum_j e(y_{t+1}|j)b_{t+1}(j)p(j|i)\\
&= (9.1).
\end{aligned}$$

9.2.2 Forward algorithm

The forward algorithm relies on the idea of *forward probability*. To see how this idea works, we consider again our $n = 3$ case but now we rewrite the probability as

$$\begin{aligned}
L(\nu, \mathbf{P}, \mathbf{E}|\mathbf{y}) &= \sum_{x_1,x_2,x_3} \nu(x_1)e(y_1|x_1)e(y_2|x_2)p(x_2|x_1)e(y_3|x_3)p(x_3|x_2)\\
&= \sum_{x_1,x_2,x_3} e(y_3|x_3)p(x_3|x_2)e(y_2|x_2)p(x_2|x_1)\nu(x_1)e(y_1|x_1)\\
&= \sum_{x_3} e(y_3|x_3) \underbrace{\sum_{x_2} p(x_3|x_2)\, \underbrace{e(y_2|x_2) \sum_{x_1} p(x_2|x_1)\, \underbrace{\nu(x_1)e(y_1|x_1)}_{a_1(x_1)}}_{a_2(x_2)}}_{a_3(x_3)}
\end{aligned}$$

The quantity $a_t(i)$ is called the **forward probability**. Using a similar derivation as the backward probability, you can show that

$$a_t(i) = P(\mathbf{y}_{1:t}, x_t = i). \tag{9.3}$$

The forward algorithm is a method based on the forward probability to evaluate the likelihood function. It works as follows:

1. Initialize $a_1(i) = \nu(i)e(y_1|i)$ for $i \in S$.
2. For $t = 1, \cdots, n-1$, compute

$$a_{t+1}(i) = e(y_{t+1}|i) \left\{ \sum_{j=1}^{s} p(i|j)a_t(j) \right\} \tag{9.4}$$

for $i \in S$.

3. Finally, we compute
$$\sum_{j=1}^{s} a_n(j) = \sum_{j=1}^{s} P(\mathbf{y}, x_n = j) = P(\mathbf{y}) = L(\nu, \mathbf{E}, \mathbf{P}|\mathbf{y}).$$

Here, we are sweeping from $t = 1$ to n so it is called the forward algorithm.

9.3 Parameter Estimation and the Baum-Welch algorithm

To infer $\theta = (\nu, \mathbf{P}, \mathbf{E})$, a simple approach is the MLE. However, there is no closed form solution to the MLE because we have hidden variables $\mathbf{X}$. Thus, we need to design an algorithm to find the MLE. As is mentioned in the previous lecture, we will use the EM algorithm in this case.

The EM algorithm for HMM is also called the **Baum-Welch algorithm**. The complete-data likelihood is
$$P(\mathbf{x}, \mathbf{y}; \theta) = \nu(x_1) \prod_{t=2}^{n} p(x_t|x_{t-1}) \prod_{r=1}^{n} e(y_r|x_r)$$
and the corresponding log-likelihood is
$$\log P(\mathbf{x}, \mathbf{y}; \theta) = \log \nu(x_1) + \sum_{t=2}^{n} \log p(x_t|x_{t-1}) + \sum_{r=1}^{n} \log e(y_r|x_r).$$

E-step:
To derive the E-step, we first analyze the expected log-likelihood value given a parameter $\theta^{(k)} = (\nu^{(k)}, \mathbf{E}^{(k)}, \mathbf{P}^{(k)})$:
$$\begin{aligned}
E\left\{\log P(\mathbf{x}, \mathbf{y}; \theta)|\mathbf{y}; \theta^{(k)}\right\} &= \sum_{\mathbf{x}} \log P(\mathbf{x}, \mathbf{y}; \theta) P(\mathbf{x}|\mathbf{y}; \theta^{(k)}) \\
&= \underbrace{\sum_{\mathbf{x}} \log \nu(x_1) P(\mathbf{x}|\mathbf{y}; \theta^{(k)})}_{F_1(\nu; \theta^{(k)})} + \underbrace{\sum_{\mathbf{x}} \sum_{t=2}^{n} \log p(x_t|x_{t-1}) P(\mathbf{x}|\mathbf{y}; \theta^{(k)})}_{F_2(\mathbf{P}; \theta^{(k)})} \\
&+ \underbrace{\sum_{\mathbf{x}} \sum_{t=1}^{n} \log e(y_t|x_t) P(\mathbf{x}|\mathbf{y}; \theta^{(k)})}_{F_3(\mathbf{E}; \theta^{(k)})}.
\end{aligned}$$

The additive form makes the M-step a lot easier–we can separately maximize each term.

M-step:
Updating ν:
$$\begin{aligned}
F_1(\nu; \theta^{(k)}) &= \sum_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}; \theta^{(k)}) \log \nu(x_1) \\
&= \sum_{x_1} P(x_1|\mathbf{y}; \theta^{(k)}) \log \nu(x_1) \\
&= \sum_{i} \gamma_1(i; \theta^{(k)}) \log \nu(i),
\end{aligned}$$

where

$$\gamma_t(i;\theta^{(k)}) = P(x_t = i|\mathbf{y};\theta^{(k)}) = \frac{a_t(i;\theta^{(k)})b_t(i;\theta^{(k)})}{\sum_j a_t(j;\theta^{(k)})b_t(j;\theta^{(k)})}. \tag{9.5}$$

Note that $a_t(j;\theta)$ and $b_t(j;\theta)$ are the forward and backward probability assuming θ is the generating probability.

Using Lagrange multiplier, one can show that the maximizer is

$$\nu^{(k+1)}(i) = \gamma_1(i;\theta^{(k)}) = \text{ imputed frequency of state } i \text{ at time 1.}$$

Updating $\mathbf{P}$*:*

$$\begin{aligned}
F_2(\mathbf{P};\theta^{(k)}) &= \sum_{\mathbf{x}}\sum_{t=2}^{n} \log p(x_t|x_{t-1}) P(\mathbf{x}|\mathbf{y};\theta^{(k)}) \\
&= \sum_{t=2}^{n}\sum_{x_t,x_{t-1}} \log p(x_t|x_{t-1}) \sum_{\mathbf{x}_{1:(t-2)},\mathbf{x}_{(t+1):n}} P(\mathbf{x}|\mathbf{y};\theta^{(k)}) \\
&= \sum_{t=2}^{n}\sum_{i,j} P(X_t = j, X_{t-1} = i|\mathbf{y};\theta^{(k)}) \log p(j|i) \\
&= \sum_{i,j}\left\{\sum_{t=2}^{n} P(X_t = j, X_{t-1} = i|\mathbf{y};\theta^{(k)})\right\} \log p(j|i) \\
&= \sum_{i,j} N_t(i,j;\theta^{(k)}) \log p(j|i),
\end{aligned}$$

where

$$N(i,j;\theta^{(k)}) = \sum_{t=2}^{n} P(X_t = j, X_{t-1} = i|\mathbf{y};\theta^{(k)}).$$

The maximization problem is like the maximization of the Markov chain with the number of observed transitions from state i to state j is $N(i,j;\theta^{(k)})$. Thus, we know that the maximizer will be

$$\begin{aligned}
p^{(k+1)}(j|i) &= \frac{N(i,j;\theta^{(k)})}{\sum_j N(i,j;\theta^{(k)})} \\
&= \frac{\text{expected number of transitions from } i \to j}{\text{expected number of transitions from } i}.
\end{aligned}$$

Note that there is an interesting relation between $N(i,j;\theta)$ and $\gamma_t(i;\theta)$:

$$\begin{aligned}
\sum_j N(i,j;\theta^{(k)}) &= \sum_j \sum_{t=2}^{n} P(X_t = j, X_{t-1} = i|\mathbf{y};\theta^{(k)}) \\
&= \sum_{t=2}^{n}\sum_j P(X_t = j, X_{t-1} = i|\mathbf{y};\theta^{(k)}) \\
&= \sum_{t=2}^{n} P(X_{t-1} = i|\mathbf{y};\theta^{(k)}) \\
&= \sum_{t=2}^{n} \gamma_{t-1}(i;\theta^{(k)}).
\end{aligned}$$

This observation also gives a hint on how to efficiently compute $N(i,j;\theta^{(k)}) = \sum_{t=2}^n P(X_t = j, X_{t-1} = i|\mathbf{y};\theta^{(k)})$:

$$\begin{aligned}
P(X_t = j, X_{t-1} = i|\mathbf{y};\theta^{(k)}) &\propto P(X_t = j, X_{t-1} = i, \mathbf{y};\theta^{(k)}) \\
&= P(\mathbf{y}_{(t+1):n}|X_t = j, X_{t-1} = i, \mathbf{y}_{1:t};\theta^{(k)})P(\mathbf{y}_{1:t}, X_t = j, X_{t-1} = i;\theta^{(k)}) \\
&= P(\mathbf{y}_{(t+1):n}|X_t = j;\theta^{(k)})P(y_t|\mathbf{y}_{1:(t-1)}, X_t = j, X_{t-1} = i;\theta^{(k)}) \\
&\quad \times P(X_t = j|X_{t-1} = i, \mathbf{y}_{1:(t-1)};\theta^{(k)})P(X_{t-1} = i, \mathbf{y}_{1:(t-1)};\theta^{(k)}) \\
&= P(\mathbf{y}_{(t+1):n}|X_t = j;\theta^{(k)})e(y_t|j;\theta^{(k)})P(j|i;\theta^{(k)})P(\mathbf{y}_{1:(t-1)}, X_{t-1} = i;\theta^{(k)}) \\
&= b_t(j;\theta^{(k)})e^{(k)}(y_t|j)p^{(k)}(j|i)a_{t-1}(i;\theta^{(k)}).
\end{aligned}$$

Thus,

$$P(X_t = j, X_{t-1} = i|\mathbf{y};\theta^{(k)}) = \frac{b_t(j;\theta^{(k)})e^{(k)}(y_t|j)p^{(k)}(j|i)a_{t-1}(i;\theta^{(k)})}{\sum_{i',j'} b_t(j';\theta^{(k)})e^{(k)}(y_t|j')p^{(k)}(j'|i')a_{t-1}(i';\theta^{(k)})}.$$

We can compute this probability easily and then sum over t to obtain $N(i,j;\theta^{(k)})$ and update $p^{(k+1)}(j|i)$.

Updating $\mathbf{E}$*:*

$$\begin{aligned}
F_3(\mathbf{E};\theta^{(k)}) &= \sum_{\mathbf{x}}\sum_{t=1}^n \log e(y_t|x_t)P(\mathbf{x}|\mathbf{y};\theta^{(k)}) \\
&= \sum_{t=1}^n\sum_{x_t} \log e(y_t|x_t)\sum_{\mathbf{x}_{-t}} P(\mathbf{x}|\mathbf{y};\theta^{(k)}) \\
&= \sum_{t=1}^n\sum_{x_t} P(x_t|\mathbf{y};\theta^{(k)})\log e(y_t|x_t) \\
&= \sum_{x_t}\sum_{t=1}^n \gamma_t(x_t;\theta^{(k)})\log e(y_t|x_t).
\end{aligned}$$

Again, using Lagrange multiplier, the maximizer is

$$e^{(k+1)}(\ell|i) = \frac{\sum_{t=1}^n \gamma_t(i;\theta^{(k)})I(y_t = \ell)}{\sum_{t=1}^n \gamma_t(i;\theta^{(k)})}.$$

With all the above updating equation, the Baum-Welch algorithm is

1. Start with $\theta^{(0)} = (\nu^{(0)}, \mathbf{P}^{(0)}, \mathbf{E}^{(0)})$. Repeat the following steps until some convergence criterion is met:
   (a) Compute the forward and backward probabilities $\{a_t(i|\theta^{(k)})\}$ and $\{b_t(i|\theta^{(k)})\}$.
   (b) Update HMM parameters

$$\begin{aligned}
\nu^{(k+1)}(i) &= \gamma_1(i;\theta^{(k)}) \\
p^{(k+1)}(j|i) &= \frac{N(i,j;\theta^{(k)})}{\sum_j N(i,j;\theta^{(k)})} \\
e^{(k+1)}(\ell|i) &= \frac{\sum_{t=1}^n \gamma_t(i;\theta^{(k)})I(y_t = \ell)}{\sum_{t=1}^n \gamma_t(i;\theta^{(k)})}.
\end{aligned}$$

9.4 Hidden State Reconstruction and Viterbi Algorithm

Note that here we assume that the parameters $\nu, \mathbf{P}, \mathbf{E}$ are given so we can easily evaluate the forward and the backward probabilities. We can estimate these parameters using the BW algorithm.

The forward and backward probabilities together imply a very interesting result:

$$\begin{aligned} P(X_t = i|\mathbf{y}) &= \frac{P(X_t = i, \mathbf{y})}{P(\mathbf{y})} \\ &= \frac{P(\mathbf{y}_{1:t}, \mathbf{y}_{(t+1):n}, X_t = i)}{P(\mathbf{y})} \\ &= \frac{P(\mathbf{y}_{(t+1):n}|\mathbf{y}_{1:t}, X_t = i)P(\mathbf{y}_{1:t}, X_t = i)}{P(\mathbf{y})} \\ &= \text{using graphical model} \\ &= \frac{P(\mathbf{y}_{(t+1):n}|X_t = i)P(\mathbf{y}_{1:t}, X_t = i)}{P(\mathbf{y})} \\ &= \frac{a_t(i)b_t(i)}{\sum_{j=1}^{s} a_t(j)b_t(j)}. \end{aligned}$$

Namely,

$$P(X_t = i|\mathbf{y}) \propto a_t(i)b_t(i)$$

so this gives us the marginal probability of a hidden state at time t. Thus, a method to reconstruct the hidden state is

$$\widehat{x}_t = \mathsf{argmax}_i a_t(i)b_t(i)$$

and by applying this to every t, we obtain

$$\widehat{x}_1, \cdots, \widehat{x}_n.$$

However, this idea is based on the *marginal* probability, not the joint probability. So it may give us a path such that $P(\widehat{x}_{t+1}|\widehat{x}_t) = 0$!

A better way is to find the states according to the joint distribution function. Namely,

$$\mathbf{x}^* = \mathsf{argmax}_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}) = \mathsf{argmax}_{\mathbf{x}} P(\mathbf{x}, \mathbf{y}).$$

In this case, $\mathbf{x}^*$ is the MAP estimator. Note that $\mathbf{x}^*$ involves n elements so there are totally s^n possible elements we need to search to obtain $\mathbf{x}^*$. Ideally, we want to avoid searching all possible configurations. Here we will use the idea that for a bivariate function $f(i, j)$,

$$\max_{i,j} f(i, j) = \max_j \{\max_i f(i, j)\}.$$

The joint distribution function up to time point $t + 1$ can be factored as

$$\begin{aligned} P(\mathbf{x}_{1:(t+1)}, \mathbf{y}_{1:(t+1)}) &= P(x_{t+1}, y_{t+1}|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})P(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) \\ &= P(y_{t+1}|x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})P(x_{t+1}|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})P(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) \\ &= \text{using graphical model} \\ &= P(y_{t+1}|x_{t+1})P(x_{t+1}|x_t)P(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) \\ &= e(y_{t+1}|x_{t+1})P(x_{t+1}|x_t)P(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}). \end{aligned}$$

This implies a very interesting recursive relation. Define

$$d_t(i) = \max_{\mathbf{x}_{1:(t-1)}} P(\mathbf{x}_{1:(t-1)}, x_t = i, \mathbf{y}_{1:t})$$

for $i \in S$. Then for $t = 2, \cdots, n$ and $i = 1, \cdots, s$,

$$\begin{aligned}
d_t(i) &= \max_{\mathbf{x}_{1:(t-1)}} P(\mathbf{x}_{1:(t-1)}, x_t = i, \mathbf{y}_{1:t}) \\
&= \max_{\mathbf{x}_{1:(t-1)}} e(y_t|x_t = i)P(x_t = i|x_{t-1})P(\mathbf{x}_{1:(t-1)}, \mathbf{y}_{1:(t-1)}) \\
&= e(y_t|x_t = i) \max_{\mathbf{x}_{1:(t-1)}} P(x_t = i|x_{t-1})P(\mathbf{x}_{1:(t-1)}, \mathbf{y}_{1:(t-1)}) \\
&= e(y_t|x_t = i) \max_{x_{t-1}} P(x_t = i|x_{t-1}) \underbrace{\max_{\mathbf{x}_{1:(t-2)}} P(\mathbf{x}_{1:(t-1)}, \mathbf{y}_{1:(t-1)})}_{d_{t-1}(x_{t-1})} \\
&= e(y_t|x_t = i) \left\{ \max_{x_{t-1}} P(x_t = i|x_{t-1}) d_{t-1}(x_{t-1}) \right\}.
\end{aligned}$$

Again, we obtain a recursive relation between each $d_t(i)$.

This suggests a recursive approach to find the joint maximizer, which is known as the **Viterbi algorithm**:

1. Compute $d_1(i) = \nu(i)e(y_1|i)$ for each $i \in S$.
2. For $t = 2, \cdots, n$, compute
$$\begin{aligned}
d_t(i) &= e(y_t|i) \cdot \max_j d_{t-1}(j)p(i|j) \quad \text{for all } i \in S \\
f_t(i) &= \mathsf{argmax}_j d_{t-1}(j)p(i|j) \quad \text{for all } i \in S.
\end{aligned}$$
3. Let $p^* = \max_j d_n(j)$ and $x_n^* = \mathsf{argmax}_j d_n(j)$.
4. (backtracking) For $t = n-1, \cdots, 1$:
$$x_t^* = f_{t+1}(x_{t+1}^*).$$
5. Return $\mathbf{x}^* = (x_1^*, \cdots, x_n^*)$.

The logic of Viterbi algorithm is that for a function $g(i,j)$

$$(i^*, j^*) = \mathsf{argmax}_{i,j} g(i,j) \Rightarrow j^* = \mathsf{argmax} \left\{ \max_i g(i,j) \right\}.$$

It is clear that $x_n^* = \mathsf{argmax}_j d_n(j)$ is indeed the maximizer at the last time point. To see that x_t^* indeed maximizes the joint probability, we consider the case of $n-1$ (and you can generalize it to any t using induction). Given x_n^*, the true maximizer of x_{n-1} should be

$$\begin{aligned}
x_{n-1}^* &= \mathsf{argmax}_j \max_{\mathbf{x}_{1:(n-2)}} P(\mathbf{x}_{1:n-2}, x_{n-1} = j, x_n = x_n^*, \mathbf{y}) \\
&= \mathsf{argmax}_j \max_{\mathbf{x}_{1:(n-2)}} e(y_n|x_n = x_n^*)P(x_n = x_n^*|x_{n-1} = j)P(\mathbf{x}_{1:(n-2)}, x_{n-1} = j, \mathbf{y}_{1:(n-1)}) \\
&= \mathsf{argmax}_j P(x_n = x_n^*|x_{n-1} = j) \max_{\mathbf{x}_{1:(n-2)}} P(\mathbf{x}_{1:(n-2)}, x_{n-1} = j, \mathbf{y}_{1:(n-1)}) \\
&= \mathsf{argmax}_j P(x_n = x_n^*|x_{n-1} = j) d_{n-1}(j) \\
&= f_n(x_n^*).
\end{aligned}$$

Thus, indeed the Viterbi algorithm finds the path that maximizes the joint probability.

Note that when n is large, the rounding error of computing the probability could be a problem. To avoid this problem, one can move all calculation to the log scale and use

$$\tilde{d}_t(i) = \max_{\mathbf{x}_{1:(t-1)}} \log P(\mathbf{x}_{1:(t-1)}, x_t = i, \mathbf{y}_{1:t}).$$

We will obtain another recursive relation:

$$\tilde{d}_t(i) = \log e(y_t|i) + \max_j \left\{ \tilde{d}_{t-1}(j) + \log p(i|j) \right\}.$$

9.5 Forecasting

The forecasting problem is that case where we have observations up to time point n and we would like to predict the possible outcomes at time point $n+h$ for $h = 1, 2, \cdots$. Specifically, we want to obtain a forecast probability of each possible outcome $y \in M$. Using the notion of probability, what we really want is $P(Y_{n+1} = y_{n+1}|\mathbf{y})$ for each $y_{n+1} \in M$.

We start with the simple case $h = 1$.

$$\begin{aligned}
P(Y_{n+1} = y_{n+1}|\mathbf{y}) &= \sum_i P(Y_{n+1} = y_{n+1}, X_n = i|\mathbf{y}) \\
&= \sum_i P(Y_{n+1} = y_{n+1}|X_n = i)P(X_n = i|\mathbf{y}) \\
&= \sum_i \left\{ \sum_j P(Y_{n+1} = y_{n+1}, X_{n+1} = j|X_n = i) \right\} P(X_n = i|\mathbf{y}) \\
&= \sum_i \left\{ \sum_j P(Y_{n+1} = y_{n+1}|X_{n+1} = j)p(j|i) \right\} P(X_n = i|\mathbf{y}).
\end{aligned}$$

Recall that $P(X_n = i|\mathbf{y}) = \frac{b_n(i)a_n(i)}{\sum_k b_n(k)a_n(k)} = \frac{a_n(i)}{\sum_k a_n(k)}$ since $b_n(k) = 1$ for all k. Using this fact, we can rewrite the above as

$$\begin{aligned}
P(Y_{n+1} = y_{n+1}|\mathbf{y}) &= \sum_i \left\{ \sum_j P(Y_{n+1} = y_{n+1}|X_{n+1} = j)P(j|i) \right\} P(X_n = i|\mathbf{y}) \\
&= \frac{\sum_i \left\{ \sum_j e(y_{n+1}|j)p(j|i) \right\} a_n(i)}{\sum_k a_n(k)}.
\end{aligned}$$

With an estimate of e and $P(j|i)$ and the corresponding forward probability, we can easily compute the forecast probability of each state $y_{n+1} \in M$.

For a general h case (forecasting h-time point future), we have

$$\begin{aligned}
P(Y_{n+h} = y_{n+h}|\mathbf{y}) &= \sum_i P(Y_{n+h} = y_{n+h}, X_n = i|\mathbf{y}) \\
&= \sum_i \left\{ \sum_j P(Y_{n+h} = y_{n+h}, X_{n+h} = j|X_n = i, \mathbf{y}) \right\} P(X_n = i|\mathbf{y}) \\
&= \sum_i \left\{ \sum_j P(Y_{n+h} = y_{n+h}|X_{n+h} = j) P(X_{n+h} = j|X_n = i) \right\} P(X_n = i|\mathbf{y}) \\
&= \sum_i \left\{ \sum_j e(y_{n+h}|j) p^{(h)}(j|i) \right\} P(X_n = i|\mathbf{y}) \\
&= \frac{\sum_i \left\{ \sum_j e(y_{n+h}|j) p^{(h)}(j|i) \right\} a_n(i)}{\sum_k a_n(k)},
\end{aligned}$$

where $p^{(h)}(j|i)$ is the h-step transition probability of the Markov chain. Using this equation, we can make prediction about the likelihood of each outcome in the future.

9.6 Model Selection

In all of the above analysis, we assume that the total number of hidden states s is known. However, in reality, we often do not know this number so we need to choose it using some smart trick. The study of selecting the right amount of hidden states is part of a general problem called *model selection*.

An intuitive way is to consider various s and see which one gives the highest likelihood value. However, this idea will suffer the problem of *overfitting*–the more hidden states we fit to the data, the higher likelihood value we will obtain.

This overfitting problem is something we need to be very cautious about when analyzing the data. A common approach to model selection is to use a penalized criterion. For the HMM with s hidden states, let $\ell_n^*(s)$ be the log-likelihood function corresponding to the MLE of this model. We then select s by maximizing

$$\ell_n^*(s) - \Psi(s),$$

where $\Psi(s)$ is an increasing function with respect to s. We call $\Psi(s)$ the penalty or regularizer. In many cases, we frame the problem as the minimizer problem and choose s by minimizing

$$R(s) = -2\ell_n^*(s) + 2\Psi(s).$$

While there are many choices of $\Psi(s)$, two most popular ones are the Akaike information criterion (AIC) and Bayesian information criterion (BIC), which leads to

$$R_{\mathsf{AIC}}(s) = -2\ell_n^*(s) + 2p, \quad R_{\mathsf{BIC}}(s) = -2\ell_n^*(s) + p \log n,$$

where $p = p(s)$ is the total number of parameters. In an HMM,

$$p(s) = \underbrace{(s-1)}_{\sim\nu} + \underbrace{s(s-1)}_{\sim\mathbf{P}} + \underbrace{s(m-1)}_{\sim\mathbf{E}} = s^2 - 1 + s(m-1).$$

AIC has a smaller penalty compared to BIC so it often selects a model with more variables. AIC aims at selecting the model that is the best for prediction under certain criterion whereas BIC aims at finding the true model according to Bayesian's point of view.

9.6.1 Intuition of AIC and BIC

Here we use an informal way of deriving AIC and BIC that gives us some intuition on how they are constructed[1]. Note that the notations in this subsection is independent of the notations in the previous sections. To illustrate the idea, we assume that we observed $X_1, \cdots, X_n$ that are IID from an unknown PDF $p_0(x)$. And we consider different models

$$M_j = \{p(x;\theta_j) : \theta_j \in \Theta_j\}, \quad \mathsf{dim}(\Theta_j) = d_j$$

for $j = 1, \cdots, m$.

Intuition of AIC: Let $\widehat{p}_j(x) = p(x;\widehat{\theta}_j)$ with $\widehat{\theta}_j$ be the MLE under parameter space Θ_j. Recall that a common measure of closeness between a fitted model $\widehat{p}_j \in M_j$ and the true model is the KL divergence:

$$K(p_0, \widehat{p}_j) = \int p_0(x) \log\left(\frac{p_0(x)}{\widehat{p}_j(x)}\right) = -\int p_0(x)\log \widehat{p}_j(x) + \int p_0(x) \log p_0(x) dx.$$

Since the last term is independent of j, minimizing $K(p_0, \widehat{p}_j)$ across different j is equivalent to maximizing

$$K_j = \int p_0(x) \log \widehat{p}_j(x) = \int p_0(x) \log p(x; \widehat{\theta}_j).$$

AIC aims at choosing j that minimizing the KL divergence so it is equivalent to choosing j maximizing K_j.

However, K_j involves p_0 so we have to estimate it. A natural estimate of K_j is to use the data, leading to

$$\widehat{K}_j = \frac{1}{n}\sum_{i=1}^n \log p(X_i; \widehat{\theta}_j).$$

However, this method is a biased estimator since the data is used both in finding the MLE $\widehat{\theta}_j$ and evaluating $\widehat{K}_j$.

Akaike showed that the bias of $\widehat{K}_j$ is approximately $\frac{d_j}{n}$, which motivates us to use

$$\widehat{K}_j - \frac{d_j}{n}$$

as an estimate of K_j. Now, notice that the log-likelihood function $\ell_n(\theta) = \sum_{i=1}^n \log p(X_i;\theta) = n\widehat{K}_j$. So maximizing $\widehat{K}_j - \frac{d_j}{n}$ is equivalent to minimizing

$$-2n\widehat{K}_j + 2d_j = -2\ell_n(\widehat{\theta}_j) + 2d_j,$$

which is the AIC criterion.

Intuition of BIC: The original idea of BIC is from the posterior probability–we want to choose the model with a highest posterior probability. But it is easier to understand it using the Bayesian evidence (very

[1]If you are interested in more details, I would recommend reading http://www.stat.cmu.edu/~larry/=stat705/Lecture16.pdf

similar to the Bayes factor). Let $\pi(M_j)$ be the posterior probability of j-th model. Recall that in terms of Bayesian evidence, we want to choose the model that maximizes

$$p(X_1, \cdots, X_n|M_j).$$

Assume that for model j, we have a prior distribution π_j on the parameter $\theta_j \in \Theta_j$. Then the evidence can be written as

$$\begin{aligned} p(X_1, \cdots, X_n|M_j) &= \int p(X_1, \cdots, X_n|\theta_j)\pi_j(\theta_j)d\theta_j \\ &= \int L(\theta_j|X_1, \cdots, X_n)\pi_j(\theta_j)d\theta_j \\ &= \int e^{\sum_{i=1}^n \ell(\theta_j|X_i)}\pi_j(\theta_j)d\theta_j. \end{aligned}$$

When the likelihood model is smooth, the expected log-likelihood function $\mathbb{E}(\ell(\theta_j|X_1))$ is a quadratic function around its model. So when n is large, Tayler expansion gives us

$$\frac{1}{n}\sum_{i=1}^n \ell(\theta_j|X_i) \approx \frac{1}{n}\sum_{i=1}^n \ell(\widehat{\theta}_j|X_i) + \frac{1}{2}(\theta_j - \widehat{\theta}_j)^T \left(\frac{1}{n}\sum_{i=1}^n \ell''(\widehat{\theta}_j|X_i)\right)(\theta_j - \widehat{\theta}_j).$$

Also,

$$\sum_{i=1}^n \ell''(\widehat{\theta}_j|X_i) \approx nH(\widehat{\theta}_j),$$

where $H(\theta) = \mathbb{E}(\ell''(\theta|X_i))$. Thus, putting these into exponent, we obtain

$$e^{\sum_{i=1}^n \ell(\theta_j|X_i)} \approx e^{\sum_{i=1}^n \ell(\widehat{\theta}_j|X_i) + \frac{1}{2}(\theta_j - \widehat{\theta}_j)^T nH(\widehat{\theta}_j)(\theta_j - \widehat{\theta}_j)}.$$

Therefore, the logarithmic of the Bayesian evidence (taking log will not affect the maximizer) will be

$$\begin{aligned} \log p(X_1, \cdots, X_n|M_j) &\approx \log \int e^{\sum_{i=1}^n \ell(\widehat{\theta}_j|X_i) + \frac{1}{2}(\theta_j - \widehat{\theta}_j)^T nH(\widehat{\theta}_j)(\theta_j - \widehat{\theta}_j)}\pi_j(\theta_j)d\theta_j \\ &= \sum_{i=1}^n \ell(\widehat{\theta}_j|X_i) + \log \int e^{\frac{1}{2}(\theta_j - \widehat{\theta}_j)^T nH(\widehat{\theta}_j)(\theta_j - \widehat{\theta}_j)}\pi_j(\theta_j)d\theta_j \\ &\approx \sum_{i=1}^n \ell(\widehat{\theta}_j|X_i) + \log\left(\sqrt{(2\pi/n)^p \mathsf{det}(H^{-1}(\widehat{\theta}_j))}\right) \\ &\approx \sum_{i=1}^n \ell(\widehat{\theta}_j|X_i) - \frac{p}{2}\log n \end{aligned}$$

which is what the BIC criterion is about.