Lecture topics:

- Hidden Markov Models (cont'd)

Hidden Markov Models (cont'd)

We will continue here with the three problems outlined previously. Consider having given a set of sequences of observations $y_1, \ldots, y_n$. The observations typically do not contain the hidden state sequence and we are left with the following problems to solve:

1. Evaluate the probability of observed data or
$$P(y_1, \ldots, y_n) = \sum_{x_1,\ldots,x_n} P(x_1, \ldots, x_n, y_1, \ldots, y_n) \quad (1)$$

2. Find the most likely hidden state sequence $x_1^*, \ldots, x_n^*$ given observations $y_1, \ldots, y_n$, i.e.,
$$\{x_1^*, \ldots, x_n^*\} = \arg \max_{x_1,\ldots,x_n} P(x_1, \ldots, x_n, y_1, \ldots, y_n) \quad (2)$$

3. Estimate the parameters of the model from multiple sequences of $y_1^{(l)}, \ldots, y_{n_l}^{(l)}$, $l = 1, \ldots, L$.

We have already solved the first problem. For example, this can be done with the forward algorithm

$$q(j)P(y_1|j) = \alpha_1(j) \quad (3)$$
$$\left(\sum_{i=1}^{k} \alpha_{t-1}(i)P_{ij}\right) P(y_t|j) = \alpha_t(j) \quad (4)$$

where $\alpha_t(j) = P(y_1, \ldots, y_t, X(t) = j)$ so that $P(y_1, \ldots, y_n) = \sum_j \alpha_n(j)$.

Problem 2: most likely hidden state sequence

The most likely hidden state sequence can be found with a small modification to the forward pass algorithm. The goal is to first evaluate "max-probability" of data

$$\max_{x_1,\ldots,x_n} P(y_1, \ldots, y_n, x_1, \ldots, x_n) = P(y_1, \ldots, y_n, x_1^*, \ldots, x_n^*) \quad (5)$$

and subsequently reconstruct the maximizing sequence $x_1^*, \ldots, x_n^*$. The max operation is similar to evaluating

$$\sum_{x_1,\ldots,x_n} P(y_1, \ldots, y_n, x_1, \ldots, x_n) = P(y_1, \ldots, y_n) \tag{6}$$

which we were able to do with just the forward algorithm. In fact, we can obtain the max-probability of data by merely changing the 'sum' in the forward algorithm to a 'max':

$$q(j)P(y_1|j) = d_1(j) \tag{7}$$

$$\left(\max_i d_{t-1}(i)P_{ij}\right) P(y_t|j) = d_t(j) \tag{8}$$

where

$$d_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1, \ldots, y_t, x_1, \ldots, x_{t-1}, X(t) = j) \tag{9}$$

In the forward algorithm we finally summed over the last state j in $\alpha_n(j)$ to get the probability of observations. Analogously, here we need to maximize over that last state so as to get the max-probability:

$$\max_{x_1,\ldots,x_n} P(y_1, \ldots, y_n, x_1, \ldots, x_n) = \max_j d_n(j) \tag{10}$$

We now have the maximum value but not yet the maximizing sequence. We can easily reconstruct the sequence by backtracking search, i.e., by sequentially fixing states starting with the last one:

$$x_n^* = \arg\max_j d_n(j) \tag{11}$$

$$x_t = \arg\max_i d_t(i)P_{i,x_{t+1}^*} \tag{12}$$

In other words, the backward iteration simply finds i that attains the maximum in Eq.(8) when j has already been fixed to the maximizing value x_{t+1}^* for the next state. The resulting algorithm that evaluates $d_t(j)$ through the above recursive formula, and follows up with the backtracking search to realize (one of) the most likely hidden state sequences, is known as the *Viterbi algorithm*.

Consider an HMM with two underlying states and transition probabilities as described in Figure 1. Note that the model cannot return to state 1 after it has left it. Each state $j = 1, 2$ is associated with a Gaussian output distribution $P(y|j) = N(y; \mu_j, \sigma^2)$, where

$\mu_1 = 3$, $\mu_2 = 1$, and the variances are assumed to be the same. We are given 8 observations $y_1, \ldots, y_8$ shown in the figure. Now, for any specific value of σ^2, we can find the most likely hidden state sequence $x_1^*, \ldots, x_8^*$ with the Viterbi algorithm.

Let's try to understand how this state sequence behaves as a function of the common variance σ^2. When σ^2 is large, the two output distributions, $P(y|1)$ and $P(y|2)$, assign essentially the same probability to all the observations in the figure. Thus the most likely hidden state sequence is one that is guided solely by the Markov model (no observations). The resulting sequence is all 2's. The probability of this sequence under the Markov model is just 1/2 (there's only one choice, the initial selection). The probability of any other state sequence is at most 1/4. Now, let's consider the other extreme, when the variance σ^2 is very small. In this case, the state sequence is essentially only guided by the observations with the constraint that the you cannot transition out of state 2. The most likely state sequence in this case is five 1's followed by three 2's, i.e., 11111222. The two observations y_4 and y_5 keep the model in state 1 even though y_3 is low. This is because the Markov chain forces us to either capture y_3 or $\{y_4, y_5\}$ but not both. If the model could return to state 1, the most likely state sequence would become 11211222. For intermediate values of σ^2 the most likely state sequence tries to balance the tendency of the Markov chain to choose 2 as soon as possible and the need to assign a reasonable probability to all observations. For example, if $\sigma^2 \approx 1$ then the resulting most likely state sequence is 11222222.

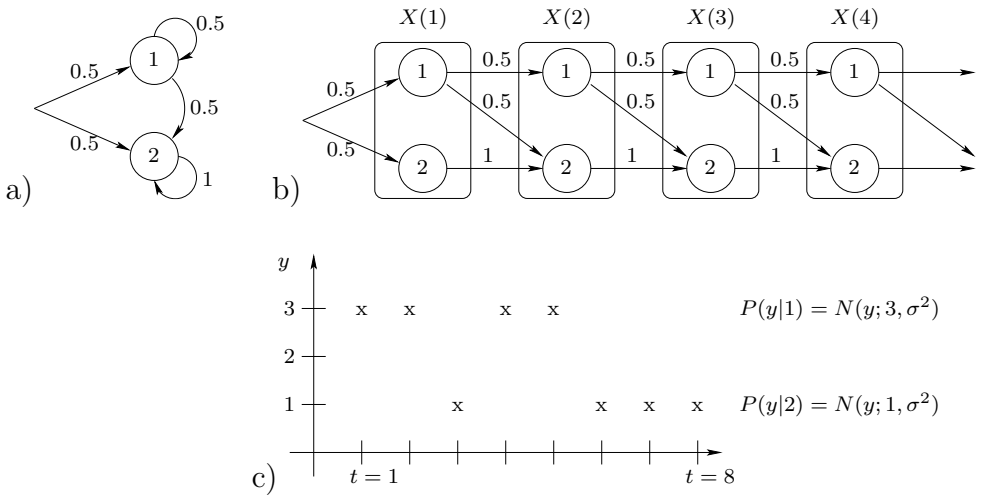

Figure 1: a) A two-state markov chain with transition probabilities, b) the same chain unfolded in time with the corresponding state variables, c) example observations over 8 time points and the output distributions.

Problem 3: estimation

In most cases we have to estimate HMMs only on the basis of output sequences such as $y_1, \ldots, y_n$ without knowing the corresponding states of the Markov chain. This is akin to mixture models discussed earlier. These are incomplete data estimation problems, i.e., we do not have observations for all the variables involved in the model. As before, the estimation can be performed iteratively via the EM algorithm.

A simple way to derive the EM algorithm is to start with complete observations, i.e., we assume we have $x_1, \ldots, x_n$ as well as $y_1, \ldots, y_n$. Note that while typically we would have multiple observation sequences, we will focus here on a single sequence to keep the equations simple. Now, we can encode the complete observations by defining

$$\delta(i|t) = \begin{cases} 1, & \text{if } x_t = i \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$\delta(i,j|t) = \begin{cases} 1, & \text{if } x_t = i \text{ and } x_{t+1} = j \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

The complete log-likelihood then becomes

$$\begin{aligned} l(\{x_t\},\{y_t\}) &= \sum_{i=1}^{k} \delta(i|1) \log q(i) + \sum_{i=1}^{k} \left(\sum_{t=1}^{n} \delta(i|t) \log P(y_t|i) \right) \\ &\quad + \sum_{i=1}^{k} \sum_{j=1}^{k} \left(\sum_{t=1}^{n} \delta(i,j|t) \right) \log P_{ij} \end{aligned} \tag{15}$$

where the first term simply picks out $\log q(x_1)$; in the second term, for each i, we consider all the observations that had to be generated from state i; in the last expression, the term in the brackets counts how many times each $i \to j$ transition occurred in the sequence $x_1, \ldots, x_n$. Given the counts $\delta(\cdot)$, we can now solve for the maximizing parameters as in the case of simple Markov chains (the first and the last expression), and as in mixture models (second expression).

The EM algorithm now follows directly from replacing the hard counts, $\delta(i|t)$ and $\delta(i,j|t)$, with the corresponding posterior probabilities $p(i|t)$ and $p(i,j|t)$, evaluated on the basis of the current HMM parameters. The posteriors we need are

$$p(i|t) = P(X(t) = i | y_1, \ldots, y_n) \tag{16}$$

$$p(i,j|t) = P(X(t) = i, X(t+1) = j | y_1, \ldots, y_n) \tag{17}$$

It remains to show how these posterior probabilities can be computed. We can start by writing

$$\begin{aligned} P(y_1, \ldots, y_n, X(t) = i) &= P(y_1, \ldots, y_t, X(t) = i) P(y_{t+1}, \ldots, y_n | X(t) = i) \quad (18) \\ &= \alpha_t(i)\beta_t(i) \quad (19) \end{aligned}$$

which follows directly from the Markov property (future observations do not depend on the past ones provided that we know which state we are in currently). The posterior is now obtained by normalization

$$P(X(t) = i | y_1, \ldots, y_n) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i'=1}^{k} \alpha_t(i')\beta_t(i')} \quad (20)$$

Similarly,

$$\begin{aligned} &P(y_1, \ldots, y_n, X(t) = i, X(t+1) = j) \\ &= P(y_1, \ldots, y_t, X(t) = i)\, P_{ij}\, P(y_{t+1}|j)\, P(y_{t+2}, \ldots, y_n | X(t+1) = j) \quad (21) \\ &= \alpha_t(i) P_{ij} P(y_{t+1}|j) \beta_{t+1}(j) \quad (22) \end{aligned}$$

The posterior again results from normalizing across i and j:

$$P(X(t) = i, X(t) = j | y_1, \ldots, y_n) = \frac{\alpha_t(i) P_{ij} P(y_{t+1}|j) \beta_{t+1}(j)}{\sum_{i'=1}^{k} \sum_{j'=1}^{k} \alpha_t(i') P_{i'j'} P(y_{t+1}|j') \beta_{t+1}(j')} \quad (23)$$

It is important to understand that $P(X(t) = i, X(t) = j | y_1, \ldots, y_n)$ are not the transition probabilities we have as parameters in the HMM model. These are posterior probabilities that a likely hidden state sequence that had to generate the observation sequence went through i at time t and transitioned into j; they are evaluated on the basis of the model and the observed sequence.

Multiple (partial) alignment

As another example of the use of the Viterbi algorithm as well as the EM algorithm for estimating HMM models, let's consider the problem of multiple alignment of sequences. Here we are interested in finding a pattern, a fairly conserved sequence of observations, embedded in unknown locations in multiple observed sequences. We assume we know very little about the pattern other than that it appeared once in all the sequences (a constraint we could easily relax further). For simplicity, we will assume here that we know the length of the pattern (four time points/positions). The sequences could be speech signals where a particular word was uttered in each but we don't know when the word appeared in

the signal, nor what exactly the word was. The sequences could also be protein or DNA sequences where we are looking for a sequence fragment that appears in all the available sequences (e.g., a binding site).

Perhaps the simplest possible HMM model we could specify is given in Figure 2. The states $m_1, \ldots, m_4$ are "match states" that we envision will be used to generate the pattern we are looking for. States I_1 and I_2 are "insert states" that generate the remaining parts of each observation sequence, before and after the pattern. Each state is associated with an output distribution: $P(y|I_i)$, $i = 1, 2$, $P(y|m_i)$, $i = 1, \ldots, 4$. The parameters p and the output distributions need to be learned from the available data.

Note that this is a model that generates a finite length sequence. We will first enter the insert state, spend there on average $1/(1-p)$ time steps, then generate the pattern, i.e., one observation in succession from each of the match states, and finally spend another $1/(1-p)$ time steps on average to generate flanking observations. We have specifically set the p parameter associated with the first insert state to agree with that of the second. This tying of parameters (balancing the cost of repeating insert states) ensures that, given any specific observation sequence, $y_1, \ldots, y_n$, there is no bias towards finding the pattern in any particular location of the sequence.

This model is useless for finding a pattern in a single observation sequence. However, it becomes more useful when we have multiple observation sequences that can all be assumed to contain the pattern. The stereotypical way that the pattern is generated, one observation from each successive match states, and the freedom to associate any observation with each of the match states, encourages the match states to take over the recurring pattern in the sequences (rather than being generated from the insert states). The output distributions for the insert states cannot become very specific since they will have to be used to generate most of the observations in the sequences.

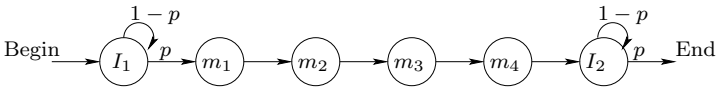

Figure 2: A simple HMM model for the multiple alignment task (only the Markov chain part shown).

Now, given multiple sequences, we can simply train the parameters in our HMM model via the EM algorithm to maximize the log-likelihood that the model assigns to those sequences. At this point we are not concerned about where the patterns actually occur, just interested in finding appropriate parameter values (output distributions). Similarly to mixture models

for clustering, the location of the pattern (and what the pattern is), is resolved in a soft manner through the posterior assignments of observations to match or insert states.

Once the parameters are found, we can use the Viterbi algorithm to "label" each observation in a sequence with the corresponding most likely hidden state. So, for example, for a particular observation sequence, $y_1, \ldots, y_n$, we might get

$$\begin{array}{cccccccccccc} I_1 & I_1 & \ldots & I_1 & m_1 & m_2 & m_3 & m_4 & I_2 & I_2 & \ldots & I_2 \\ y_1 & y_2 & \ldots & y_{t-1} & y_t & y_{t+1} & y_{t+2} & y_{t+3} & y_{t+4} & y_{t+5} & \ldots & y_n \end{array} \tag{24}$$

as the most likely hidden state sequence. The states in the most likely state sequence are in one to one correspondence with the observations. So, in this case, the pattern clearly occurs exactly at time/position t, where the sequence of match states begins.

The sequence fragments in all the observation sequences that were identified with the pattern can be subsequently aligned as in the figure below.