15 *Hidden Markov Models*

We relax the assumption that instances in a sample are independent and introduce Markov models to model input sequences as generated by a parametric random process. We discuss how this modeling is done as well as introduce an algorithm for learning the parameters of such a model from example sequences.

15.1 Introduction

UNTIL NOW, we assumed that the instances that constitute a sample are iid. This has the advantage that the likelihood of the sample is simply the product of the likelihoods of the individual instances. This assumption, however, is not valid in applications where successive instances are dependent. For example, in a word successive letters are dependent; in English 'h' is very likely to follow 't' but not 'x'. Such processes where there is a *sequence* of observations—for example, letters in a word, base pairs in a DNA sequence—cannot be modeled as simple probability distributions. A similar example is speech recognition where speech utterances are composed of speech primitives called phonemes; only certain sequences of phonemes are allowed, which are the words of the language. At a higher level, words can be written or spoken in certain sequences to form a sentence as defined by the syntactic and semantic rules of the language.

A sequence can be characterized as being generated by a *parametric random process*. In this chapter, we discuss how this modeling is done and also how the parameters of such a model can be learned from a training sample of example sequences.

15.2 Discrete Markov Processes

Consider a system that at any time is in one of a set of N distinct states: $S_1, S_2, \ldots, S_N$. The state at time t is denoted as $q_t, t = 1, 2, \ldots$, so, for example, $q_t = S_i$ means that at time t, the system is in state S_i. Though we write "time" as if this should be a temporal sequence, the methodology is valid for any sequencing, be it in time, space, position on the DNA string, and so forth.

At regularly spaced discrete times, the system moves to a state with a given probability, depending on the values of the previous states:

$$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, \cdots)$$

MARKOV MODEL

For the special case of a first-order *Markov model*, the state at time $t+1$ depends only on state at time t, regardless of the states in the previous times:

$$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, \cdots) = P(q_{t+1} = S_j | q_t = S_i) \tag{15.1}$$

This corresponds to saying that, given the present state, the future is independent of the past. This is just a mathematical version of the saying, Today is the first day of the rest of your life.

TRANSITION PROBABILITY

We further simplify the model—that is, regularize—by assuming that the *transition probability* from S_i to S_j is independent of time:

$$a_{ij} \equiv P(q_{t+1} = S_j | q_t = S_i) \tag{15.2}$$

satisfying

$$a_{ij} \geq 0 \text{ and } \sum_{j=1}^{N} a_{ij} = 1 \tag{15.3}$$

So, going from S_i to S_j has the same probability no matter when it happens, or where it happens in the observation sequence. $\mathbf{A} = [a_{ij}]$ is a $N \times N$ matrix whose rows sum to 1.

STOCHASTIC AUTOMATON

INITIAL PROBABILITY

This can be seen as a *stochastic automaton* (see figure 15.1). From each state S_i, the system moves to state S_j with probability a_{ij}, and this probability is the same for any t. The only special case is the first state. We define the *initial probability*, π_i, which is the probability that the first state in the sequence is S_i:

$$\pi_i \equiv P(q_1 = S_i) \tag{15.4}$$

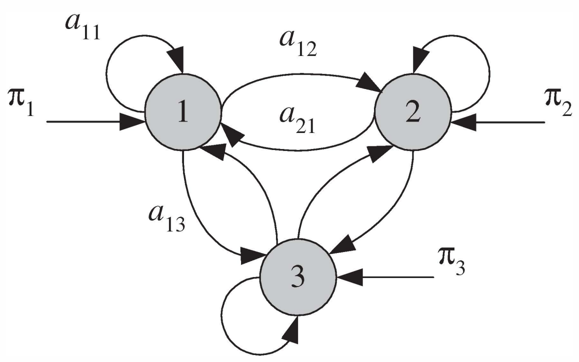


Figure 15.1 Example of a Markov model with three states. This is a stochastic automaton where π_i is the probability that the system starts in state S_i, and a_{ij} is the probability that the system moves from state S_i to state S_j.

satisfying

$$\sum_{i=1}^{N} \pi_i = 1 \tag{15.5}$$

$\mathbf{\Pi} = [\pi_i]$ is a vector of N elements that sum to 1.

OBSERVABLE MARKOV MODEL

In an *observable Markov model*, the states are observable. At any time t, we know q_t, and as the system moves from one state to another, we get an observation sequence that is a sequence of states. The output of the process is the set of states at each instant of time where each state corresponds to a physical observable event.

We have an observation sequence O that is the state sequence $O = Q = \{q_1 q_2 \cdots q_T\}$, whose probability is given as

$$P(O = Q|\mathbf{A}, \mathbf{\Pi}) = P(q_1) \prod_{t=2}^{T} P(q_t|q_{t-1}) = \pi_{q_1} a_{q_1 q_2} \cdots a_{q_{T-1} q_T} \tag{15.6}$$

π_{q_1} is the probability that the first state is q_1, $a_{q_1 q_2}$ is the probability of going from q_1 to q_2, and so on. We multiply these probabilities to get the probability of the whole sequence.

Let us now see an example (Rabiner and Juang 1986) to help us demonstrate. Assume we have N urns where each urn contains balls of only one color. So there is an urn of red balls, another of blue balls, and so forth.

Somebody draws balls from urns one by one and shows us their color. Let q_t denote the color of the ball drawn at time t. Let us say we have three states:

S_1 : red, S_2 = blue, S_3 : green

with initial probabilities:

$\mathbf{\Pi} = [0.5, 0.2, 0.3]^T$

a_{ij} is the probability of drawing from urn j (a ball of color j) after drawing a ball of color i from urn i. The transition matrix is, for example,

$$\mathbf{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

Given $\mathbf{\Pi}$ and $\mathbf{A}$, it is easy to generate K random sequences each of length T. Let us see how we can calculate the probability of a sequence. Assume that the first four balls are "red, red, green, green." This corresponds to the observation sequence $O = \{S_1, S_1, S_3, S_3\}$. Its probability is

$$\begin{aligned} P(O|\mathbf{A}, \mathbf{\Pi}) &= P(S_1) \cdot P(S_1|S_1) \cdot P(S_3|S_1) \cdot P(S_3|S_3) \\ &= \pi_1 \cdot a_{11} \cdot a_{13} \cdot a_{33} \\ &= 0.5 \cdot 0.4 \cdot 0.3 \cdot 0.8 = 0.048 \end{aligned} \tag{15.7}$$

Now, let us see how we can learn the parameters, $\mathbf{\Pi}, \mathbf{A}$. Given K sequences of length T, where q_t^k is the state at time t of sequence k, the initial probability estimate is the number of sequences starting with S_i divided by the number of sequences:

$$\hat{\pi}_i = \frac{\#\{\text{sequences starting with } S_i\}}{\#\{\text{sequences}\}} = \frac{\sum_k 1(q_1^k = S_i)}{K} \tag{15.8}$$

where $1(b)$ is 1 if b is true and 0 otherwise.

As for the transition probabilities, the estimate for a_{ij} is the number of transitions from S_i to S_j divided by the total number of transitions from S_i over all sequences:

$$\hat{a}_{ij} = \frac{\#\{\text{transitions from } S_i \text{ to } S_j\}}{\#\{\text{transitions from } S_i\}} = \frac{\sum_k \sum_{t=1}^{T-1} 1(q_t^k = S_i \text{ and } q_{t+1}^k = S_j)}{\sum_k \sum_{t=1}^{T-1} 1(q_t^k = S_i)} \tag{15.9}$$

$\hat{a}_{12}$ is the number of times a blue ball follows a red ball divided by the total number of red ball draws over all sequences.

15.3 Hidden Markov Models

HIDDEN MARKOV MODEL

In a *hidden Markov model* (HMM), the states are not observable, but when we visit a state, an observation is recorded that is a probabilistic function of the state. We assume a discrete observation in each state from the set $\{v_1, v_2, \ldots, v_M\}$:

$$b_j(m) \equiv P(O_t = v_m | q_t = S_j) \tag{15.10}$$

OBSERVATION PROBABILITY EMISSION PROBABILITY

$b_j(m)$ is the *observation probability*, or *emission probability*, that we observe the value $v_m, m = 1, \ldots, M$ in state S_j. We again assume a homogeneous model in which the probabilities do not depend on t. The values thus observed constitute the observation sequence O. The state sequence Q is not observed, that is what makes the model "hidden," but it should be inferred from the observation sequence O. Note that there are typically many different state sequences Q that could have generated the same observation sequence O, but with different probabilities; just as, given an iid sample from a normal distribution, there are an infinite number of (μ, σ) value pairs possible, we are interested in the one having the highest likelihood of generating the sample.

Note also that in this case of a hidden Markov model, there are two sources of randomness. In addition to randomly moving from one state to another, the observation in a state is also random.

Let us go back to our example. The hidden case corresponds to the urn-and-ball example where each urn contains balls of different colors. Let $b_j(m)$ denote the probability of drawing a ball of color m from urn j. We again observe a sequence of ball colors but without knowing the sequence of urns from which the balls were drawn. So it is as if now the urns are placed behind a curtain and somebody picks a ball at random from one of the urns and shows us only the ball, without showing us the urn from which it is picked. The ball is returned to the urn to keep the probabilities the same. The number of ball colors may be different from the number of urns. For example, let us say we have three urns and the observation sequence is

$$O = \{\text{red, red, green, blue, yellow}\}$$

In the previous case, knowing the observation (ball color), we knew the state (urn) exactly because there were separate urns for separate colors and each urn contained balls of only one color. The observable model is a special case of the hidden model where $M = N$ and $b_j(m)$ is 1 if $j = m$

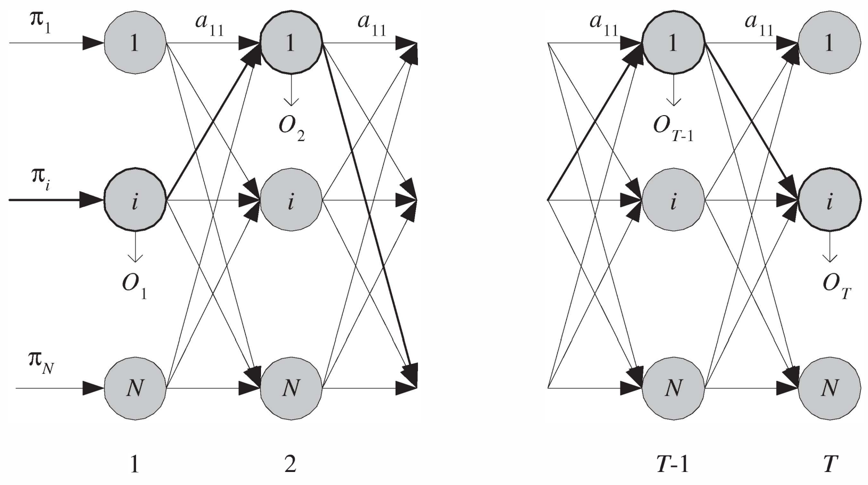


Figure 15.2 An HMM unfolded in time as a lattice (or trellis) showing all the possible trajectories. One path, shown in thicker lines, is the actual (unknown) state trajectory that generated the observation sequence.

and 0 otherwise. But in the case of a hidden model, a ball could have been picked from any urn. In this case, for the same observation sequence O, there may be many possible state sequences Q that could have generated O (see figure 15.2).

To summarize and formalize, an HMM has the following elements:

1. N: Number of states in the model

   $$S = \{S_1, S_2, \ldots, S_N\}$$

2. M: Number of distinct observation symbols in the *alphabet*

   $$V = \{v_1, v_2, \ldots, v_M\}$$

3. State transition probabilities:

   $$\mathbf{A} = [a_{ij}] \text{ where } a_{ij} \equiv P(q_{t+1} = S_j | q_t = S_i)$$

4. Observation probabilities:

   $$\mathbf{B} = [b_j(m)] \text{ where } b_j(m) \equiv P(O_t = v_m | q_t = S_j)$$

5. Initial state probabilities:

   $\mathbf{\Pi} = [\pi_i]$ where $\pi_i \equiv P(q_1 = S_i)$

N and M are implicitly defined in the other parameters so $\lambda = (\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$ is taken as the parameter set of an HMM. Given λ, the model can be used to generate an arbitrary number of observation sequences of arbitrary length, but as usual, we are interested in the other direction, that of estimating the parameters of the model given a training set of sequences.

15.4 Three Basic Problems of HMMs

Given a number of sequences of observations, we are interested in three problems:

1. Given a model λ, we would like to evaluate the probability of any given observation sequence, $O = \{O_1 O_2 \cdots O_T\}$, namely, $P(O|\lambda)$.

2. Given a model λ and an observation sequence O, we would like to find out the state sequence $Q = \{q_1 q_2 \cdots q_T\}$, which has the highest probability of generating O; namely, we want to find Q^* that maximizes $P(Q|O, \lambda)$.

3. Given a training set of observation sequences, $\mathcal{X} = \{O^k\}_k$, we would like to learn the model that maximizes the probability of generating $\mathcal{X}$; namely, we want to find λ^* that maximizes $P(\mathcal{X}|\lambda)$.

Let us see solutions to these one by one, with each solution used to solve the next problem, until we get to calculating λ or learning a model from data.

15.5 Evaluation Problem

Given an observation sequence $O = \{O_1 O_2 \cdots O_T\}$ and a state sequence $Q = \{q_1 q_2 \cdots q_T\}$, the probability of observing O given the state sequence Q is simply

$$P(O|Q, \lambda) = \prod_{t=1}^{T} P(O_t|q_t, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdots b_{q_T}(O_T) \tag{15.11}$$

which we cannot calculate because we do not know the state sequence. The probability of the state sequence Q is

$$P(Q|\lambda) = P(q_1)\prod_{t=2}^{T} P(q_t|q_{t-1}) = \pi_{q_1} a_{q_1 q_2} \cdots a_{q_{T-1} q_T} \tag{15.12}$$

Then the joint probability is

$$\begin{aligned} P(O,Q|\lambda) &= P(q_1)\prod_{t=2}^{T} P(q_t|q_{t-1}) \prod_{t=1}^{T} P(O_t|q_t) \\ &= \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T) \end{aligned} \tag{15.13}$$

We can compute $P(O|\lambda)$ by marginalizing over the joint, namely, by summing up over all possible Q:

$$P(O|\lambda) = \sum_{\text{all possible } Q} P(O,Q|\lambda)$$

However, this is not practical since there are N^T possible Q, assuming that all the probabilities are nonzero. Fortunately, there is an efficient procedure to calculate $P(O|\lambda)$, which is called the *forward-backward procedure* (see figure 15.3). It is based on the idea of dividing the observation sequence into two parts: the first one starting from time 1 until time t, and the second one from time $t+1$ until T.

FORWARD-BACKWARD PROCEDURE

FORWARD VARIABLE

We define the *forward variable* $\alpha_t(i)$ as the probability of observing the partial sequence $\{O_1 \cdots O_t\}$ until time t and being in S_i at time t, given the model λ:

$$\alpha_t(i) \equiv P(O_1 \cdots O_t, q_t = S_i|\lambda) \tag{15.14}$$

The nice thing about this is that it can be calculated recursively by accumulating results on the way.

- Initialization:

$$\begin{aligned} \alpha_1(i) &\equiv P(O_1, q_1 = S_i|\lambda) \\ &= P(O_1|q_1 = S_i, \lambda)P(q_1 = S_i|\lambda) \\ &= \pi_i b_i(O_1) \end{aligned} \tag{15.15}$$

- Recursion (see figure 15.3a):

$$\alpha_{t+1}(j) \equiv P(O_1 \cdots O_{t+1}, q_{t+1} = S_j|\lambda)$$

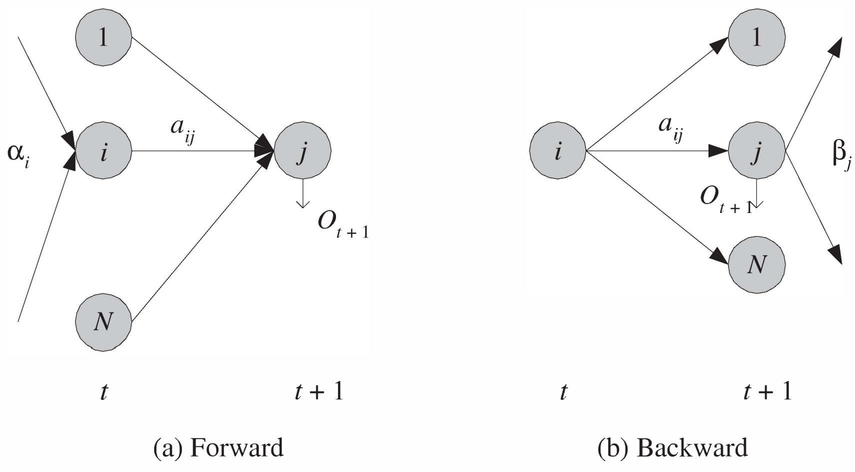


Figure 15.3 Forward-backward procedure: (a) computation of $\alpha_t(j)$ and (b) computation of $\beta_t(i)$.

$$
\begin{aligned}
&= P(O_1 \cdots O_{t+1} | q_{t+1} = S_j, \lambda) P(q_{t+1} = S_j | \lambda) \\
&= P(O_1 \cdots O_t | q_{t+1} = S_j, \lambda) P(O_{t+1} | q_{t+1} = S_j, \lambda) P(q_{t+1} = S_j | \lambda) \\
&= P(O_1 \cdots O_t, q_{t+1} = S_j | \lambda) P(O_{t+1} | q_{t+1} = S_j, \lambda) \\
&= P(O_{t+1} | q_{t+1} = S_j, \lambda) \sum_i P(O_1 \cdots O_t, q_t = S_i, q_{t+1} = S_j | \lambda) \\
&= P(O_{t+1} | q_{t+1} = S_j, \lambda) \\
&\quad \sum_i P(O_1 \cdots O_t, q_{t+1} = S_j | q_t = S_i, \lambda) P(q_t = S_i | \lambda) \\
&= P(O_{t+1} | q_{t+1} = S_j, \lambda) \\
&\quad \sum_i P(O_1 \cdots O_t | q_t = S_i, \lambda) P(q_{t+1} = S_j | q_t = S_i, \lambda) P(q_t = S_i | \lambda) \\
&= P(O_{t+1} | q_{t+1} = S_j, \lambda) \\
&\quad \sum_i P(O_1 \cdots O_t, q_t = S_i | \lambda) P(q_{t+1} = S_j | q_t = S_i, \lambda)
\end{aligned}
$$

$$
(15.16) \qquad = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1})
$$

$\alpha_t(i)$ explains the first t observations and ends in state S_i. We multiply this by the probability a_{ij} to move to state S_j, and because there are

N possible previous states, we need to sum up over all such possible previous S_i. $b_j(O_{t+1})$ then is the probability we generate the $(t+1)$st observation while in state S_j at time $t+1$.

When we calculate the forward variables, it is easy to calculate the probability of the observation sequence:

$$\begin{aligned} P(O|\lambda) &= \sum_{i=1}^{N} P(O, q_T = S_i|\lambda) \\ &= \sum_{i=1}^{N} \alpha_T(i) \qquad (15.17) \end{aligned}$$

$\alpha_T(i)$ is the probability of generating the full observation sequence and ending up in state S_i. We need to sum up over all such possible final states.

Computing $\alpha_t(i)$ is $\mathcal{O}(N^2T)$, and this solves our first evaluation problem in a reasonable amount of time. We do not need it now but let us similarly define the *backward variable*, $\beta_t(i)$, which is the probability of being in S_i at time t and observing the partial sequence $O_{t+1}\cdots O_T$:

BACKWARD VARIABLE

$$\beta_t(i) \equiv P(O_{t+1}\cdots O_T|q_t = S_i, \lambda) \qquad (15.18)$$

This can again be recursively computed as follows, this time going in the backward direction:

- Initialization (arbitrarily to 1):

$$\beta_T(i) = 1$$

- Recursion (see figure 15.3b):

$$\begin{aligned} \beta_t(i) &\equiv P(O_{t+1}\cdots O_T|q_t = S_i, \lambda) \\ &= \sum_j P(O_{t+1}\cdots O_T, q_{t+1} = S_j|q_t = S_i, \lambda) \\ &= \sum_j P(O_{t+1}\cdots O_T|q_{t+1} = S_j, q_t = S_i, \lambda) P(q_{t+1} = S_j|q_t = S_i, \lambda) \\ &= \sum_j P(O_{t+1}|q_{t+1} = S_j, q_t = S_i, \lambda) \\ &\quad P(O_{t+2}\cdots O_T|q_{t+1} = S_j, q_t = S_i, \lambda) P(q_{t+1} = S_j|q_t = S_i, \lambda) \\ &= \sum_j P(O_{t+1}|q_{t+1} = S_j, \lambda) \end{aligned}$$

$$P(O_{t+2}\cdots O_T|q_{t+1}=S_j,\lambda)P(q_{t+1}=S_j|q_t=S_i,\lambda)$$

$$= \sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\beta_{t+1}(j) \tag{15.19}$$

When in state S_i, we can go to N possible next states S_j, each with probability a_{ij}. While there, we generate the $(t+1)$st observation and $\beta_{t+1}(j)$ explains all the observations after time $t+1$, continuing from there.

One word of caution about implementation is necessary here: Both α_t and β_t values are calculated by multiplying small probabilities, and with long sequences we risk getting underflow. To avoid this, at each time step, we normalize $\alpha_t(i)$ by multiplying it with

$$c_t = \frac{1}{\sum_j \alpha_t(j)}$$

We also normalize $\beta_t(i)$ by multiplying it with the same c_t ($\beta_t(i)$ do not sum to 1). We cannot use equation 15.17 after normalization; instead, we have (Rabiner 1989)

$$P(O|\lambda) = \frac{1}{\prod_t c_t} \text{ or } \log P(O|\lambda) = -\sum_t \log c_t \tag{15.20}$$

15.6 Finding the State Sequence

We now move on to the second problem, that of finding the state sequence $Q = \{q_1q_2\cdots q_T\}$ having the highest probability of generating the observation sequence $O = \{O_1O_2\cdots O_T\}$, given the model λ.

Let us define $\gamma_t(i)$ as the probability of being in state S_i at time t, given O and λ, which can be computed as follows:

$$\begin{aligned}
\gamma_t(i) &\equiv P(q_t=S_i|O,\lambda) \qquad (15.21)\\
&= \frac{P(O|q_t=S_i,\lambda)P(q_t=S_i|\lambda)}{P(O|\lambda)}\\
&= \frac{P(O_1\cdots O_t|q_t=S_i,\lambda)P(O_{t+1}\cdots O_T|q_t=S_i,\lambda)P(q_t=S_i|\lambda)}{\sum_{j=1}^N P(O,q_t=S_j|\lambda)}\\
&= \frac{P(O_1\cdots O_t,q_t=S_i|\lambda)P(O_{t+1}\cdots O_T|q_t=S_i,\lambda)}{\sum_{j=1}^N P(O|q_t=S_j,\lambda)P(q_t=S_j|\lambda)}\\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^N \alpha_t(j)\beta_t(j)} \qquad (15.22)
\end{aligned}$$

Here we see how nicely $\alpha_t(i)$ and $\beta_t(i)$ split the sequence between them: The forward variable $\alpha_t(i)$ explains the starting part of the sequence until time t and ends in S_i, and the backward variable $\beta_t(i)$ takes it from there and explains the ending part until time T.

The numerator $\alpha_t(i)\beta_t(i)$ explains the whole sequence given that at time t, the system is in state S_i. We need to normalize by dividing this over all possible intermediate states that can be traversed at time t, and guarantee that $\sum_i \gamma_t(i) = 1$.

To find the state sequence, for each time step t, we can choose the state that has the highest probability:

(15.23) $$q_t^* = \arg\max_i \gamma_t(i)$$

but this may choose S_i and S_j as the most probable states at time t and $t+1$ even when $a_{ij} = 0$. To find the single best state *sequence* (path), we use the *Viterbi algorithm*, based on dynamic programming, which takes such transition probabilities into account.

VITERBI ALGORITHM

Given state sequence $Q = q_1 q_2 \cdots q_T$ and observation sequence $O = O_1 \cdots O_T$, we define $\delta_t(i)$ as the probability of the highest probability path at time t that accounts for the first t observations and ends in S_i:

(15.24) $$\delta_t(i) \equiv \max_{q_1 q_2 \cdots q_{t-1}} p(q_1 q_2 \cdots q_{t-1}, q_t = S_i, O_1 \cdots O_t | \lambda)$$

Then we can recursively calculate $\delta_{t+1}(i)$ and the optimal path can be read by backtracking from T, choosing the most probable at each instant. The algorithm is as follows:

1. Initialization:

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(O_1) \\ \psi_1(i) &= 0 \end{aligned}$$

2. Recursion:

$$\begin{aligned} \delta_t(j) &= \max_i \delta_{t-1}(i) a_{ij} \cdot b_j(O_t) \\ \psi_t(j) &= \arg\max_i \delta_{t-1}(i) a_{ij} \end{aligned}$$

3. Termination:

$$\begin{aligned} p^* &= \max_i \delta_T(i) \\ q_T^* &= \arg\max_i \delta_T(i) \end{aligned}$$

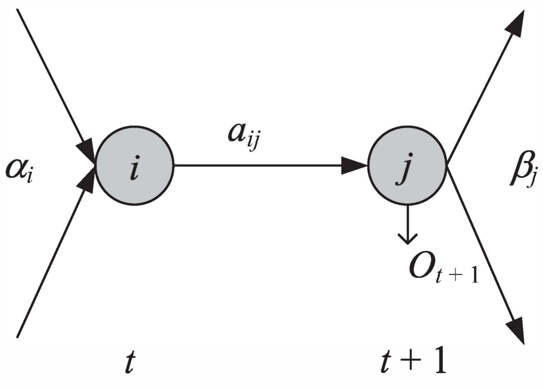


Figure 15.4 Computation of arc probabilities, $\xi_t(i,j)$.

4. Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*),\ t = T-1, T-2, \ldots, 1$$

Using the lattice structure of figure 15.2, $\psi_t(j)$ keeps track of the state that maximizes $\delta_t(j)$ at time $t-1$, that is, the best previous state. The Viterbi algorithm has the same complexity with the forward phase, where instead of the sum, we take the maximum at each step.

15.7 Learning Model Parameters

We now move on to the third problem, learning an HMM from data. The approach is maximum likelihood, and we would like to calculate λ^* that maximizes the likelihood of the sample of training sequences, $\mathcal{X} = \{O^k\}_{k=1}^K$, namely, $P(\mathcal{X}|\lambda)$. We start by defining a new variable that will become handy later on.

We define $\xi_t(i,j)$ as the probability of being in S_i at time t and in S_j at time $t+1$, given the whole observation O and λ:

$$\xi_t(i,j) \equiv P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{15.25}$$

which can be computed as follows (see figure 15.4):

$$\xi_t(i,j) \equiv P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

$$\begin{aligned}
&= \frac{P(O|q_t = S_i, q_{t+1} = S_j, \lambda)P(q_t = S_i, q_{t+1} = S_j|\lambda)}{P(O|\lambda)} \\
&= \frac{P(O|q_t = S_i, q_{t+1} = S_j, \lambda)P(q_{t+1} = S_j|q_t = S_i, \lambda)P(q_t = S_i|\lambda)}{P(O|\lambda)} \\
&= \left(\frac{1}{P(O|\lambda)}\right) P(O_1 \cdots O_t|q_t = S_i, \lambda)P(O_{t+1}|q_{t+1} = S_j, \lambda) \\
&\quad P(O_{t+2} \cdots O_T|q_{t+1} = S_j, \lambda)a_{ij}P(q_t = S_i|\lambda) \\
&= \left(\frac{1}{P(O|\lambda)}\right) P(O_1 \cdots O_t, q_t = S_i|\lambda)P(O_{t+1}|q_{t+1} = S_j, \lambda) \\
&\quad P(O_{t+2} \cdots O_T|q_{t+1} = S_j, \lambda)a_{ij} \\
&= \frac{\alpha_t(i)b_j(O_{t+1})\beta_{t+1}(j)a_{ij}}{\sum_k \sum_l P(q_t = S_k, q_{t+1} = S_l, O|\lambda)} \\
&= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_k \sum_l \alpha_t(k)a_{kl}b_l(O_{t+1})\beta_{t+1}(l)} \qquad (15.26)
\end{aligned}$$

$\alpha_t(i)$ explains the first t observations and ends in state S_i at time t. We move on to state S_j with probability a_{ij}, generate the $(t+1)$st observation, and continue from S_j at time $t+1$ to generate the rest of the observation sequence. We normalize by dividing for all such possible pairs that can be visited at time t and $t+1$.

If we want, we can also calculate the probability of being in state S_i at time t by marginalizing over the arc probabilities for all possible next states:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j) \qquad (15.27)$$

Note that if the Markov model were not hidden but observable, both $\gamma_t(i)$ and $\xi_t(i, j)$ would be 0/1. In this case when they are not, we estimate them with posterior probabilities that give us *soft counts*. This is just like the difference between supervised classification and unsupervised clustering where we did and did not know the class labels, respectively. In unsupervised clustering using EM (section 7.4), not knowing the class labels, we estimated them first (in the E-step) and calculated the parameters with these estimates (in the M-step).

SOFT COUNTS

BAUM-WELCH ALGORITHM

Similarly here we have the *Baum-Welch algorithm*, which is an EM procedure. At each iteration, first in the E-step, we compute $\xi_t(i, j)$ and $\gamma_t(i)$ values given the current $\lambda = (\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$, and then in the M-step, we recalculate λ given $\xi_t(i, j)$ and $\gamma_t(i)$. These two steps are alternated until convergence during which, it has been shown, $P(O|\lambda)$ never decreases.

Assume indicator variables z_i^t as

(15.28) $$z_i^t = \begin{cases} 1 & \text{if } q_t = S_i \\ 0 & \text{otherwise} \end{cases}$$

and

(15.29) $$z_{ij}^t = \begin{cases} 1 & \text{if } q_t = S_i \text{ and } q_{t+1} = S_j \\ 0 & \text{otherwise} \end{cases}$$

These are 0/1 in the case of an observable Markov model and are hidden random variables in the case of an HMM. In this latter case, we estimate them in the E-step as

(15.30) $$\begin{aligned} E[z_i^t] &= \gamma_t(i) \\ E[z_{ij}^t] &= \xi_t(i,j) \end{aligned}$$

In the M-step, we calculate the parameters given these estimated values. The expected number of transitions from S_i to S_j is $\sum_t \xi_t(i,j)$ and the total number of transitions from S_i is $\sum_t \gamma_t(i)$. The ratio of these two gives us the probability of transition from S_i to S_j at any time:

(15.31) $$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Note that this is the same as equation 15.9, except that the actual counts are replaced by estimated soft counts.

The probability of observing v_m in S_j is the expected number of times v_m is observed when the system is in S_j over the total number of times the system is in S_j:

(15.32) $$\hat{b}_j(m) = \frac{\sum_{t=1}^{T} \gamma_t(j) 1(O_t = v_m)}{\sum_{t=1}^{T} \gamma_t(j)}$$

When there are multiple observation sequences

$$\mathcal{X} = \{O^k\}_{k=1}^K$$

which we assume to be independent

$$P(\mathcal{X}|\lambda) = \prod_{k=1}^{K} P(O^k|\lambda)$$

the parameters are now averages over all observations in all sequences:

(15.33)
$$\hat{a}_{ij} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\xi_t^k(i,j)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\gamma_t^k(i)}$$
$$\hat{b}_j(m) = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T_k}\gamma_t^k(j)1(O_t^k = v_m)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k}\gamma_t^k(j)}$$
$$\hat{\pi}_i = \frac{\sum_{k=1}^{K}\gamma_1^k(i)}{K}$$

15.8 Continuous Observations

In our discussion, we assumed discrete observations modeled as a multinomial

(15.34)
$$P(O_t|q_t = S_j, \lambda) = \prod_{m=1}^{M} b_j(m)^{r_m^t}$$

where

(15.35)
$$r_m^t = \begin{cases} 1 & \text{if } O_t = v_m \\ 0 & \text{otherwise} \end{cases}$$

If the inputs are continuous, one possibility is to discretize them and then use these discrete values as observations. Typically, a vector quantizer (section 7.3) is used for this purpose of converting continuous values to the discrete index of the closest reference vector. For example, in speech recognition, a word utterance is divided into short speech segments corresponding to phonemes or part of phonemes; after preprocessing, these are discretized using a vector quantizer and an HMM is then used to model a word utterance as a sequence of them.

We remember that *k*-means used for vector quantization is the hard version of a Gaussian mixture model:

(15.36)
$$p(O_t|q_t = S_j, \lambda) = \sum_{l=1}^{L} P(\mathcal{G}_l)p(O_t|q_t = S_j, \mathcal{G}_l, \lambda)$$

where

(15.37)
$$p(O_t|q_t = S_j, \mathcal{G}_l, \lambda) \sim \mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)$$

and the observations are kept continuous. In this case of Gaussian mixtures, EM equations can be derived for the component parameters (with suitable regularization to keep the number of parameters in check) and the mixture proportions (Rabiner 1989).

Let us see the case of a scalar continuous observation, $O_t \in \Re$. The easiest is to assume a normal distribution:

$$p(O_t|q_t = S_j, \lambda) \sim \mathcal{N}(\mu_j, \sigma_j^2) \tag{15.38}$$

which implies that in state S_j, the observation is drawn from a normal with mean μ_j and variance σ_j^2. The M-step equations in this case are

$$\begin{aligned} \hat{\mu}_j &= \frac{\sum_t \gamma_t(j) O_t}{\sum_t \gamma_t(j)} \\ \hat{\sigma}_j^2 &= \frac{\sum_t \gamma_t(j)(O_t - \hat{\mu}_j)^2}{\sum_t \gamma_t(j)} \end{aligned} \tag{15.39}$$

15.9 The HMM as a Graphical Model

We discussed graphical models in chapter 14, and the hidden Markov model can also be depicted as a graphical model. The three successive states q_{t-2}, q_{t-1}, q_t correspond to the three states on a chain in a first-order Markov model. The state at time t, q_t, depends only on the state at time $t-1$, q_{t-1}, and given q_{t-1}, q_t is independent of q_{t-2}

$$P(q_t|q_{t-1}, q_{t-2}) = P(q_t|q_{t-1})$$

as given by the state transition probability matrix $\mathbf{A}$ (see figure 15.5). Each hidden variable generates a discrete observation that is observed, as given by the observation probability matrix $\mathbf{B}$. The forward-backward procedure of hidden Markov models we discuss in this chapter is an application of belief propagation that we discussed in section 14.5.

Continuing with the graphical formalism, different HMM types can be devised and depicted as different graphical models. In figure 15.6a, an

INPUT-OUTPUT HMM

input-output HMM is shown where there are two separate observed input-output sequences and there is also a sequence of hidden states (Bengio and Frasconi 1996). In some applications this is the case, namely, additional to the observation sequence O_t, we have an input sequence, x_t, and we know that the observation depends also on the input. In such a case, we condition the observation O_t in state S_j on the input x^t and write $P(O_t|q_t = S_j, x_t)$. When the observations are numeric, for example, we replace equation 15.38 with a generalized model

$$p(O_t|q_t = S_j, x_t, \lambda) \sim \mathcal{N}(g_j(x^t|\theta_j), \sigma_j^2) \tag{15.40}$$

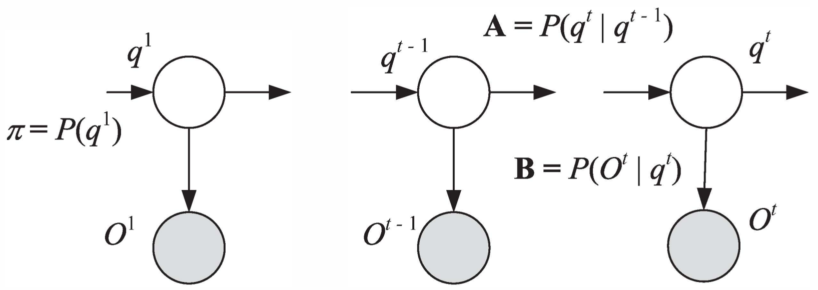

q^1
$\pi = P(q^1)$
O^1
q^{t-1}
O^{t-1}
$\mathbf{A} = P(q^t \mid q^{t-1})$
$\mathbf{B} = P(O^t \mid q^t)$
q^t
O^t

Figure 15.5 A hidden Markov model can be drawn as a graphical model where q^t are the hidden states and shaded O^t are observed.

where, for example, assuming a linear model, we have

$$g_j(x^t|w_j, w_{j0}) = w_j x^t + w_{j0} \tag{15.41}$$

If the observations are discrete and multinomial, we have a classifier taking x^t as input and generating a 1-of-M output, or we can generate posterior class probabilities and keep the observations continuous.

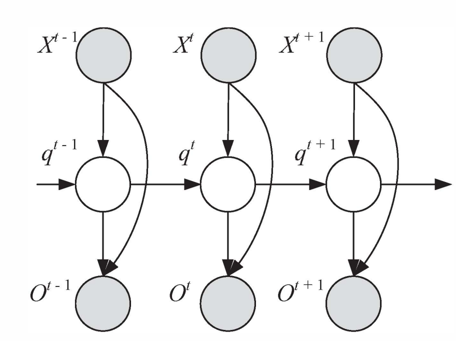
Similarly, the state transition probabilities can also be conditioned on the input, namely, $P(q_{t+1} = S_j|q_t = S_i, x_t)$, which is implemented by a classifier choosing the state at time $t+1$ as a function of the state at time t and the input. This is a *Markov mixture of experts* (Meila and Jordan 1996) and is a generalization of the mixture of experts architecture (section 12.8) where the gating network keeps track of the decision it made in the previous time step. This has the advantage that the model is no longer homogeneous; different observation and transition probabilities are used at different time steps. There is still a single model for each state, parameterized by θ_j, but it generates different transition or observation probabilities depending on the input seen. It is possible that the input is not a single value but a window around time t making the input a vector; this allows handling applications where the input and observation sequences have different lengths.

MARKOV MIXTURE OF EXPERTS

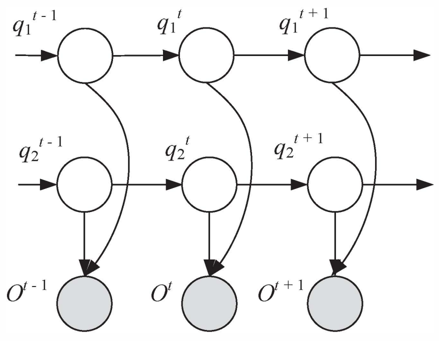
Even if there is no other explicit input sequence, an HMM with input can be used by generating an "input" through some prespecified function of previous observations

$$\boldsymbol{x}_t = \boldsymbol{f}(O_{t-\tau}, \ldots, O_{t-1})$$

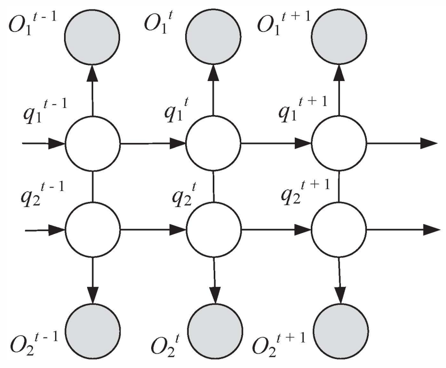
thereby providing a window of size τ of contextual input.



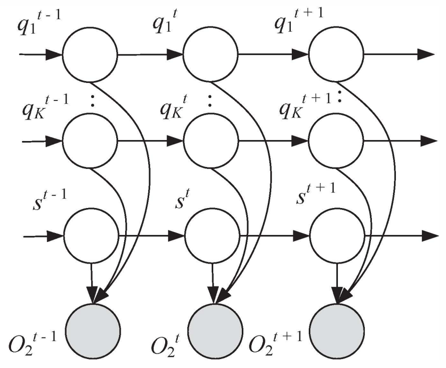
(a) Input-output HMM



(b) Factorial HMM



(c) Coupled HMM



(d) Switching HMM

Figure 15.6 Different types of HMM model different assumptions about the way the observed data (shown shaded) is generated from Markov sequences of latent variables.

FACTORIAL HMM

PEDIGREE

Another HMM type that can be easily visualized is a *factorial HMM*, where there are multiple separate hidden sequences that interact to generate a single observation sequence. An example is a *pedigree* that displays the parent-child relationship (Jordan 2004); figure 15.6b models *meiosis* where the two sequences correspond to the chromosomes of the father and the mother (which are independent), and at each locus (gene), the offspring receives one allele from the father and the other allele from the mother.

COUPLED HMM

A *coupled HMM*, shown in figure 15.6c, models two parallel but interacting hidden sequences that generate two parallel observation sequences. For example, in speech recognition, we may have one observed acoustic sequence of uttered words and one observed visual sequence of lip images, each having its hidden states where the two are dependent.

SWITCHING HMM

In a *switching HMM*, shown in figure 15.6d, there are K parallel independent hidden state sequences, and the state variable S at any one time picks one of them and the chosen one generates the output. That is, we switch between state sequences as we go along.

LINEAR DYNAMICAL SYSTEM
KALMAN FILTER

In HMM proper, though the observation may be continuous, the state variable is discrete; in a *linear dynamical system*, also known as the *Kalman filter*, both the state and the observations are continuous. In the basic case, the state at time t is a linear function of the state at $t-1$ with additive zero-mean Gaussian noise, and, at each state, the observation is another linear function of the state with additive zero-mean Gaussian noise. The two linear mappings and the covariances of the two noise sources make up the parameters. All HMM variants we discussed earlier can similarly be generalized to use continuous states.

By suitably modifying the graphical model, we can adapt the architecture to the characteristics of the process that generates the data. This process of matching the model to the data is a model selection procedure to best trade off bias and variance. The disadvantage is that exact inference may no longer be possible on such extended HMMs, and we would need approximation or sampling methods (Ghahramani 2001; Jordan 2004).

15.10 Model Selection in HMMs

Just like any model, the complexity of an HMM should be tuned so as to balance its complexity with the size and properties of the data at hand. One possibility is to tune the topology of the HMM. In a fully connected (ergodic) HMM, there is transition from a state to any other state, which makes $\mathbf{A}$ a full $N \times N$ matrix. In some applications, only certain transitions are allowed, with the disallowed transitions having their $a_{ij} = 0$. When there are fewer possible next states, $N' < N$, the complexity of forward-backward passes and the Viterbi procedure is $\mathcal{O}(NN'T)$ instead of $\mathcal{O}(N^2T)$.

LEFT-TO-RIGHT HMMS

For example, in speech recognition, *left-to-right HMMs* are used, which have their states ordered in time so that as time increases, the state index increases or stays the same. Such a constraint allows modeling sequences whose properties change over time as in speech, and when we get to a state, we know approximately the states preceding it. There is the property that we never move to a state with a smaller index, namely, $a_{ij} = 0$, for $j < i$. Large changes in state indices are not allowed either, namely, $a_{ij} = 0$, for $j > i + \tau$. The example of the left-to-right HMM given in figure 15.7 with $\tau = 2$ has the state transition matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

Another factor that determines the complexity of an HMM is the number of states N. Because the states are hidden, their number is not known and should be chosen before training. This is determined using prior information and can be fine-tuned by cross-validation, namely, by checking the likelihood of validation sequences.

When used for classification, we have a set of HMMs, each one modeling the sequences belonging to one class. For example, in spoken word recognition, examples of each word train a separate model, λ_i. Given a new word utterance O to classify, all of the separate word models are evaluated to calculate $P(O|\lambda_i)$. We then use Bayes' rule to get the posterior probabilities

(15.42) $$P(\lambda_i|O) = \frac{P(O|\lambda_i)P(\lambda_i)}{\sum_j P(O|\lambda_j)P(\lambda_j)}$$

where $P(\lambda_i)$ is the prior probability of word i. The utterance is assigned to the word having the highest posterior. This is the likelihood-based approach; there is also work on discriminative HMM trained directly to maximize the posterior probabilities. When there are several pronunciations of the same word, these are defined as parallel paths in the HMM for the word.

In the case of a continuous input like speech, the difficult task is that of segmenting the signal into small discrete observations. Typically, *phones* are used that are taken as the primitive parts, and combining them, longer sequences (e.g., words) are formed. Each phone is recognized in parallel (by the vector quantizer), then the HMM is used to combine them serially. If the speech primitives are simple, then the HMM becomes complex and vice versa. In connected speech recognition where the words are

PHONES

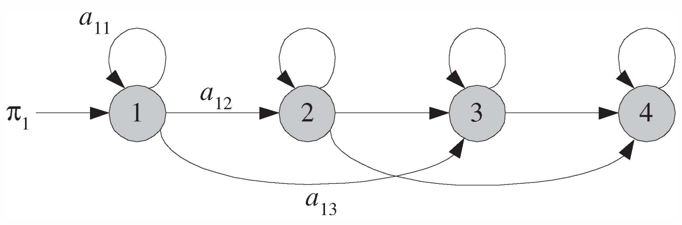


Figure 15.7 Example of a left-to-right HMM.

not uttered one by one with clear pauses between them, there is a hierarchy of HMMs at several levels; one combines phones to recognize words, another combines words to recognize sentences by building a language model, and so forth.

Hybrid neural network/HMM models were also used for speech recognition (Morgan and Bourlard 1995). In such a model, a multilayer perceptron (chapter 11) is used to capture temporally local but possibly complex and nonlinear primitives, for example, phones, while the HMM is used to learn the temporal structure. The neural network acts as a preprocessor and translates the raw observations in a time window to a form that is easier to model than the output of a vector quantizer.

An HMM can be visualized as a graphical model and evaluation in an HMM is a special case of the belief propagation algorithm that we discuss in chapter 14. The reason we devote a special chapter is the widespread successful use of this particular model, especially in automatic speech recognition. But the basic HMM architecture can be extended—for example, by having multiple sequences, or by introducing hidden (latent) variables, as we discuss in section 15.9.

In chapter 16, we discuss the Bayesian approach and in section 16.8, we discuss the nonparametric Bayesian methods where the model structure can be made more complex over time as more data arrives. One application of that is the *infinite HMM* (Beal, Ghahramani, and Rasmussen 2002).

15.11 Notes

The HMM is a mature technology, and there are HMM-based commercial speech recognition systems in actual use (Rabiner and Juang 1993; Jelinek 1997). In section 11.12, we discussed how to train multilayer perceptrons for recognizing sequences. HMMs have the advantage over time delay neural networks in that no time window needs to be defined a priori, and they train better than recurrent neural networks. HMMs are applied to diverse sequence recognition tasks. Applications of HMMs to bioinformatics is given in Baldi and Brunak 1998, and to natural language processing in Manning and Schütze 1999. It is also applied to online handwritten character recognition, which differs from optical recognition in that the writer writes on a touch-sensitive pad and the input is a sequence of (x, y) coordinates of the pen tip as it moves over the pad and is not a static image. Bengio et al. (1995) explain a hybrid system for online recognition where an MLP recognizes individual characters, and an HMM combines them to recognize words. Various applications of the HMM and several extensions, for example, discriminative HMMs, are discussed in Bengio 1999. A more recent survey of what HMMs can and cannot do is Bilmes 2006.

In any such recognition system, one critical point is to decide how much to do things in parallel and what to leave to serial processing. In speech recognition, phonemes may be recognized by a parallel system that corresponds to assuming that all the phoneme sound is uttered in one time step. The word is then recognized serially by combining the phonemes. In an alternative system, phonemes themselves may be designed as a sequence of simpler speech sounds, if the same phoneme has many versions, for example, depending on the previous and following phonemes. Doing things in parallel is good but only to a degree; one should find the ideal balance of parallel and serial processing. To be able to call anyone at the touch of a button, we would need millions of buttons on our telephone; instead, we have ten buttons and we press them in a sequence to dial the number.

We discussed graphical models in chapter 14, and we know that HMMs can be considered a special class of graphical models and inference and learning operations on HMMs are analogous to their counterparts in graphical models (Smyth, Heckerman, and Jordan 1997). There are various extensions to HMMs, such as *factorial HMMs*, where at each time step, there are a number of states that collectively generate the observation and *tree-structured HMMs* where there is a hierarchy of states. The general formalism also allows us to treat continuous as well as discrete states, known as *linear dynamical systems*. For some of these models, exact inference is not possible and one needs to use approximation or sampling methods (Ghahramani 2001).

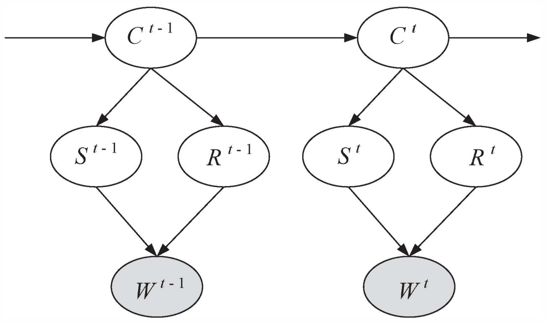


Figure 15.8 A dynamic version where we have a chain of graphs to show dependency in weather on consecutive days.

Actually, any graphical model can be extended in time by unfolding it in time and adding dependencies between successive copies. In fact, a hidden Markov model is nothing but a sequence of clustering problems where the cluster index at time t is dependent not only on observation at time t but also on the index at time $t - 1$, and the Baum-Welch algorithm is expectation-maximization extended to also include this dependency in time. In section 6.5, we discussed factor analysis where a small number of hidden factors generate the observation; similarly, a linear dynamical system may be viewed as a sequence of such factor analysis models where the current factors also depend on the previous factors.

This dynamic dependency may be added when needed. For example, figure 14.5 models the cause of wet grass for a particular day; if we believe that yesterday's weather has an influence on today's weather (and we should—it tends to be cloudy on successive days, then sunny for a number of days, and so on), we can have the dynamic graphical model shown in figure 15.8 where we model this dependency.

15.12 Exercises

1. Given the observable Markov model with three states, S_1, S_2, S_3, initial probabilities

   $\mathbf{\Pi} = [0.5, 0.2, 0.3]^T$

and transition probabilities

$$\mathbf{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

generate 100 sequences of 1,000 states.

2. Using the data generated by the previous exercise, estimate Π, $\mathbf{A}$ and compare with the parameters used to generate the data.
3. Formalize a second-order Markov model. What are the parameters? How can we calculate the probability of a given state sequence? How can the parameters be learned for the case of an observable model?

   SOLUTION: In a second-order model, the current state depends on the two previous states:

   $a_{ijk} \equiv P(q_{t+2} = S_k | q_{t+1} = S_j, q_t = S_i)$

   Initial state probability defines the probability of the first state:

   $\pi_i \equiv P(q_1 = S_i)$

   We also need parameters to define the probability of the second state given the first state:

   $\theta_{ij} \equiv P(q_2 = S_j | q_1 = S_i)$

   Given a second-order observable MM with parameters $\lambda = (\Pi, \Theta, \mathbf{A})$, the probability of an observed state sequence is

$$\begin{aligned} P(O = Q|\lambda) &= P(q_1)P(q_2|q_1)\prod_{t=3}^{T} P(q_t|q_{t-1}, q_{t-2}) \\ &= \pi_{q_1}\theta_{q_2 q_1} a_{q_3 q_2 q_1} a_{q_4 q_3 q_2} \cdots a_{q_T q_{T-1} q_{T-2}} \end{aligned}$$

   The probabilities are estimated as proportions:

$$\begin{aligned} \hat{\pi}_i &= \frac{\sum_k 1(q_1^k = S_i)}{K} \\ \hat{\theta}_{ij} &= \frac{\sum_k 1(q_2^k = S_j \text{ and } q_1^k = S_i)}{\sum_k 1(q_1^k = S_i)} \\ \hat{a}_{ijk} &= \frac{\sum_k \sum_{t=3}^{T} 1(q_t^k = S_k \text{ and } q_{t-1}^k = S_j \text{ and } q_{t-2}^k = S_i)}{\sum_k \sum_{t=3}^{T} 1(q_{t-1}^k = S_j \text{ and } q_{t-2}^k = S_i)} \end{aligned}$$

4. Show that any second- (or higher-order) Markov model can be converted to a first-order Markov model.

   SOLUTION: In a second-order model, each state depends on the two previous states. We can define a new set of states corresponding to the Cartesian product of the original set of states with itself. A first-order model defined on this new N^2 states corrresponds to a second-order model defined on the original N states.

5. Some researchers define a Markov model as generating an observation while traversing an arc, instead of on arrival at a state. Is this model any more powerful than what we have discussed?

   SOLUTION: Similar to the case of the previous exercise, if the output depends not only on the current state but also on the next state, we can define new states corresponding to this pair and have the output generated by this (joint) state.

6. Generate training and validation sequences from an HMM of your choosing. Then train different HMMs by varying the number of hidden states on the same training set and calculate the validation likelihoods. Observe how the validation likelihood changes as the number of states increases.

7. If in equation 15.38 we have multivariate observations, what will the M-step equations be?

   SOLUTION: If we have d-dimensional $\boldsymbol{O}_t \in \Re^d$, drawn from d-variate Gaussians with their mean vectors and covariance matrices

   $$p(\boldsymbol{O}_t|q_t = S_j, \lambda) \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

   the M-step equations are

   $$\begin{aligned}\hat{\boldsymbol{\mu}}_j &= \frac{\sum_t \gamma_t(j)\boldsymbol{O}_t}{\sum_t \gamma_t(j)}\\ \hat{\boldsymbol{\Sigma}}_j &= \frac{\sum_t \gamma_t(j)(\boldsymbol{O}_t - \hat{\boldsymbol{\mu}}_j)(\boldsymbol{O}_t - \hat{\boldsymbol{\mu}}_j)^T}{\sum_t \gamma_t(j)}\end{aligned}$$

8. Consider the urn-and-ball example where we draw *without replacement*. How will it be different?

   SOLUTION: If we draw without replacement, then at each iteration, the number of balls change, which means that the observation probabilities, **B**, change. We will no longer have a homogenous model.

9. Let us say at any time we have two observations from two different alphabets; for example, let us say we are observing the values of two currencies every day. How can we implement this using HMM?

   SOLUTION: In such a case, what we have is a hidden state generating two different observations. That is, we have two **B**, each trained with its own observation sequence. These two observations then need to be combined to estimate **A** and $\boldsymbol{\pi}$.

10. How can we have an incremental HMM where we add new hidden states when necessary?

    SOLUTION: Again, this is a state space search. Our aim may be to maximize validation log likelihood, and an operator allows us to add a hidden state. We do then a forward search. There are structure learning algorithms for the more general case of graphical models, which we discussed in chapter 14.

15.13 References

Baldi, P., and S. Brunak. 1998. *Bioinformatics: The Machine Learning Approach*. Cambridge, MA: MIT Press.

Beal, M. J., Z. Ghahramani, and C. E. Rasmussen. 2002. “The Infinite Hidden Markov Model.” In *Advances in Neural Information Processing Systems 14*, ed. T. G. Dietterich, S. Becker, and Z. Ghahramani, 577–585. Cambridge, MA: MIT Press.

Bengio, Y. 1999. “Markovian Models for Sequential Data.” *Neural Computing Surveys* 2: 129–162.

Bengio, Y., and P. Frasconi. 1996. “Input-Output HMMs for Sequence Processing.” *IEEE Transactions on Neural Networks* 7:1231–1249.

Bengio, Y., Y. Le Cun, C. Nohl, and C. Burges. 1995. “LeRec: A NN/HMM Hybrid for On-line Handwriting Recognition.” *Neural Computation* 7:1289–1303.

Bilmes, J. A. 2006. “What HMMs Can Do.” *IEICE Transactions on Information and Systems* E89-D:869–891.

Ghahramani, Z. 2001. “An Introduction to Hidden Markov Models and Bayesian Networks.” *International Journal of Pattern Recognition and Artificial Intelligence* 15:9–42.

Jelinek, F. 1997. *Statistical Methods for Speech Recognition*. Cambridge, MA: MIT Press.

Jordan, M. I. 2004. “Graphical Models.” *Statistical Science* 19:140–155.

Manning, C. D., and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Meila, M., and M. I. Jordan. 1996. “Learning Fine Motion by Markov Mixtures of Experts.” In *Advances in Neural Information Processing Systems 8*, ed. D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, 1003–1009. Cambridge, MA: MIT Press.

Morgan, N., and H. Bourlard. 1995. “Continuous Speech Recognition: An Introduction to the Hybrid HMM/Connectionist Approach.” *IEEE Signal Processing Magazine* 12:25–42.

Smyth, P., D. Heckerman, and M. I. Jordan. 1997. “Probabilistic Independence Networks for Hidden Markov Probability Models.” *Neural Computation* 9:227–269.

Rabiner, L. R. 1989. “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.” *Proceedings of the IEEE* 77:257–286.

Rabiner, L. R., and B. H. Juang. 1986. “An Introduction to Hidden Markov Models.” *IEEE Acoustics, Speech, and Signal Processing Magazine* 3:4–16.

Rabiner, L. R., and B. H. Juang. 1993. *Fundamentals of Speech Recognition*. New York: Prentice Hall.