An Introduction to Probabilistic Graphical Models

Michael I. Jordan
University of California, Berkeley

June 30, 2003

Chapter 12

Hidden Markov Models

In this chapter we finally relax the assumption of independent, identically distributed (IID) sampling that we have labored under until now. A hidden Markov model (HMM) is a graphical model that is appropriate for modeling sequential data; i.e., data sets in which successive samples are no longer assumed to be independent.

An HMM is a natural generalization of a mixture model; indeed, it is perhaps best viewed as a "dynamical" mixture model. To reflect this point of view, we adjust our terminology somewhat, referring to the "mixture components" of the mixture model as "states." To see exactly what kind of generalization is involved, let us recall the process of generating IID data under a mixture model, using the new language (cf. Figure 12.1(a)):

- At each step, a state is selected according to the distribution $p(z)$. This selection is made independently of the choice of states at other steps.

- Given the state, a data vector is chosen from a distribution $p(x|z)$.

Within the HMM framework we no longer assume that the states are chosen independently at each step, but rather we assume that the choice of a state at a given step depends on the choice of the state at the previous step. Thus we augment the basic mixture model to include a matrix of *transition probabilities* linking the states at neighboring steps. If there are M states, then this is an $M \times M$ matrix, whose (i, j)th entry represents the probability of transitioning from the ith state at a given step to the jth state at the following step. The process of generating data under the HMM is suggested in Figure 12.1(b), where we have drawn arrows between the probability distributions labeled by the states to suggest the transition probabilities. Other than the introduction of a state transition matrix, the HMM is the same as the simpler mixture model—in particular, given the state at a given step a data vector is generated from a distribution that depends only on that state.

As in any mixture model, the states underlying the data generation process are assumed to be "hidden" from the learner. We envision an HMM-based learning system observing the pattern of data in Figure 12.1(a)–one data point at a time–and interpreting the sequence in terms of the hypothesized states and state transitions of Figure 12.1(b). Just as in the simpler mixture model, the fact that the data form clusters is grist for the HMM mill, allowing the learner to differentiate the states. But while the clustering of the data is necessary for an HMM-based learner to be

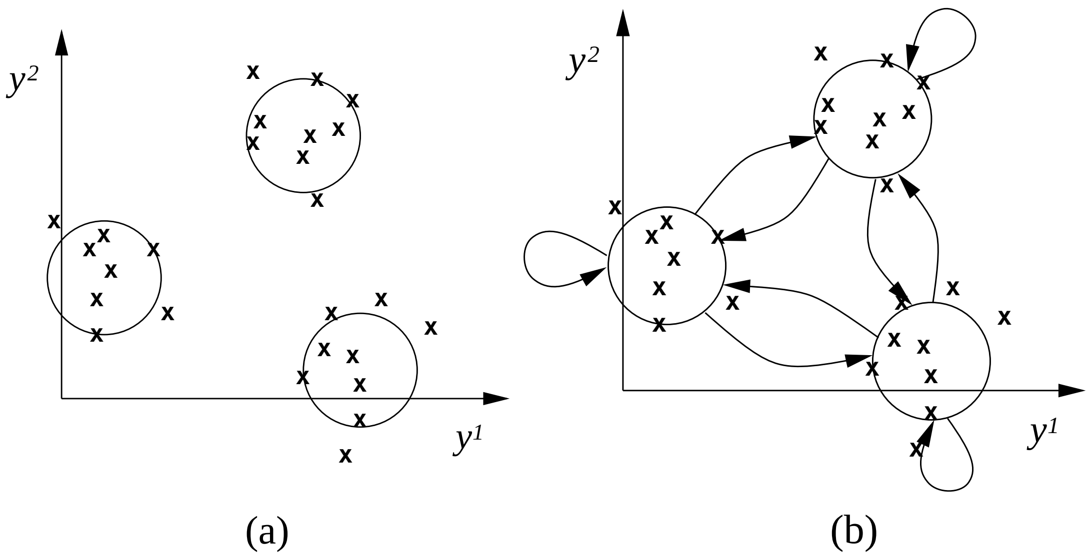

Figure 12.1: (a). A sample point is generated from a mixture model by first selecting a mixture component and then generating a data point from that mixture component. (b) An HMM generalizes the mixture model by allowing the choice of the mixture component at a given step to depend on the choice of the mixture component at the previous step. The arrows in the diagram represent these transitions between the mixture components.

justified in a given problem, it is not sufficient—there should also be regularities in the transitions between clusters.

The inference problem for HMMs involves taking as input the sequence of observed data and yielding as output a probability distribution on the underlying states. Given the dependence between the states, this problem is substantially more complex than the analogous inference problem for mixture models. Nonetheless, it is readily solved. Guided by Bayes rule, we will uncover a simple recursion that neatly computes the desired posterior probabilities. In fact, this algorithm marks an important milestone for us—it begins to suggest the general machinery for propagating probabilities on graphs that we will be our focus in much of the remainder of the book. With HMMs we begin our study of inference in graphical models in earnest.

12.1 The graphical model

The graphical model representation of an HMM is shown in Figure 12.2. As the diagram makes clear, the HMM can be viewed as a linked sequence of mixture models, with the linking occurring at the level of the mixture components, or "states." We denote the state at time t as q_t, and let y_t represent the observable "output" at time t.[1]

[1]Throughout the chapter we refer to t as a temporal variable for concreteness; the HMM model is of course applicable to any kind of sequential data.

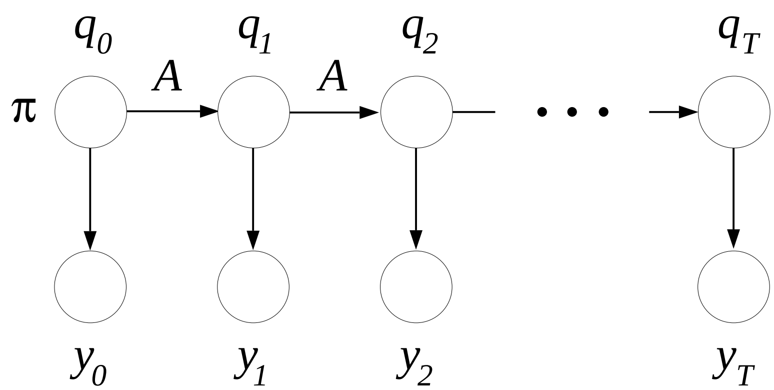

Figure 12.2: The representation of a HMM as a graphical model. Each vertical slice represents a time step. The top node in each slice represents the multinomial q_t variable and the bottom node in each slice represents the observable y_t variable.

We represent the state at time t as a multinomial random variable q_t, with components q_t^i, for $i = 0, \ldots, M$. Thus q_t^i is equal to one for a particular value of i and is equal to zero for $j \neq i$. As for the output variables y_t, these variables are always observed in the HMM setting and thus they play a minimal role in the inference problem. We will accordingly leave their type undefined for now (the reader can think of them as multinomial or multivariate Gaussian for concreteness).

From the graphical model we can read off various conditional independencies. The main conditional independency of interest is that obtained by conditioning on a single state node. Conditioning on q_t renders q_{t-1} and q_{t+1} independent; moreover it renders q_s independent of q_u, for $s < t$ and $t < u$. Thus, "the future is independent of the past, given the present." This statement is also true for output nodes y_s and y_u, again conditioning on the state node q_t.

Note that conditioning on an output node, on the other hand, does not separate nodes in the graph and thus does not yield any conditional independencies. It is not true that the future is independent of the past, given the present, if by "present" we mean the current output.

Indeed, conditioning on *all* of the output nodes fails to separate any of the remaining nodes. That is, given the observable data, we cannot expect any independencies to be induced between the state nodes. Thus we should expect that our inference algorithm must take into account possible dependencies between states at arbitrary locations along the chain. In particular, learning something about the final state node in the chain, q_T (e.g., by observing y_T), can change the posterior probability distribution for the first node in the chain, q_0. We expect that our inference algorithm will have to propagate information from one end of the chain to the other.

12.2 The parameterization

We now parameterize the HMM by assigning local conditional probabilities to each of the nodes. The first state node in the sequence has no parents; thus we endow this node with an unconditional distribution π, where $\pi^i \triangleq p(q_0^i = 1)$. Each successive state node has the previous state node in the chain as its (sole) parent; thus we need a $M \times M$ matrix to specify its local conditional probability. We define a *state transition matrix* A, where the (i,j)th entry a_{ij} of A is defined to be the transition probability $p(q^j_{t+1} = 1 | q^i_t = 1)$. Note that we assume that this transition probability is independent of t; that is, we assume a *homogeneous* HMM. (All of the algorithms that we describe are readily generalized to the case of a varying A matrix, however this case is less common in practice than the homogeneous case).

Each of the output nodes has a single state node as a parent, thus we require a probability distribution $p(y_t|q_t)$. We again assume this distribution to be independent of t. We make no further assumptions regarding the form of $p(y_t|q_t)$ for now; for the purposes of developing the HMM inference algorithms we need only be able to evaluate $p(y_t|q_t)$ for a fixed value of y_t.

The joint probability is obtained as always by taking the product over the local conditional probabilities. Thus, for a particular configuration $(q, y) = (q_0, q_1, \ldots, q_T, y_0, y_1, \ldots, y_T)$, we obtain the following joint probability:

$$p(q, y) = p(q_0) \prod_{t=0}^{T-1} p(q_{t+1}|q_t) \prod_{t=0}^{T} p(y_t|q_t). \tag{12.1}$$

To introduce the A and π parameters into this equation, we adopt a notation in which state variables can be used as indices. Thus, when q_t takes on its ith value and q_{t+1} takes on its jth value, we let $a_{q_t, q_{t+1}}$ denote the (i,j)th entry of the matrix A. Formally, this interpretation is achieved via the following definition:

$$a_{q_t, q_{t+1}} \triangleq \prod_{i,j=1}^{M} [a_{ij}]^{q^i_t q^j_{t+1}}. \tag{12.2}$$

Recall that only one of the components of q_t is one, and thus only one factor in the product on the right-hand side is different from one; this picks out the appropriate entry in the matrix A. Similarly, we define π_{q_0} via:

$$\pi_{q_0} \triangleq \prod_{i=1}^{M} [\pi_i]^{q^i_0} \tag{12.3}$$

which has the effect of picking out the appropriate entry in the π vector. We use the simple shorthand forms $a_{q_t, q_{t+1}}$ and π_{q_0} throughout the chapter, although the expanded forms in Eqs. 12.2 and 12.3 will also prove useful when we discuss parameter estimation.

Plugging the definitions into the joint probability, we have:

$$p(q, y) = \pi_{q_0} \prod_{t=0}^{T-1} a_{q_t, q_{t+1}} \prod_{t=0}^{T} p(y_t|q_t). \tag{12.4}$$

This is the parameterized probability distribution in which we wish to do inference.

12.3 The inference problem

There are quite a number of inference problems that are of interest in the setting of the HMM. The general inference problem involves computing the probability of a hidden state sequence q given an observable output sequence y. Various marginal probabilities are also of interest, in particular the probability of a particular hidden state q_t given the output sequence.

It is also of interest to compute various probabilities conditioned on partial output sequences. In particular, consider the "on-line" problem in which a sequence of outputs y_t arrives and it is desired to compute the probability of the state at time t immediately, without waiting for future data. Computing this probability, $p(q_t|y_0,\ldots,y_t)$, is generally called the *filtering problem*.[2] Another inference problem involves the calculation of $p(q_t|y_0,\ldots,y_s)$, where $t > s$. This is referred to as the *prediction problem*. Finally, the problem of calculating a posterior probability based on data up to and including a future time, i.e., $p(q_t|y_0,\ldots,y_u)$ for $t < u$, is referred to as the *smoothing problem*.

Let us consider the problem of computing the posterior probability $p(q|y)$ where $y = (y_0,\ldots,y_T)$ is the entire observed output sequence at our disposal. Let q be an arbitrary fixed state sequence whose probability we wish to compute. By definition we have $p(q|y) = p(q,y)/p(y)$. The numerator is readily calculated by substituting q and y in Eq. 12.4. What about the denominator $p(y)$?

Calculating the denominator involves taking a sum across all possible values of the hidden states:

$$p(y) = \sum_{q_0}\sum_{q_1}\cdots\sum_{q_T}\pi(q_0)\prod_{t=0}^{T-1}a_{q_t,q_{t+1}}\prod_{t=0}^{T}p(y_t|q_t,\eta). \tag{12.5}$$

This sum should give us pause. Each state node q_t can take on M values, and we have T state nodes. This implies that we must perform M^T sums, a wildly intractable number for reasonable values of M and T. Is it possible to perform inference efficiently for HMMs?

The way out of our seeming dilemma lies in the factorized form of the joint probability distribution (Eq. 12.4). Each factor involves only one or two of the state variables, and the factors form a neatly organized chain. This suggests that it ought to be possible to move these sums "inside" the product in a systematic way. Moving the sums as far as possible ought to reduce the computational burden significantly. Consider, for example, the sum over q_T. This sum can be brought inside until the end of the chain and applied to the two factors involving q_T. Once this sum is performed the result can be combined with the two factors involving q_{T-1} and the sum over q_{T-1} can be performed. We begin to hope that we can organize our calculation as a recursion.

12.4 Inference

To reveal the recursion behind the HMM inference problem as simply as possible, let us consider an inference problem that is seemingly easier than the full problem. Rather than calculating $p(q|y)$ for the entire state sequence q, we focus on a particular state node q_t and ask to calculate its posterior probability, that is, we calculate $p(q_t|y)$. This posterior probability also has $p(y)$ in its denominator,

[2] Why "filtering"? The terminology arises from the interpretation of the outputs y_t as providing "noisy" information about the underlying "signal" q_t. The inference problem is then one of "filtering" the noise from the signal. In the linear stochastic systems setting in which this terminology originally arose (cf. Chapter 15), the calculation of quantities such as $p(q_t|y_0,\ldots,y_t)$ often had a frequency domain interpretation in which some frequencies are passed and not others. In such a setting the terminology is rather natural. While recognizing the possible unnaturalness of the terminology outside of the linear systems setting, we bow to its wide usage and adopt it here.

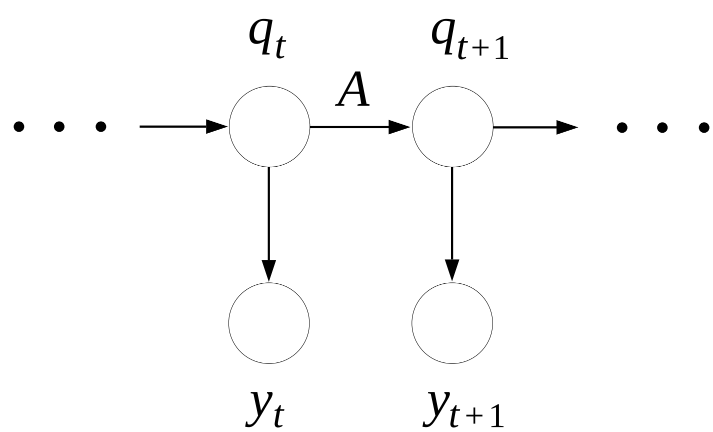

Figure 12.3: A fragment of the graphical model representation of an HMM.

and in fact we can easily adapt our algorithm for computing $p(q_t|y)$ to compute $p(q|y)$ or marginals over substrings of q (see Exercise **??**).

We thus turn to the calculation of $p(q_t|y)$. To make progress, we need to take advantage of the conditional independencies in our graphical model, breaking the problem into pieces. To do so we condition on a state node (see Figure 12.3). We reverse the terms q_t and y via an application of Bayes rule, conditioning now on q_t, and use conditional independence:

$$\begin{aligned} p(q_t|y) &= \frac{p(y|q_t)p(q_t)}{p(y)} &\quad (12.6) \\ &= \frac{p(y_0, \ldots, y_t|q_t)p(y_{t+1}, \ldots, y_T|q_t)p(q_t)}{p(y)}. &\quad (12.7) \end{aligned}$$

Finally, we regroup the terms and make a definition:

$$\begin{aligned} p(q_t|y) &= \frac{p(y_0, \ldots, y_t, q_t)p(y_{t+1}, \ldots, y_T|q_t)}{p(y)} &\quad (12.8) \\ &= \frac{\alpha(q_t)\beta(q_t)}{p(y)}, &\quad (12.9) \end{aligned}$$

where

$$\alpha(q_t) \triangleq p(y_0, \ldots, y_t, q_t) \tag{12.10}$$

is the probability of emitting a partial sequence of outputs $y_0, \ldots, y_t$ and ending up in state q_t, and

$$\beta(q_t) = p(y_{t+1}, \ldots, y_T|q_t) \tag{12.11}$$

is the probability of emitting a partial sequence of outputs $y_{t+1}, \ldots, y_T$ given that the system starts in state q_t.

Given that the sum of $p(q_t|y)$ over the possible values of q_t must equal one, we use Eq. 12.9 to obtain:

$$p(y) = \sum_{q_t} \alpha(q_t)\beta(q_t). \tag{12.12}$$

That is, we can obtain the likelihood $p(y)$ by calculating $\alpha(q_t)$ and $\beta(q_t)$ for any t and summing their product.

We make one additional definition: $\gamma(q_t)$ will denote the posterior probability $p(q_t|y)$. Thus:

$$\gamma(q_t) \triangleq \frac{\alpha(q_t)\beta(q_t)}{p(y)}, \tag{12.13}$$

where $p(y)$ is computed once, as the normalization constant for a particular (arbitrary) choice of t.

We have reduced our problem to that of calculating the alphas and the betas. This is a useful reduction because, as we now see, these quantities can be computed recursively.

Let us first consider the alpha variables. Given that $\alpha(q_t)$ depends only on quantities up to time t, and given the Markov properties of our model, we might hope to obtain a recursion between $\alpha(q_t)$ and $\alpha(q_{t+1})$. Indeed, referring to Figure 12.3 to justify the conditional independencies we need, we obtain:

$$\begin{aligned}
\alpha(q_{t+1}) &= p(y_0, \ldots, y_{t+1}, q_{t+1}) && (12.14)\\
&= p(y_0, \ldots, y_{t+1}|q_{t+1})p(q_{t+1}) && (12.15)\\
&= p(y_0, \ldots, y_t|q_{t+1})p(y_{t+1}|q_{t+1})p(q_{t+1}) && (12.16)\\
&= p(y_0, \ldots, y_t, q_{t+1})p(y_{t+1}|q_{t+1}) && (12.17)\\
&= \sum_{q_t} p(y_0, \ldots, y_t, q_t, q_{t+1})p(y_{t+1}|q_{t+1}) && (12.18)\\
&= \sum_{q_t} p(y_0, \ldots, y_t, q_{t+1}|q_t)p(q_t)p(y_{t+1}|q_{t+1}) && (12.19)\\
&= \sum_{q_t} p(y_0, \ldots, y_t|q_t)p(q_{t+1}|q_t)p(q_t)p(y_{t+1}|q_{t+1}) && (12.20)\\
&= \sum_{q_t} p(y_0, \ldots, y_t, q_t)p(q_{t+1}|q_t)p(y_{t+1}|q_{t+1}) && (12.21)\\
&= \sum_{q_t} \alpha(q_t)a_{q_t,q_{t+1}}p(y_{t+1}|q_{t+1}). && (12.22)
\end{aligned}$$

Throughout this derivation the key idea is to condition on a state and then use the conditional independence properties of the model to decompose the equation. This is done in Eqs. 12.16 and 12.22, both of which can be verified via the graphical model fragment. The second key idea is to introduce a variable, in this case q_t, by marginalizing over it (cf. Eq. 12.18). Once q_t is introduced the recursion follows readily.

The computational complexity of each step of the alpha recursion is $O(M^2)$; in particular, for each of the M values of q_{t+1}, we require M multiplications to compute the inner product of $\alpha(q_t)$

with the appropriate column of the A matrix. To compute all of the alpha variables from $t = 1$ to $t = T$ thus requires time $O(M^2T)$.

Note that the algorithm proceeds "forward" in time. The definition of alpha at the first time step yields:

$$\begin{aligned}
\alpha(q_0) &= p(y_0, q_0) && (12.23)\\
&= p(y_0|q_0)p(q_0) && (12.24)\\
&= p(y_0|q_0)\pi_{q_0}. && (12.25)
\end{aligned}$$

and these values are used to initialize the recursion.

For the beta variables we obtain a "backward" recursion in which $\beta(q_t)$ is expressed in terms of $\beta(q_{t+1})$, where once again the various steps are justified by making reference to the graphical model fragment in Figure 12.3:

$$\begin{aligned}
\beta(q_t) &= p(y_{t+1}, \ldots, y_T|q_t) && (12.26)\\
&= \sum_{q_{t+1}} p(y_{t+1}, \ldots, y_T, q_{t+1}|q_t) && (12.27)\\
&= \sum_{q_{t+1}} p(y_{t+1}, \ldots, y_T|q_{t+1}, q_t)p(q_{t+1}|q_t) && (12.28)\\
&= \sum_{q_{t+1}} p(y_{t+2}, \ldots, y_T|q_{t+1})p(y_{t+1}|q_{t+1})p(q_{t+1}|q_t) && (12.29)\\
&= \sum_{q_{t+1}} \beta(q_{t+1})a_{q_t,q_{t+1}}p(y_{t+1}|q_{t+1}). && (12.30)
\end{aligned}$$

Note that the beta recursion is a backwards recursion; that is, we start at the final time step T and proceed backwards to the initial time step.

As for the initialization of the beta recursion, the definition of $\beta(q_T)$ is unhelpful, given that it makes reference to a non-existent y_{T+1}, but we see from applying the recursion once to compute $\beta(q_{T-1})$ that this value will be calculated correctly if we define $\beta(q_T)$ to be a vector of ones. Alternatively, computing $p(y)$ at time T, we have:

$$\begin{aligned}
p(y) &= \sum_i \alpha(q_T^i)\beta(q_T^i) && (12.31)\\
&= \sum_i \alpha(q_T^i) && (12.32)\\
&= \sum_i p(y_0, \ldots, y_T, q_T^i) && (12.33)\\
&= p(y), && (12.34)
\end{aligned}$$

and we see that the definition makes sense.

If we need only the likelihood $p(y)$, Eq. 12.31 shows us that it is not necessary to compute the betas; a single forward pass for the alphas will suffice. Moreover, Eq. 12.12 tell us that any partial forward pass up to time t to compute $\alpha(q_t)$, accompanied by a partial backward pass to compute $\beta(q_t)$, will also suffice. To compute the posterior probabilities for all of the states q_t, however, requires us to compute alphas and betas for each time step. Thus we require a forward pass and a backward pass for a complete solution to the inference problem.

12.5 An alternative inference algorithm

The alpha-beta algorithm is not the only way to compute the posterior probabilities of the states. In this section we describe an alternative approach in which the backward phase is a recursion defined directly on the $\gamma(q_t)$ variables. An interesting feature of this algorithm is that the backward phase makes no use of the observations y_t; only the forward phase uses the observed data. We can throw away the data as we filter.

The algorithm differs from the alpha-beta algorithm only in the backward phase. In the forward direction we run the alpha algorithm as before, calculating the filtered quantities $\alpha(q_t) = p(y_0, \ldots, y_t, q_t)$.

To uncover a backward recursion linking the γ_t variables, we refer once again to the graphical model fragment in Figure 12.3. Our goal is to compute $\gamma(q_t) = p(q_t|y_0, \ldots, y_T)$. As in our earlier calculations, our main tool for computing such quantities recursively is to condition on a state variable; such conditioning breaks the problem into two pieces. In particular, we condition on q_{t+1} and obtain:

$$p(q_t|q_{t+1}, y_0, \ldots, y_T) = p(q_t|q_{t+1}, y_0, \ldots, y_t). \tag{12.35}$$

This shows that we can get a conditional probability that depends on all of the data via a conditional probability that depends only on the partial sequence up to t. Moreover, the left-hand side can be readily converted into $\gamma(q_t)$ by multiplying by $p(q_{t+1}|y_0, \ldots, y_T)$—which is $\gamma(q_{t+1})$ by definition—and summing over q_{t+1}. The details are as follows:

$$\begin{aligned}
\gamma(q_t) &= \sum_{q_{t+1}} p(q_t, q_{t+1}|y_0, \ldots, y_T) && (12.36)\\
&= \sum_{q_{t+1}} p(q_t|q_{t+1} y_0, \ldots, y_T) p(q_{t+1}|y_0, \ldots, y_T) && (12.37)\\
&= \sum_{q_{t+1}} p(q_t|q_{t+1}, y_0, \ldots, y_t) p(q_{t+1}|y_0, \ldots, y_T) && (12.38)\\
&= \sum_{q_{t+1}} \frac{p(q_t, q_{t+1}, y_0, \ldots, y_t)}{\sum_{q_t} p(q_t, q_{t+1}, y_0, \ldots, y_t)} p(q_{t+1}|y_0, \ldots, y_T) && (12.39)\\
&= \sum_{q_{t+1}} \frac{p(q_t, y_0, \ldots, y_t) p(q_{t+1}|q_t)}{\sum_{q_t} p(q_t, y_0, \ldots, y_t) p(q_{t+1}|q_t)} p(q_{t+1}|y_0, \ldots, y_T) && (12.40)\\
&= \sum_{q_{t+1}} \frac{\alpha(q_t) a_{q_t, q_{t+1}}}{\sum_{q_t} \alpha(q_t) a_{q_t, q_{t+1}}} \gamma(q_{t+1}) && (12.41)
\end{aligned}$$

We see that this recursion makes use of the alpha variables, which therefore must be computed before the gamma recursion begins. The gamma recursion is initialized with $\gamma(q_T) = \alpha(q_T)$.

Note that the data y_t are not referenced in the gamma recursion; the alpha recursion has absorbed all of the necessary data likelihoods.

12.6 The $\xi(q_t, q_{t+1})$ variables

The alpha-beta or the alpha-gamma algorithm provide us with the posterior probability of the hidden states of the HMM. These quantities are the direct analogs of the posterior probabilities h_i that we studied in the simpler mixture setting. Moreover, they play the same role in estimating the parameters of the output distribution—as we will see in Section 12.8 they are the expected sufficient statistics for these parameters. To estimate the transition probability matrix A, however, we need something more. It is clear intuitively, and justified in Section 12.8, where we write out the complete log likelihood, that what is required is the matrix of cooccurrence probabilities $p(q_t, q_{t+1}|y)$. In this section we show how to calculate these posterior probabilities.

Let us define

$$\xi(q_t, q_{t+1}) \triangleq p(q_t, q_{t+1}|y). \tag{12.42}$$

There are several ways to calculate this quantity. One way is to return to first principles and develop recursions for the $\xi(q_t, q_{t+1})$, following much the same procedure as we followed for the singleton probabilities $\gamma(q_t)$. This is indeed a rather useful exercise (which we ask the reader to carry out in Exercise **??**), not only because it reinforces the Markovian calculations that we have engaged in, but because it provides a stepping-stone to the general "junction tree algorithm" that we discuss in Chapter 17. That algorithm provides a general framework from which to derive all of the recursions that we describe in this chapter. Moreover, if we have an algorithm for calculating $\xi(q_t, q_{t+1})$ in hand, we can also obtain the singleton probabilities via $\gamma(q_t) = \sum_{q_{t+1}} \xi(q_t, q_{t+1})$.

A second approach to calculating $\xi(q_t, q_{t+1})$ is to build on the recursions already developed for the alphas and betas:

$$\begin{aligned}
\xi(q_t, q_{t+1}) &= p(q_t, q_{t+1}|y) && (12.43)\\
&= \frac{p(y|q_t, q_{t+1})p(q_{t+1}|q_t)p(q_t)}{p(y)} \\
&= \frac{p(y_0, \ldots, y_t|q_t)p(y_{t+1}|q_{t+1})p(y_{t+2}, \ldots, y_T|q_{t+1})p(q_{t+1}|q_t)p(q_t)}{p(y)} \\
&= \frac{\alpha(q_t)p(y_{t+1}|q_{t+1})\beta(q_{t+1})a_{q_t,q_{t+1}}}{p(y)}. && (12.44)
\end{aligned}$$

This result can also be expressed in terms of alphas and gammas:

$$\xi(q_t, q_{t+1}) = \frac{\alpha(q_t)p(y_{t+1}|q_{t+1})\gamma(q_{t+1})a_{q_t,q_{t+1}}}{\alpha(q_{t+1})}. \tag{12.45}$$

In either case we see that we can readily calculate the $\xi(q_t, q_{t+1})$ variables once we have finished the recursive calculation of the singleton probabilities $\gamma(q_t)$.

12.7 Numerical issues

To summarize, we have found that we can calculate all of the necessary posterior probabilities for the HMM recursively. Given an observed sequence y, we run the alpha recursion forward in time. If we require only the likelihood we simply sum the alphas at the final time step. If we also require the posterior probabilities of the states, we proceed to either the beta recursion or the gamma recursion.

Before these recursions are implemented on the computer, attention must be paid to numerical issues. In particular, the recursions involve repeated multiplications of small numbers and it is generally not long before the numbers underflow. To avoid underflow it suffices to normalize. We outline the basic ideas in Exercise **??**, but in brief the procedure is as follows. The alpha variables, $p(y_0, \ldots, y_t, q_t)$, can be viewed as unnormalized conditional probabilities. Indeed, normalizing means division by $p(y_0, \ldots, y_t)$, which yields conditionals $p(q_t | y_0, \ldots, y_t)$. Not only are these conditionals scaled in a numerically sensible manner, but they also have a sensible semantics—they are the filtered estimates of the states. In sum, one should always compute normalized alphas. In the backward direction, if one uses the gamma recursion one is already on safe ground—the gammas are conditional probabilities and hence sum to one. Moreover, it is easy to verify that normalized alphas can be used in Eq. 12.41 in place of the unnormalized alphas. Alternatively, if one uses the beta recursion, it turns out that a numerically sensible solution (although one that is devoid of probabilistic interpretation), is to use the normalization factors from the forward recursion to rescale the beta variables (these rescaled betas will not sum to one). It turns out that the rescaled variables are then used exactly as the original alphas and betas are used in the formulas for the posteriors $\gamma(q_t)$ and $\xi(q_t, q_{t+1})$ (i.e., the normalization factors cancel). See Exercise **??** for further discussion.

12.8 Parameter estimation

The parameters of an HMM are the transition matrix A, the initial probability distribution π and the parameters that are associated with the output probability distribution. In this section we discuss the problem of estimating these parameters from data.

Let $\theta = (\pi, A, \eta)$ represent all of the parameters of the HMM model, where $p(y_t | q_t, \eta)$ is the output distribution. The likelihood is given by $p(y | \theta)$, for a fixed observable sequence y. Taking the logarithm of Eq. 12.5 we have the following log likelihood:

$$p(y|\theta) = \log \sum_{q_0} \sum_{q_1} \cdots \sum_{q_T} \pi(q_0) \prod_{t=0}^{T-1} a_{q_t, q_{t+1}} \prod_{t=0}^{T} p(y_t | q_t, \eta). \tag{12.46}$$

Our goal is to maximize this expression with respect to θ.

As with our earlier models, this is in principle just another optimization problem that can be solved via standard numerical optimization methods. In practice, however, the EM algorithm is generally used to estimate HMM parameters.

12.8.1 EM algorithm

The EM algorithm for the HMM presents no new difficulties to surmount and we will make relatively short work of the derivation. For concreteness we derive the algorithm for the case in which the outputs y_t are multinomial variables; it should be obvious how to change the derivation to accommodate other output types (cf. Exercise **??**).

In the multinomial case, y_t is an N-component vector such that y_t^j is equal to one for a particular component and zero for all other components. We use the symbol η_{ij} to denote the probability that the jth component of y_t is one, given that the ith component of q_t is one; i.e., $\eta_{ij} \triangleq p(y_t^j = 1 | q_t^i = 1, \eta)$. Using this notation we have:

$$p(y_t|q_t, \eta) = \prod_{i,j=1}^{M} [\eta_{ij}]^{q_t^i y_t^j} \tag{12.47}$$

as the general expression for the output distribution.

As usual we begin by writing down the complete log likelihood to discover the form of the M step estimates as well as the sufficient statistics that are needed for the E step. We have:

$$\begin{aligned}
\log p(q, y) &= \log \left\{ \pi_{q_0} \prod_{t=0}^{T-1} a_{q_t, q_{t+1}} \prod_{t=0}^{T} p(y_t|q_t, \eta) \right\} && (12.48)\\
&= \log \left\{ \prod_{i=1}^{M} [\pi_i]^{q_0^i} \prod_{t=0}^{T-1} \prod_{i,j=1}^{M} [a_{ij}]^{q_t^i q_{t+1}^j} \prod_{t=0}^{T} \prod_{i,j=1}^{M} [\eta_{ij}]^{q_t^i y_t^j} \right\} && (12.49)\\
&= \sum_{i=1}^{M} q_0^i \log \pi_i + \sum_{t=0}^{T-1} \sum_{i,j=1}^{M} q_t^i q_{t+1}^j \log a_{ij} + \sum_{t=0}^{T} \sum_{i,j=1}^{M} q_t^i y_t^j \log \eta_{ij}. && (12.50)
\end{aligned}$$

From this expression, we see that $m_{ij} \triangleq \sum_{t=0}^{T-1} q_t^i q_{t+1}^j$ is the sufficient statistic for a_{ij}, $n_{ij} \triangleq \sum_{t=0}^{T-1} q_t^i y_t^j$ is the sufficient statistic for η_{ij}, and q_0^i is the sufficient statistic for π_i. The maximum likelihood estimates for the case of complete data are therefore given by:

$$\begin{aligned}
\hat{a}_{ij} &= \frac{m_{ij}}{\sum_{k=1}^{M} m_{ik}} && (12.51)\\
\hat{\eta}_{ij} &= \frac{n_{ij}}{\sum_{k=1}^{N} n_{ik}} && (12.52)\\
\hat{\pi}_i &= q_0^i. && (12.53)
\end{aligned}$$

All of these estimates have natural interpretations. Note that m_{ij} is the count of the number of times that the process is in state i and transitions to state j. Dividing by $\sum_k m_{ik}$ yields the proportion of those transitions out of state i that go to state j—a natural estimate of a_{ij}. Similarly the estimate of η_{ij} is given by the proportion of times that the chain is in state i and produces the

jth output value. Finally, for the estimate of π_i, we obtain a singular distribution that puts all of the probability mass at the observed initial state.[3]

We turn to the E step of the EM algorithm. Consider first the expectation of the sufficient statistic $n_{ij} = \sum_{t=0}^{T} q_t^i y_t^j$. We have:

$$\begin{aligned} E(n_{ij}|y, \theta^{(p)}) &= \sum_{t=0}^{T} E(q_t^i|y, \theta^{(p)}) y_t^j && (12.54) \\ &= \sum_{t=0}^{T} p(q_t^i = 1|y, \theta^{(p)}) y_t^j && (12.55) \\ &\triangleq \sum_{t=0}^{T} \gamma_t^i y_t^j, && (12.56) \end{aligned}$$

where we introduce the notation γ_t^i in the last line. By definition γ_t^i is equal to $\gamma(q_t)$, evaluated at that value of q_t such that $q_t^i = 1$. Note finally that the dependence of γ_t^i on $\theta^{(p)}$ has been suppressed.

Similarly, for the sufficient statistic m_{ij}, we have:

$$\begin{aligned} E(m_{ij}|y, \theta^{(p)}) &= \sum_{t=0}^{T-1} E(q_t^i q_{t+1}^j|y, \theta^{(p)}) && (12.57) \\ &= \sum_{t=0}^{T-1} p(q_t^i q_{t+1}^j|y, \theta^{(p)}) && (12.58) \\ &\triangleq \sum_{t=0}^{T-1} \xi_{t,t+1}^{ij}, && (12.59) \end{aligned}$$

where we let $\xi_{t,t+1}^{ij}$ denote $\xi(q_t, q_{t+1})$ for (q_t, q_{t+1}) such that $q_t^i = 1$ and $q_{t+1}^j = 1$.

In summary, the sufficient statistics are calculated via the recursive forward-backward procedure from Section 12.4. We calculate the γ variables via either Eq.12.13 or Eq.12.41. The ξ variables are then calculated via Eq. 12.44 or Eq. 12.45.

With the estimated sufficient statistics in hand, we substitute into the maximum likelihood formulas (Eqs. 12.51, 12.52, and 12.53), to obtain the M step of the EM algorithm (also known, in the case of HMMs, as the "Baum-Welch updates"). We obtain:

$$\hat{\eta}_{ij}^{(p+1)} = \frac{\sum_{t=0}^{T} \gamma_t^i y_t^j}{\sum_{k=1}^{N} \sum_{t=0}^{T} \gamma_t^i y_t^k} = \frac{\sum_{t=0}^{T} \gamma_t^i y_t^j}{\sum_{t=0}^{T} \gamma_t^i}. \tag{12.60}$$

[3] In a more general setting, in which we have multiple repeated observations from a single HMM (i.e., data that are IID at the level of entire sequences), the estimate of π_i becomes the proportion of times that the chain starts in state i, and indeed the estimates in Eq. 12.51 and Eq. 12.52 also become averages over the multiple repetitions (cf. Exercise **??**).

where we use the fact that $\sum_{k=1}^{N} y_t^k = 1$,

$$\hat{a}_{ij}^{(p+1)} = \frac{\sum_{t=0}^{T-1} \xi_{t,t+1}^{i,j}}{\sum_{k=1}^{M} \sum_{t=0}^{T-1} \xi_{t,t+1}^{i,k}} = \frac{\sum_{t=0}^{T-1} \xi_{t,t+1}^{i,j}}{\sum_{t=0}^{T-1} \gamma_t^i}, \tag{12.61}$$

where by definition $\sum_{k=1}^{M} \xi_{t,t+1}^{i,k} = \gamma_t^i$, and:

$$\hat{\pi}_i^{(p+1)} = \gamma_0^i. \tag{12.62}$$

The EM algorithm iterates between performing these updates (the M step) and the forward-backward pass using the updated values (the E step).

12.9 Historical remarks and bibliography