An Introduction to Graphical Models and Machine Learning

Michael I. Jordan
University of California, Berkeley

Christopher M. Bishop
Microsoft Research

Hidden Markov Models

This is a lightly edited version of a chapter in a book being written by Jordan and Bishop. Since this is a draft, please do not distribute this to anyone who is not a student of CS 188 this term.

The model

A hidden Markov model is characterized by a set of M states, by an initial probability distribution for the first state, by a transition probability matrix linking successive states, and by a state-dependent probability distribution on the outputs.

We represent the state at time t as a multinomial random variable q_t, with components q_t^i, for $i = 1, \ldots, M$.[1] Thus q_t^i is equal to one for a particular value of i and is equal to zero for $j \neq i$. We use a subscript to denote the time step, thus q_t is the multinomial state at time t. The transition probability matrix A is the probability of transitioning between the multinomial states at successive time steps; in particular, the (i, j)th entry a_{ij} is the transition probability $P(q_{t+1}^j = 1 | q_t^i = 1)$. Note that we assume that this transition probability is constant as a function of t; that is, we assume a *homogeneous* hidden Markov model. 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.

We also need an initial condition. The vector π represents the probability distribution on the initial state; in particular, we have $\pi_i = P(q_1^i = 1)$.

There are three related graphical representations of hidden Markov models that it is important to distinguish. The first representation, shown in Figure 1, is the *stochastic automaton*. In this diagram, the components of the multinomial state are shown as separate nodes and the arcs represent the transition probabilities. This diagram is *not* a graphical model; in particular there are cycles in the graph and the arcs do not represent assertions of conditional independence. The diagram is useful, however, as an explicit representation of the one-step dynamics of the HMM.

To represent a sequence, we can "unroll" the automaton in space, copying each of the state component nodes at each time step. This yields the *lattice diagram* shown in Figure 2, Notice that we have also

[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.

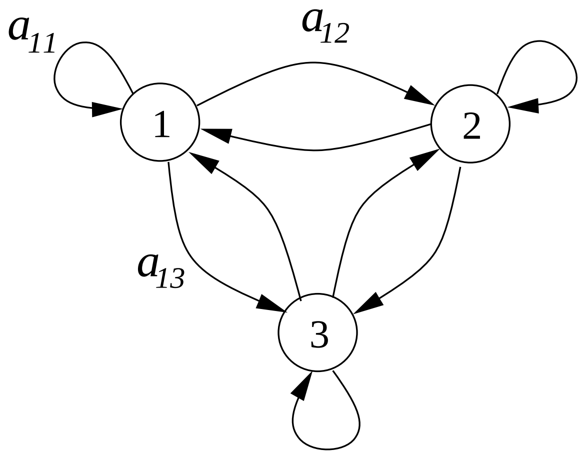

Figure 1: A representation of a three-state HMM as a stochastic automaton. The states are labeled with integers, which correspond to the three components of the multinomial q_t variable. We have shown the transition probabilities a_{1j} associated with transitions from the first state.

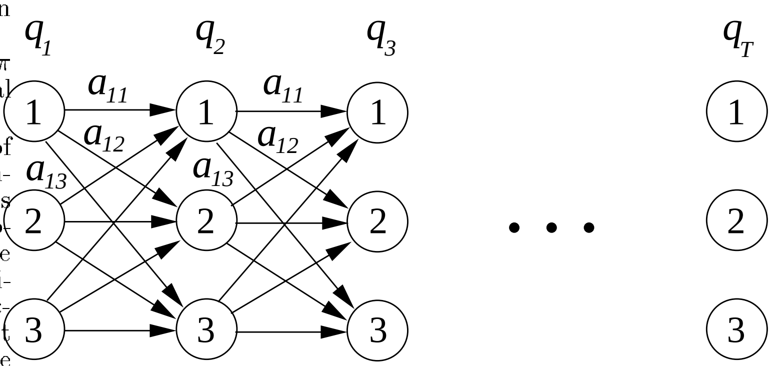

Figure 2: A representation of a three-state HMM as a lattice. Each vertical slice represents a time step, and the three nodes represent the components of the multinomial q_t variable. The states are labeled with integers, which correspond to the three components of the q_t variable. We have shown the transition probabilities a_{1j} associated with transitions from the first state.

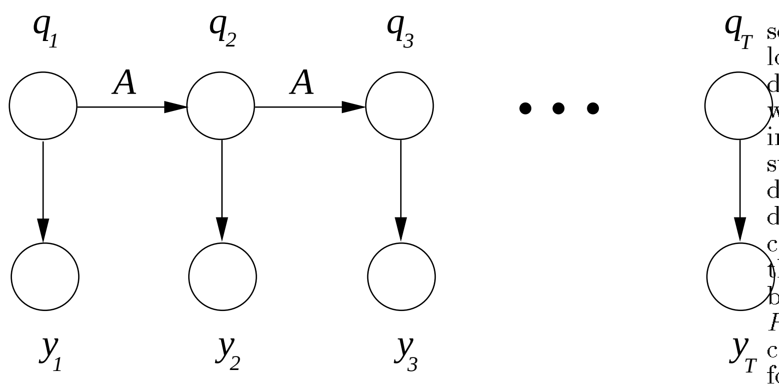

Figure 3: A representation of a three-state HMM as a graphical model. Each vertical slice represents a time step. The top node represents the multinomial q_t variable and the bottom node represents the observable y_t variable. Note that the components of q_t are suppressed in this diagram. The transition probabilities a_{ij} are the components of the A matrix.

represented the output sequence in this diagram, as a sequence of nodes that depends via the vertical links in the diagram on the choice of state at each step. The lattice diagram is useful for some purposes, in particular for understanding the recursive inference algorithms that we discuss below. The diagram contains more detail than we generally care to see, however, and moreover it is unable to represent the critical fact that only one of the state nodes can be "on" at each step; i.e., that the nodes are components of a multinomial random variable.

We obtain the third representation of the HMM by grouping the state component nodes of the lattice diagram into a single multinomial state node at each step. This diagram, shown in Figure 3, is the standard graphical model representation of an HMM. The diagram hides the numerical detail associated with the transition matrix, and reveals the conditional independence assumptions behind the HMM. The diagram makes clear that the HMM can be viewed as a linked sequence of mixture models (cf. Figure X), with the linking occurring at the level of the mixture components, or "states."

Focusing on the graphical representation of an HMM in Figure 3, let us now consider assigning a joint probability distribution to the HMM. As always we must assign local conditional probabilities to each of the nodes, conditioning on each nodes' parents. The first state node in the sequence has no parents and thus requires an unconditional distribution; the initial probability distribution π. Each subsequent state node has a single previous state node as its (sole) parent, and thus requires a MxM matrix to specify its local conditional probability. This matrix is the state transition matrix A.

The observable or *output* nodes are attached to the state nodes. We denote the tth node as y_t and denote the sequence of output nodes as y. Each output node has a single state node as a parent, thus we require a set of M probability distributions to characterize the local conditional probability of an output node. We denote these distributions generically as $P(y_t|q_t, \eta)$, where η is a parameter vector. In our discussion of inference we will not need to make any particular assumptions about the functional form of this local conditional probability nor indeed about the type of random variables represented by the output nodes (they can be discrete-valued, continuous-valued, or mixed—the choice depending on the particular kind of data being modeled). We need only be able to evaluate $P(y_t|q_t, \eta)$ for a fixed value of y_t. Later, when we discuss the parameter estimation problem, the functional form will of course become highly relevant, and at that point we will discuss particular choices for $P(y_t|q_t, \eta)$.

Conditional independencies

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_1. We expect that our inference algorithm will have to propagate information from one end of the chain to the other.

Joint probabilities and conditional probabilities

Let us now assemble the local conditional probabilities into a joint probability distribution. As always, the joint probability is obtained by taking a product over the local conditional probabilities. Thus, for a particular sample point $(q, y) = (q_1, q_2, \ldots, q_T, y_1, y_2, \ldots, y_T)$, we obtain the following joint probability:

$$P(q, y) = \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t, q_{t+1}} \prod_{t=1}^{T} P(y_t|q_t, \eta). \qquad (1)$$

In writing this equation we have introduced a convention under which the state variables are allowed to be used as subscripts. Formally, we interpret this shorthand as follows:

$$a_{q_t,q_{t+1}} \equiv \prod_{i,j=1}^{M} [a_{ij}]^{q_t^i q_{t+1}^j}. \quad (2)$$

Given that only one of the components of q_t is one, only one term on the right-hand side is different from one, and we see that we obtain $a_{q_t,q_{t+1}} = P(q_{t+1}, q_t)$ as desired. Similarly,

$$\pi_{q_1} \equiv \prod_{i=1}^{M} [\pi_i]^{q_1^i} \quad (3)$$

is justified as the definition of π_{q_1}. Although we use the shorthand throughout the chapter, we will also find use for the expanded forms in Eqs. 2 and 3 in the section on parameter estimation.

The inference problem involves computing the probability of a hidden state vector q given an observable output y. That is, we are required to compute the probability $P(q|y)$:

$$P(q|y) = \frac{P(q,y)}{P(y)} \quad (4)$$

Calculating the numerator in this expression simply involves substituting the particular values of q and y into Eq. 1. Calculating the denominator, on the other hand, involves computing a sum across all possible values of the hidden states:

$$P(y) = \sum_{q_1}\sum_{q_2}\cdots\sum_{q_T} \pi(q_1) \prod_{t=1}^{T-1} a_{q_t,q_{t+1}} \prod_{t=1}^{T} P(y_t|q_t,\eta). \quad (5)$$

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. 1). 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.

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

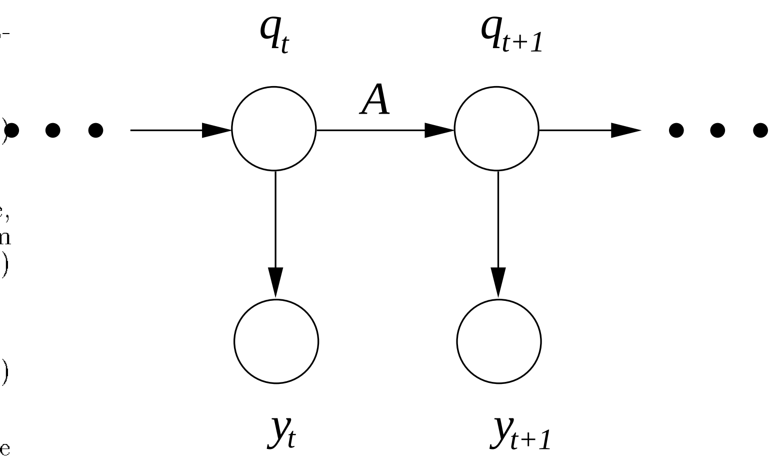

Figure 4: A fragment of the graphical model representation of an HMM.

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–analogous to the posterior probability over mixture components that we encountered for mixture models–will turn out to play a key role in our solution to the parameter estimation problem. Moreover, calculating this conditional probability in fact solves the full inference problem. To see this, note that $P(q_t|y)$ has $P(y)$ in its denominator and thus has the same core complexity as the full inference problem. Indeed, given that the calculation of the numerator $P(q,y)$ for the full inference problem (Eq. 4 involves no sums (we simply chain through the nodes and compute the product in Eq. 1), calculating the denominator in this (seemingly) simpler problem suffices.

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, and to do so we need to condition on a state node. This is achieved by reversing the terms q_t and y via an application of Bayes rule:

$$P(q_t|y) = \frac{P(y|q_t)P(q_t)}{P(y)}.$$

We now use conditional independence:

$$P(q_t|y) = \frac{P(y_1,\ldots,y_t|q_t)P(y_{t+1},\ldots,y_T|q_t)P(q_t)}{P(y)}.$$

(Verifying this equation is best done by inspecting the graphical model fragment in Figure 4 and observing that the separation properties of the graph correspond to the factorization in the equation). Finally, we regroup the terms and make a definition:

$$\begin{aligned} P(q_t|y) &= \frac{P(y_1,\ldots,y_t,q_t)P(y_{t+1},\ldots,y_T|q_t)}{P(y)} \\ &= \frac{\alpha(q_t)\beta(q_t)}{P(y)}, \end{aligned}$$

where
$$\alpha(q_t) \equiv P(y_1, \ldots, y_t, q_t)$$
is the probability of emitting a partial sequence of outputs $y_1, \ldots, y_t$ and ending up in state q_t, and

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

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.

Note that both $\alpha(q_t)$ and $\beta(q_t)$ are vectors, with components $\alpha(q_t^i)$ and $\beta(q_t^i)$. Moreover, given that the sum of $P(q_t|y)$ over the components of q_t must equal one, we use Eq. 6 to obtain:

$$P(y) = \sum_i \alpha(q_t^i)\beta(q_t^i). \tag{6}$$

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:

$$\gamma(q_t) \equiv \frac{\alpha(q_t)\beta(q_t)}{P(y)}, \tag{7}$$

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 we have the following:

$$\begin{aligned}
\alpha(q_{t+1}) &= P(y_1, \ldots, y_{t+1}, q_{t+1}) && (8)\\
&= P(y_1, \ldots, y_{t+1}|q_{t+1})P(q_{t+1}) && (9)\\
&= P(y_1, \ldots, y_t|q_{t+1})P(y_{t+1}|q_{t+1})P(q_{t+1}) && (10)\\
&= P(y_1, \ldots, y_t, q_{t+1})P(y_{t+1}|q_{t+1}) && (11)\\
&= \sum_{q_t} P(y_1, \ldots, y_t, q_t, q_{t+1})P(y_{t+1}|q_{t+1}) && (12)\\
&= \sum_{q_t} P(y_1, \ldots, y_t, q_{t+1}|q_t)P(q_t)P(y_{t+1}|q_{t+1}) && (13)\\
&= \sum_{q_t} P(y_1, \ldots, y_t|q_t)P(q_{t+1}|q_t)P(q_t)P(y_{t+1}|q_{t+1}) && (14)\\
&= \sum_{q_t} P(y_1, \ldots, y_t, q_t)P(q_{t+1}|q_t)P(y_{t+1}|q_{t+1}) && (15)\\
&= \sum_{q_t} \alpha(q_t)a_{q_t,q_{t+1}}P(y_{t+1}|q_{t+1}). && (16)
\end{aligned}$$

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. 10 and 16, both of which can be verified by examining the separation properties of the graphical model fragment in Figure 4. The second key idea is to introduce a variable, in this case q_t, by marginalizing over it (cf. Eq. 12). Once q_t is introduced the recursion follows readily.

The lattice diagram helps to clarify the computation of the alpha variables. Assuming that we have stored the vector $\alpha(q_t)$ at the tth layer of nodes in diagram, we calculate each component of $\alpha(q_{t+1})$ by considering all of the paths arriving at the corresponding node at slice $t+1$ in the diagram. The probabilities $\alpha(q_t)$ represent the probabilities of arriving at a particular state in slice t, having generated the partial output sequence $y_1, \ldots, y_t$. To evaluate the alpha vector at time $t+1$ we sum over all paths from the nodes at time t, weighted by the transition probabilities $a_{q_t,q_{t+1}}$. We then extend the output sequence by multiplying by $P(y_{t+1}|q_{t+1})$. The calculation requires $O(M^2)$ operations—for each of the M state components at time $t+1$, we require M multiplications using the alpha variables from time t. Once the vector $\alpha(q_{t+1})$ has been calculated, it replaces the vector $\alpha(q_t)$. Thus the storage requirements of the algorithm remain constant in time. Note that the algorithm proceeds "forward" in time, from the initial time step to time step T.

For the beta variables we obtain a "backward" recursion by expressing $\beta(q_t)$ in terms of $\beta(q_{t+1})$:

$$\begin{aligned}
\beta(q_t) &= P(y_{t+1}, \ldots, y_T|q_t) && (17)\\
&= \sum_{q_{t+1}} P(y_{t+1}, \ldots, y_T, q_{t+1}|q_t) && (18)\\
&= \sum_{q_{t+1}} P(y_{t+1}, \ldots, y_T|q_{t+1}, q_t)P(q_{t+1}|q_t) && (19)\\
&= \sum_{q_{t+1}} P(y_{t+2}, \ldots, y_T|q_{t+1})P(y_{t+1}|q_{t+1})P(q_{t+1}|q_t) && (20)\\
&= \sum_{q_{t+1}} \beta(q_{t+1})a_{q_t,q_{t+1}}P(y_{t+1}|q_{t+1}) && (21)\\
& && (22)
\end{aligned}$$

where the various steps involving conditional independence are again clarified by making reference to the graphical model fragment in Figure 4. 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.

We also must specify the initial conditions for the recursions. For the alpha recursion, the definition of alpha at the first time step yields:

$$\begin{aligned}
\alpha(q_1) &= P(y_1, q_1) && (23)\\
&= P(y_1|q_1)P(q_1) && (24)\\
&= P(y_1|q_1)\pi_{q_1}. && (25)
\end{aligned}$$

As for 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 the beta recursion that $\beta(q_{T-1})$

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:

$$\begin{aligned} P(y) &= \sum_i \alpha(q_T^i)\beta(q_T^i) && (26) \\ &= \sum_i \alpha(q_T^i) && (27) \\ &= \sum_i P(y_1, \ldots, y_T, q_T^i) && (28) \\ &= P(y), && (29) \end{aligned}$$

and we see that the definition makes sense.

If we need only the likelihood $P(y)$, Eq. 26 shows us that it is not necessary to compute the betas; a single forward pass for the alphas will suffice. Moreover, Eq. 6 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.

To summarize our discussion of inference, we have uncovered a pair of recursions that provide us with the probabilities that we need. 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, we proceed to the beta recursion, which is run backward in time. The alphas and betas are then substituted into Eq. 7 to calculate the $\gamma(q_t)$ posteriors.