Chapter 4 Sparse Hidden Markov Models

The **hidden Markov model** (HMM) is one of the most successful and widely used generative models in the field of statistical modeling. The statistical theory of HMMs has been driven in large part by the field of speech processing and is extremely well worked-out. Indeed, the Baum-Welch algorithm of the sixties is one of the earlier examples of an implementation of an EM algorithm, and much of the theory of EM was well understood in this context well before the publication of the general formulation. Nevertheless, advances in the theory of HMMs are still made. Recent examples include the factorial hidden Markov model Ghahramani and Jordan (1997).

In this chapter we review the generative model underlying the HMM, and discuss the applicable EM learning algorithm. We then examine a particular sub-class of the general model, the sparse HMM, in which the majority of outputs are zeros (or null). We then consider a "mixture" of these restricted models. This mixture-like compound model is a special case of the factorial HMM: we construct an EM algorithm with an imperfect E-step, of the form that was justified in section 1.8. This approach, though not exact, will come close to the true the maximum likelihood solution for certain classes of data.

4.1 The Generative Model

4.1.1 The Markov chain

The finite **Markov chain** (or Markov process) has been extensively studied in stochastic process theory. It consists of a series of N identically distributed discrete variables $\{y_i\}$, with the property that each is dependent only on the value of the preceding one. More precisely, the joint distribution over the variables factors as follows.

$$\mathsf{P}\left(y_1, y_2 \ldots\right) = \mathsf{P}\left(y_1\right) \prod_{i=2}^{N} \mathsf{P}\left(y_i \mid y_{i-1}\right) \tag{4.1}$$

As a result, y_i is conditionally independent of all of the variables $y_1 \ldots y_{i-2}$ given y_{i-1}.

The different values that the variables may take on are called the **states** of the process; in the models we discuss there is a finite number of such values and we take them to be the numbers $1 \ldots P$. The "state" terminology suggests a connection between a Markov process and a non-determinstic finite-state automaton. In fact, the sequence of states traversed by such an automaton in the absence of input (or given constant input) indeed forms a Markov sequence. We shall use the two sets of

terminology interchangeably, as is common in the field, referring, for instance, to the model as being in state p at step i when y_i takes the value p.

The joint distribution (4.1) is completely specified by the two discrete probability distributions, the **initial state probabilities** $\mathsf{P}(y_1)$ and the **state transition probabilities** $\mathsf{P}(y_i \mid y_{i-1})$ for $i > 1$. We can collect each of the transition probabilities into a $P \times P$ **transition matrix** T_+, so that $T_{+pq} = \mathsf{P}(y_i = p \mid y_{i-1} = q)$. The initial probabilities might be collected into a seperate vector T_0, however, in most cases it is more convenient to roll them into the transition matrix as follows. We introduce a new "random" variable y_0 which precedes (in the sense of the Markov conditioning criterion) the first actual random variable y_1. This variable assumes the value 0, which is not a possible outcome for any other variable, with probability one. In this model, the transition matrix is augmented to a $(P+1) \times (P+1)$ matrix T, with the first column containing the initial state probabilities; the first row being entirely zero to indicate that the system never makes a transition back into the state 0; and the remaining elements being the transition probabilities. For obvious reasons it will be convenient to number the rows and columns of T from 0, rather than 1. Once normalization requirements are accounted for, the augmented transition matrix T contains P^2-1 free parameters; $P-1$ specify the initial probabilities and $P(P-1)$ specify the transition probabilities.

Using this notation, manipulations of the probability functions becomes quite straightforward. For example, if the marginal distribution of the variable y_{i-1} is given by the vector π_{i-1}, then the marginal distribution of y_i is given by $\mathsf{P}(y_i = p) = \sum_q \mathsf{P}(y_i = p \mid y_{i-1} = q)\,\mathsf{P}(y_{i-1} = q)$, which can be written more succintly as $\pi_i = T\pi_{i-1}$. As a result, the marginal distribution of the ith variable is

$$\pi_i = T^i \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.2}$$

Given some basic regularity conditions on the transition matrix T, there exists a unique probability distribution over the states, represented by the vector π, which satisfies the condition

$$\mathsf{P}_T(y_i) = \pi \quad \Rightarrow \quad \mathsf{P}_T(y_{i+1}) = \pi \tag{4.3}$$

For obvious reasons, this is called the stationary distribution of the Markov process.

Clearly, π is a right eigenvector of the matrix T with eigenvalue 1. It can be shown, under some additional mild conditions on T (related to the ergodicity of the Markov process), that all other eigenvalues have absolute values strictly smaller than 1 (Seneta 1981; Karlin 1991). As a result, given any initial distribution on the states, after a sufficient number of steps the marginal

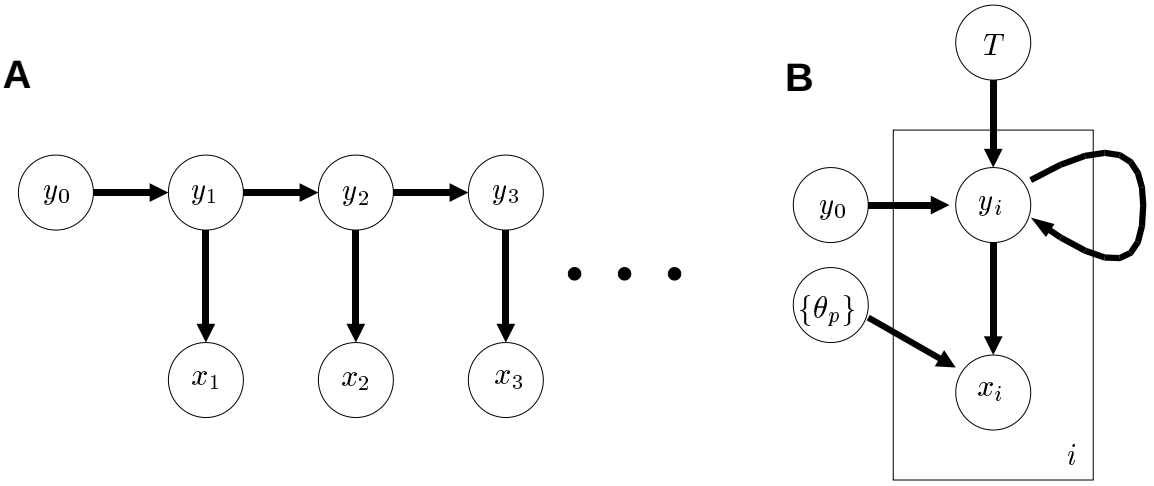

Figure 4.1: The hidden Markov model

distribution of the y_i will approach π. The stationary distribution is thus an attractor in the space of marginal distributions on the Markov variables. The magnitude of the largest non-unit eigenvalue sets the rate of decay of the non-stationary components, and thus the number of steps we need to wait in the typical case before the marginal state-distribution approaches the stationary one. This is called the **mixing time** of the (ergodic) chain.

4.1.2 The hidden Markov model

The **hidden Markov model** is a latent variable generative model derived from the basic Markov model described above. The structure of the model is drawn in graphical terms in figure 4.1. Panel A represents all of the variables of the model explicitly. The variables y_i form a Markov chain, but in this case they are not directly observed. Instead, we see output variables x_i which depend only on the corresponding state y_i; that is, each x_i is conditionally independent of all other variables, both observed and latent, given y_i. We adopt the convention of a deterministic initial state y_0 to compress all of the Markov parameters into a single matrix. There is no corresponding observable x_0.

The conditional distribution $\mathsf{P}(x_i \mid y_i)$ is stationary with respect to the instance variable i. Thus, associated with each state p (except 0) is an unchanging **output distribution** which plays a similar rôle to the component distributions of the mixture model. We will write θ_p for the parameters of this distribution and $\mathsf{P}_p(x)$ for the distribution (or density) function, just as in the case of the mixture model. Indeed, the connection between the two is quite deep. In figure 4.1B the same HMM, along with explicit parameter nodes, is shown in the more compact plate representation. It is clear that the structure is extremely similar to that of the mixture model; the only difference is the dependence of the latent variable between different instances. (As an aside, the plate notation is not well suited

for such models, since it does not make clear the essential Markov nature of the latent variable process, which is that the arrow linking the y_i nodes stretches only to the next plate.)

The parameters of model are the Markov probabilities contained in the matrix T along with all of the parameters θ_p of the output distributions. The likelihood of the parameters, with observations $\mathcal{X} = \{x_i\}$, is found by summing over all possible strings of Markov states $y_1 \ldots y_N$

$$\mathcal{L}_{\mathcal{X}}\left(T, \{\theta_p\}\right) = \sum_{y_1 \ldots y_N} \prod_i T_{y_i, y_{i-1}} \mathsf{P}_{y_i}\left(x_i\right) \tag{4.4}$$

An alternative, recursive, form for the calculation of this likelihood will appear below.

4.2 Learning: The Baum-Welch Algorithm

The commonly used learning algorithm for HMMs was developed in the course of classified work by Eric Baum and Lawrence Welch in the sixties. This algorithm turns out to be the standard EM algorithm applied to the generative model; however, its development pre-dated the publication of the original EM paper (Dempster *et al.* 1977) by at least a decade. The application is considerably more involved than the examples we have handled thus far. In particular, the E-step, in which parts of the conditional $\mathsf{P}_\theta\left(\mathcal{Y} \mid \mathcal{X}\right)$ are calculated, is sufficiently elaborate to have claimed a name of its own; it is called the forward–backward algorithm. Once this is completed, the M-step is more straightforward. The complete approach is commonly known as the Baum-Welch algorithm.

The joint data likelihood, based on observations, $\mathcal{X} = \{x_i\}$ and latent variable values $\mathcal{Y} = \{y_i\}$ is

$$\mathcal{L}_{\mathcal{X},\mathcal{Y}}\left(T, \{\theta_p\}\right) = \prod_{i=1}^{N} T_{y_i, y_{i-1}} \mathsf{P}_{y_i}\left(x_i\right) \tag{4.5}$$

leading to the log-likelihood

$$\ell_{\mathcal{X},\mathcal{Y}}\left(T, \{\theta_p\}\right) = \sum_i \log T_{y_i, y_{i-1}} + \sum_i \log \mathsf{P}_{y_i}\left(x_i\right) \tag{4.6}$$

As in the case of the mixture model, we introduce latent indicator variables in place of the discrete latent variables y_i. We define $z_{p,i}$ to take the value 1 if $y_i = p$ and 0 otherwise. We can then rewrite the log-likelihood as follows

$$\ell_{\mathcal{X},\mathcal{Z}}\left(T, \{\theta_p\}\right) = \sum_i \sum_{p,q} z_{p,i} z_{q,i-1} \log T_{pq} + \sum_i \sum_p z_{p,i} \log \mathsf{P}_p\left(x_i\right) \tag{4.7}$$

In the E-step for the nth iterarion, we take the expected value of this likelihood with respect to the conditional distribution determined by the parameter values on the $(n-1)$th step, $\mathsf{P}_{\theta^{n-1}}\left(\mathcal{Z} \mid \mathcal{X}\right)$.

This gives us

$$\begin{aligned} Q^n(T,\{\theta_p\}) &= \mathcal{E}_{\mathcal{Z}|\mathcal{X},\theta^{n-1}}\left[\ell_{\mathcal{X},\mathcal{Z}}\left(T,\{\theta_p\}\right)\right] \\ &= \sum_i \sum_{p,q} \mathcal{E}_{\mathcal{Z}|\mathcal{X},\theta^{n-1}}\left[z_{p,i}z_{q,i-1}\right]\log T_{pq} + \sum_i \sum_p \mathcal{E}_{\mathcal{Z}|\mathcal{X},\theta^{n-1}}\left[z_{p,i}\right]\log \mathsf{P}_p\left(x_i\right) \\ &= \sum_i \sum_{p,q} t^n_{pq,i}\log T_{pq} + \sum_i \sum_p s^n_{p,i}\log \mathsf{P}_p\left(x_i\right) \end{aligned} \tag{4.8}$$

where we have written $s^n_{p,i}$ for $\mathcal{E}_{\mathcal{Z}|\mathcal{X},\theta^{n-1}}\left[z_{p,i}\right]$ and $t^n_{pq,i}$ for $\mathcal{E}_{\mathcal{Z}|\mathcal{X},\theta^{n-1}}\left[z_{p,i}z_{q,i-1}\right]$. These quantities are analogous to the responsibilities of the mixture model, although that name is not used in this case. We shall call them the **state estimates** and **transition estimates** respectively. They are given by the probabilities

$$s^n_{p,i} = \mathsf{P}_{\theta^{n-1}}\left(z_{p,i}=1 \mid x_1 \ldots x_N\right) \tag{4.9}$$

$$t^n_{pq,i} = \mathsf{P}_{\theta^{n-1}}\left(z_{p,i}=1 \;\&\; z_{q,i-1}=1 \mid x_1 \ldots x_N\right) \tag{4.10}$$

Unlike in the case of the mixture model, the conditioning on the observations does not reduce to conditioning only on x_i, due to the coupling of latent variables in this model. These probabilities need to be calculated by an iterative approach known as the **forward–backward algorithm**.

4.2.1 E-step: The forward–backward algorithm

The algorithm by which the state and transition estimates are found is a special case of a general inference algorithm on probabilistic graphical models (Jordan 1998). However, we have not developed the general theory of such models here. Therefore, we simply lay out the algorithm, and then show that it does indeed achieve the necessary estimates.

We are given a hidden Markov model with known parameters, T and $\{\theta_p\}$, and a set of observations $\{x_i\}$. We wish to calculate the marginal probabilities of (4.9) and (4.10). Introduce two quantities, each a joint probability distribution, whose values can be calculated recursively at each timestep. The first is the likelihood that the system emitted the observed values $x_1 \ldots x_i$ and was then in state p at the ith time-step.

$$F_{p,i} = \mathsf{P}\left(y_i = p, x_1 \ldots x_i\right) \tag{4.11}$$

$$= \mathsf{P}_p\left(x_i\right)\sum_q T_{pq}F_{q,i-1} \tag{4.12}$$

Note that the likelihood that the model generated the complete string of observations is then just

$$\mathcal{L}_{\mathcal{X}}\left(T,\{\theta_p\}\right) = \sum_p F_{p,N} \tag{4.13}$$

thus obtaining the promised recursive expression for this likelihood. We will need this value again below, and so reserve for it the symbol L.

The second recursive quantity we need is the likelihood that, starting from state p on step i the system generated the observed string $x_{i+1} \ldots x_N$.

$$\begin{aligned} B_{p,i} &= \mathsf{P}\left(x_{i+1} \ldots x_N \mid y_i = p\right) && (4.14) \\ &= \sum_q T_{qp} \mathsf{P}_q\left(x_{i+1}\right) B_{q,i+1} && (4.15) \end{aligned}$$

Note that due to the Markov nature of the latent variable chain, observations x_{i+1} and further are independent of all previous observations given the value of y_i and so $B_{p,i}$ is also equal to $\mathsf{P}\left(x_{i+1} \ldots x_N \mid y_i = p, x_1 \ldots x_i\right)$

Both recursions can be written more succinctly if we introduce a $(P+1) \times (P+1)$ diagonal matrix R_i (indexed, like T, from 0) with $R_{pp,i} = \mathsf{P}_p\left(x_i\right)$. We then obtain, with vector forms for both F and B

$$F_i = R_i T F_{i-1} \qquad \text{and} \qquad B_i = T^{\mathsf{T}} R_{i+1} B_{i+1} \tag{4.16}$$

Notice that one of these recursions runs forward over the observations, while the other runs backwards. Thus the name "forward–backward".

The estimates $s_{p,i}$ and $t_{pq,i}$ can be expressed in terms of F and B:

$$\begin{aligned} s_{p,i} &= \mathsf{P}\left(y_i = p \mid x_1 \ldots x_N\right) \\ &= \frac{\mathsf{P}\left(x_{i+1} \ldots x_N \mid y_i = p\right) \mathsf{P}\left(y_i = p, x_1 \ldots x_i\right)}{\mathsf{P}\left(x_1 \ldots x_N\right)} \\ &= F_{p,i} B_{p,i} / L && (4.17) \end{aligned}$$

and

$$\begin{aligned} t_{pq,i} &= \mathsf{P}\left(y_i = p, y_{i-1} = q \mid x_1 \ldots x_N\right) \\ &= \frac{\mathsf{P}\left(x_{i+1} \ldots x_N \mid y_i = p\right) \mathsf{P}\left(x_i \mid y_i = p\right) \mathsf{P}\left(y_i = p \mid y_{i-1} = q\right) \mathsf{P}\left(y_{i-1} = q, x_1 \ldots x_{i-1}\right)}{\mathsf{P}\left(x_1 \ldots x_N\right)} \\ &= B_{p,i} R_{pp,i} T_{pq} F_{q,i-1} / L && (4.18) \end{aligned}$$

where, in the second step of each of these results we have used the Markovian properties of the model to remove irrelevant conditioning variables.

The E-step of the Baum-Welch algorithm, then, is achieved by substituting into (4.17) and (4.18) the $(n-1)$th iteration parameter estimates, to obtain $s^n_{p,i}$ and $t^n_{pq,q}$.

4.2.2 M-step: Parameter re-estimation

The re-estimation of the Markov transition matrix is straightforward, and reminiscient of the re-estimation of the mixing probabilities of a mixture model. We optimize the expected log-likelihood of (4.8) with respect to T_{pq}, enforcing the constraint $\sum_p T_{pq} = 1$ with a Lagrange multiplier, to obtain

$$\frac{\partial}{\partial T_{pq}}\bigg|_{T^n_{pq}} \left(\sum_i \sum_{p,q} t^n_{pq,i} \log T_{pq} - \lambda \sum_p T_{pq} \right) = \sum_i \frac{t^n_{pq,i}}{T^n_{pq}} - \lambda = 0 \tag{4.19}$$

From which we find that $T_{pq} \propto \sum_i t^n_{pq,i}$. The normalization constraint then gives us

$$T^n_{pq} = \frac{\sum_{i=1}^{N} t^n_{pq,i}}{\sum_{i=0}^{N-1} s^n_{q,i}} \tag{4.20}$$

where we use the fact that $\sum_p t^n_{pq,i} = s^n_{q,i-1}$ which follows from the marginalization of the joint distribution represented by $t_{pq,i}$

The remaining update rules, for the output distribution parameters $\{\theta_p\}$, depend on the form of the output distribution function. We can, however, make some headway. First, note that the θ_p are independent of each other, and so can each be optimized separately. Furthermore, only the second term in the expected log-likelihood (4.8) has any dependence on θ_p. As a result, we arrive at an update rule identical to that encountered in the case of the mixture model (2.15), with the responsibilities replaced by the state estimates $s^n_{p,i}$.

$$\theta^n_p = \operatorname*{argmax}_{\theta_p} \sum_i s^n_{p,i} \log \mathsf{P}_{\theta_p}(x_i) \tag{4.21}$$

As in the mixture case, we may interpret this as a weighted fit of the output distribution parameters to the observations x_i, with weights given by the estimates $s^n_{p,i}$.

4.3 Sparse HMMs

In this section, we introduce a special case of the HMM. This restricted model, the **sparse hidden Markov model** or SHMM, is one that may be encountered with some frequency in practical modeling situations; indeed we develop it here because it will be of use to us in a neural data analysis problem tackled in the following chapters. The restricted model itself will only be of limited interest from an algorithmic point of view: all of the standard HMM learning algorithms may be used and, though we will describe an adaptation of the standard Baum-Welch algorithm, the advantages thereby derived are merely in the realm of efficiency. However, the introduction of this model will allow us to speak meaningfully of a mixture of sparse HMMs, and derive an efficient learning algorithm for such a mixture.

The processes that we consider are sparse in the following sense. In each string of observations x_i, the majority yield a null value, which we represent by the symbol ø. This value tells us relatively little about the state of the underlying process; in effect, the process has no output at these observation times. Scattered within this string of øs are occasional non-null output values, but these are distributed sparsely. Nevertheless, they provide our only information about the state of the process.

We will examine hidden Markov models for such a process. Each model contains one or more states for which the output distribution produces the outcome ø with probability 1. We will refer to these as the null states. We will assume for the purposes of this discussion that the output distributions in the remaining states assign probability 0 to this outcome, although most of the results of this and the following sections can be carried through even if this were not the case. The sparsity of the process requires that the transition matrix be set up so that on the majority of time-steps the model is in a null state. On the whole, then, the transition probabilities from null states to states with full output distributions are relatively low, while transitions in the other direction are relatively likely.

How sparse is sparse? There is no precise answer to this question. All of the algorithms that we discuss can be equally well applied to models which spend little or no time in null states. However, it will be apparent that under that condition they would produce poor results. The transition between sparse and full, then, is a matter for empirical discovery within the framework of the application.

Learning in the SHMM may proceed by the standard Baum-Welch algorithm that was laid out in the case of the full HMM. However, it is possible to achieve some optimizations on the basis of the sparse output structure, which we will discuss here. Before we can do so, however, we need to recast the forward–backward algorithm slightly.

4.3.1 Another view of the forward–backward algorithm

The presentation in section 4.2.1 described the forward–backward algorithm in a notationally compact form ideal for exposition. In fact, as described, the algorithm is numerically unstable in implementations. This instability can be resolved by a small modification, which is the subject of this section. The same modification is important to adaptations of the algorithm to sparse HMMs.

The difficulty with the currently described algorithm is this. At each instance i, the conjunction of observations that appear in the likelihoods described by F_i and B_i is of a different size. For instance, F_1 describes the likelihood $\mathsf{P}(y_1, x_1)$, while F_N describes $\mathsf{P}(y_N, x_1 \ldots x_N)$. If the typical density at the observation point x_i is a, then while F_1 is of order a, F_N is of order a^N. Similarly, B_1 is of order a^{N-1}, while B_N is of order a^0. The product of the two terms is always of order a^N, and it is divided by the likelihood (also order a^N) to derive estimates $s_{p,i}$ and $t_{pq,i}$ of order 1. If the value a is considerably different from 1, the intermediate values in this calculation can become either very large or very small, and the computation may become numerically unstable.

We can resolve this problem by introducing an alternative group of recursive functions that remain of order 1 throughout. In fact, we need three functions

$$C_i = \mathsf{P}(x_i \mid x_1 \ldots x_{i-1}) \tag{4.22}$$

$$F_{p,i} = \mathsf{P}(y_i = p \mid x_1 \ldots x_i) \tag{4.23}$$

$$B_{p,i} = \frac{\mathsf{P}(x_{i+1} \ldots x_N \mid y_i = p)}{\mathsf{P}(x_{i+1} \ldots x_N \mid x_1 \ldots x_i)} \tag{4.24}$$

which are calculated recursively as follows.

$$C_i = \mathbf{1}^{\mathsf{T}} R_i T F_{i-1} \tag{4.25}$$

$$F_i = R_i T F_{i-1} / C_i \tag{4.26}$$

$$B_i = T^{\mathsf{T}} R_{i+1} B_{i+1} / C_{i+1} \tag{4.27}$$

where $\mathbf{1}$ is a vector of P ones, and is introduced to indicate a sum of the elements of the following vector-valued product.

Given these new functions, the state and transition estimates become

$$s_{p,i} = F_{p,i} B_{p,i} \qquad \text{and} \qquad t_{pq,i} = B_{p,i} R_{pp,i} T_{pq} F_{q,i-1} / C_i. \tag{4.28}$$

The normalization of the recursive terms F and B defined here is crucial to the following exposition of the forward–backward algorithm for SHMMs. Thus, all subsequent references to the algorithm, and the symbols F, B and C will refer to this recast version.

4.3.2 Forward–backward algorithm for sparse HMMs

By definition, the output sequences recorded from a sparse HMM tend to contain long stretches of null outputs. These segments leave the model in an identifiable configuration; that is, the value of F_i at the end, and B_i at the beginning of such a sequence is relatively independent of the measurements before and after such a segment.

Consider a long segment of null observations stretching from observation indices a to $a + l$. We assume that the values of the functions F_{a-1} and B_{a+l} are known, while we seek to calculate F_{a+l} and B_{a-1}.

Consider, first, the forward term. Let the notation $R_\emptyset$ stand for the value of the likelihood matrix R_i in cases where $x_i = \emptyset$. Recall that such matrices are diagonal, with $R_{pp,i} = \mathsf{P}_p(x_i)$. In this case, these elements are 1 for null states and 0 elsewhere. We then have

$$F_{a+l} \propto (R_\emptyset T)^{(l+1)} F_{a-1} \tag{4.29}$$

with the vector then normalized so that the sum of its elements is 1. Whatever the value of F_{a-1}, this expression will be dominated by the leading eigenvector of the matrix $R_\emptyset T$. We will write $F_\emptyset$ for the suitably normalized eigenvector — note that normalization here means that the sum of the elements, rather than the sum of the squares of the elements, is 1. In fact, $F_\emptyset$ is the stationary distribution of the Markov chain that is obtained by restricting the current estimate of the Markov model to only the null states, the transition matrix of which is given by renormalizing the columns of the matrix $R_\emptyset T R_\emptyset$. Thus the forward step after a sequence of null outputs is achieved by simply setting the value of the forward term to $F_\emptyset$.

Using a similar argument we can show that at the *beginning* of a long segment of nulls, the value of the backward term B_{a-1} will approach the leading eigenvector of the matrix $T^\mathsf{T} R_\emptyset$, suitably normalized. We write $\tilde{B}_\emptyset$ for the unnormalized eigenvector. Unlike the forward terms, B_i is not itself a probability distribution and thus we have no immediate way to normalize. However the products $F_i B_i = \mathsf{P}(y_i \mid x_1 \ldots x_N)$ are probabilities. Thus, knowing the value of F_{a-1} we can find the appropriate normalization for B_{a-1} (which is potentially different before each null segment).

The forward–backward steps across a sequence of nulls from a to $a+l$ is thus

$$F_{a+l} = F_\emptyset \tag{4.30}$$

$$B_{a-1} = \tilde{B}_\emptyset / F_{a-1}^\mathsf{T} \tilde{B}_\emptyset \tag{4.31}$$

The use of these forms limits the application of the full forward–backward algorithm to only those regions in which some non-null outputs are observed, often at a considerable computational savings.

4.4 Mixtures of Sparse HMMs

We consider the following model. We have M independent sparse hidden Markov models. Call the output of the mth model at time-step i, $x_{m,i}$[1]. We do not observe these variables directly, instead we make a single observation at each time-step, derived from these values according to the following

$$x_i = \begin{cases} \emptyset & \text{if all } x_{m,i} = \emptyset \\ x_{m^*,i} & \text{if only } x_{m^*,i} \neq \emptyset \\ \not\emptyset & \text{if multiple } x_{m,i} \neq \emptyset \end{cases} \tag{4.32}$$

[1] Variables in the ensuing development will often need to be identified by state, component model and observation number. We shall adopt two conventions to assist in correctly parsing all of these subscripts. 1. The order will always be (state, model, instance), but some indices might be omitted if unnecessary. 2. the letters p and q will be used to index state, m and l for model, and i for instance; n will be used in the superscript for EM iteration number as before.

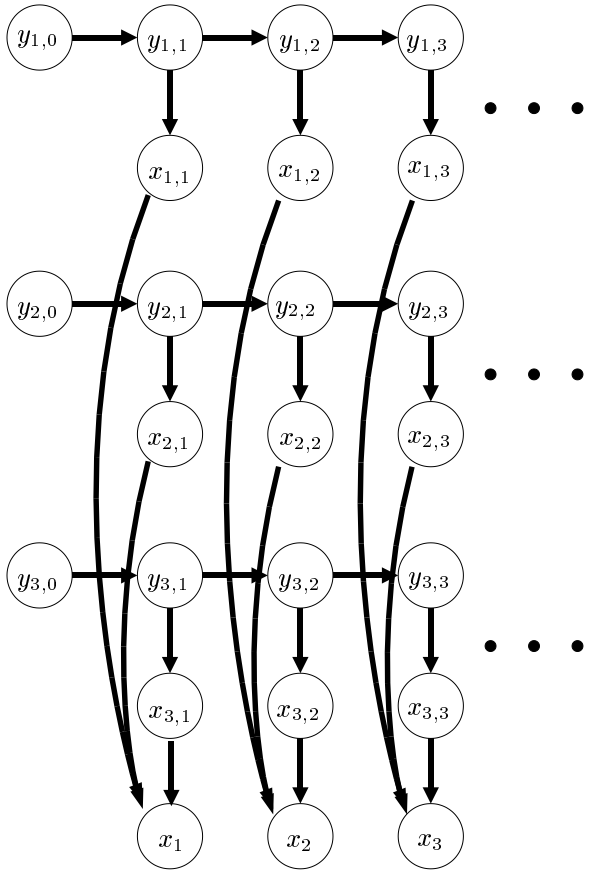

Figure 4.2: A mixture of sparse hidden Markov models

If more than one HMM has non-null output, we see only the fact that a collision occured, noted by the special output value $\not\in$. We obtain no information about which, nor even how many, of the HMMs had non-null outputs.

The model is illustrated in figure 4.2. The random variables in the model are the state variables $y_{m,i}$ and the corresponding outputs $x_{m,i}$. The observed value x_i is actually a deterministic function of the outputs, $x_{m,i}$, of each component sparse HMM.

4.4.1 Learning

Since the component SHMMs are presumed to be independent, the joint data likelihood, given observations $\mathcal{X} = \{x_i\}$, HMM outputs $\mathcal{X}_m = \{x_{m,i}\}$ and indicator variables $\mathcal{Z} = \{z_{m,i}\}$ is simply the product of the joint data likelihoods (4.5) for each of the component HMMs given observations $\{x_{m,i}\}$ and indicators $\{z_{m,i}\}$. In the log domain, this is

$$\ell_{\mathcal{X},\mathcal{X}_m,\mathcal{Z}}\left(\{T_m\},\{\theta_{p,m}\}\right) = \sum_m \sum_i \left(\sum_{p,q} z_{p,m,i} z_{q,m,i} \log T_{pq,m} + \sum_p z_{p,i} \log \mathsf{P}_{p,m}\left(x_{m,i}\right) \right) \qquad (4.33)$$

The E-step involves calculation of the expected value of this expression with respect to the distribution $\mathsf{P}\left(z_{p,m,i}, x_{m,i} \mid x_i\right)$. Note that the expectation is taken not only with respect to the $z_{m,i}$ (as usual), but also with respect to the $x_{m,i}$, which are not directly observed in this case. The expected value is

$$\begin{aligned} Q^n(\{T_m\},\{\theta_{p,m}\}) &= \mathcal{E}_{\mathcal{Z},\mathcal{X}_m|\mathcal{X},\theta^{n-1}}\left[\ell_{\mathcal{X},\mathcal{X}_m,\mathcal{Z}}\left(\{T_m\},\{\theta_{p,m}\}\right)\right] \\ &= \sum_m \sum_i \sum_{p,q} \mathcal{E}_{\mathcal{Z},\mathcal{X}_m|\mathcal{X},\theta^{n-1}}\left[z_{p,m,i} z_{q,m,i-1}\right] \log T_{pq,m} \\ &\qquad + \sum_m \sum_i \sum_p \mathcal{E}_{\mathcal{Z},\mathcal{X}_m|\mathcal{X},\theta^{n-1}}\left[z_{p,m,i} \log \mathsf{P}_{p,m}\left(x_{m,i}\right)\right] \\ &= \sum_m \sum_i \sum_{p,q} t^n_{pq,m,i} \log T_{pq,m} \\ &\qquad + \sum_m \sum_i \sum_p s^n_{p,m,i} \mathcal{E}_{x_{m,i}|z_{p,m,i}=1,\mathcal{X},\theta^{n-1}}\left[\log \mathsf{P}_{p,m}\left(x_{m,i}\right)\right] \end{aligned} \tag{4.34}$$

Note the change in distribution that appears in the expectation of the final expression; we have used the fact that $z_{p,m,i}$ is an indicator variable as follows

$$\begin{aligned} &\mathcal{E}_{\mathcal{Z},\mathcal{X}_m|\mathcal{X},\theta^{n-1}}\left[z_{p,m,i} \log \mathsf{P}_{p,m}\left(x_{m,i}\right)\right] \\ &= \sum_{z_{p,m,i}} \int dx_{m,i}\, \mathsf{P}_{\theta^{n-1}}\left(z_{p,m,i}, x_{m,i} \mid \mathcal{X}\right)\; z_{p,m,i} \log \mathsf{P}_{p,m}\left(x_{m,i}\right) \\ &= \mathsf{P}_{\theta^{n-1}}\left(z_{p,m,i}=1 \mid \mathcal{X}\right) \int dx_{m,i}\, \mathsf{P}_{\theta^{n-1}}\left(x_{m,i} \mid z_{p,m,i}=1, \mathcal{X}\right) \log \mathsf{P}_{p,m}\left(x_{m,i}\right) \\ &\qquad + \mathsf{P}_{\theta^{n-1}}\left(z_{p,m,i}=0 \mid \mathcal{X}\right)\; 0 \\ &= s^n_{p,m,i} \mathcal{E}_{x_{m,i}|z_{p,m,i}=1,\mathcal{X},\theta^{n-1}}\left[\log \mathsf{P}_{p,m}\left(x_{m,i}\right)\right] \end{aligned} \tag{4.35}$$

What is this expected value? If no collision was observed then $x_{m,i}$ is completely determined by $z_{p,m,i}$ and x_i. If state p of model m is a null state, $x_{m,i} = \emptyset$; otherwise $x_{m,i} = x_i$. On the other hand, if a collision was observed then x_i tells us nothing about the value of $x_{m,i}$. It is still true that if the state (p,m) has no output, $x_{m,i} = \emptyset$; but now, if the state is non-null, $x_{i,m}$ is distributed according to $\mathsf{P}_{p,m}\left(x\right)$. Thus, for non-null states, we have

$$\mathcal{E}_{x_{m,i}|z_{p,m,i}=1,\mathcal{X},\theta^{n-1}}\left[\log \mathsf{P}_{p,m}\left(x_{m,i}\right)\right] = \begin{cases} \log \mathsf{P}_{p,m}\left(x_i\right) & \text{if } x_i \neq \emptyset \\ -\mathsf{H}[\mathsf{P}_{p,m}] & \text{if } x_i = \emptyset \end{cases} \tag{4.36}$$

where $\mathsf{H}[\cdot]$ indicates the entropy of the distribution.

4.4.2 Coupled forward–backward algorithm

We need to calculate the state and transition estimates that appear in (4.34). We do so by running the forward–backward algorithm separately on each component SHMM. Since direct observation of

the outputs of the component models is not possible, however, we must estimate those outputs using the observed output of the entire mixture, as well as the recursive terms, $F_{m,i-1}$ and $B_{m,i-1}$, from all of the components. This use of the values of the recursive terms from other component SHMMs leads to a coupling of the different instances of the forward–backward algorithm.

Despite this coupling, however, the separation of the estimation process into multiple component recursions constrains the E-step optimization to only those distributions which satisfy a factorization constraint of the form (for the F recursion)

$$.\mathsf{P}\left(\{y_{m,i}\} \mid x_1 \ldots x_i\right) = \prod_m \mathsf{P}\left(y_{m,i} \mid x_1 \ldots x_i\right) \tag{4.37}$$

as well as a second, similar, constraint due to the B recursion. Such imperfect E-steps were discussed briefly in section 1.8. At each time-step we calculate the full joint distribution of the $y_{m,i}$ (which contains P^M terms) but then store only the marginals (needing only $P \times M$ terms). Clearly, to calculate the state and transtion estimates we only need the marginals, and so from that point of view the restriction is reasonable. However, the $F_{m,i}$ are also used to estimate the distribution at the $(i+1)$th step. Use of the factorized distribution for the ith step, rather than the full joint distribution, leads to a mis-estimation of the joint distribution at the $(i+1)$th step. It is thus, that the constraint of (4.37) appears.

We will discuss the impact of this constraint on the EM process below. First, let us proceed with the exposition of the algorithm. The recursive terms are defined much as before.

$$C_i = \mathsf{P}\left(x_i \mid x_1 \ldots x_{i-1}\right) \tag{4.38}$$

$$F_{p,m,i} = \mathsf{P}\left(y_{m,i} = p \mid x_1 \ldots x_i\right) \tag{4.39}$$

$$B_{p,m,i} = \frac{\mathsf{P}\left(x_{i+1} \ldots x_N \mid y_{m,i} = p\right)}{\mathsf{P}\left(x_{i+1} \ldots x_N \mid x_1 \ldots x_i\right)} \tag{4.40}$$

However, in this case the x_i are not the direct outputs of the HMM, but are rather the overall observations from the mixture. Thus, the calculations become slightly more elaborate. We will obtain here expressions for only the forward terms C_i and $F_{p,m,i}$. The calculation of $B_{p,m,i}$ proceeds similarly.

We write $\tilde{F}_{p,m,i}$ for $\mathsf{P}\left(y_{m,i} = p \mid x_1 \ldots x_{i-1}\right)$, the probability of finding the mth model in state p on step i given the previous observations, but not the current one. This is, of course, based recursively on our estimate of the distribution of states $y_{m,i-1}$ given observations up to x_{i-1}. With our factorial assumption on the distribution of $y_{m,i-1}$ this is given by

$$\tilde{F}_{m,i} = T_m F_{m,i-1} \tag{4.41}$$

Also of interest will be the probability that model m is in a null state. We will write $\emptyset_{p,m} = 1$ if $\mathsf{P}_{p,m}(\emptyset) = 1$ and $\emptyset_{p,m} = 0$ otherwise. Using this indicator, we obtain $\tilde{F}_{\emptyset,m,i} = \sum_p \emptyset_{p,m} \tilde{F}_{p,m,i}$.

It will be useful to treat separately the three cases where x_i is 1. null, 2. non-null and non-collision, and 3. a collision.

1. $x_i = \emptyset$

   In this case C_i is the probability that every component is in a null state,

$$C_i = \prod_m \tilde{F}_{\emptyset,m,i} \tag{4.42}$$

   To calculate $F_{p,m,i}$ we need to find the distribution $\mathsf{P}(x_i = \emptyset, y_{m,i} = p \mid x_1 \ldots x_{i-1}) = \mathsf{P}(x_i = \emptyset \mid y_{m,i} = p, x_1 \ldots x_{i-1}) \tilde{F}_{p,m,i}$. This is clearly 0 if $\emptyset_{p,m} = 0$. If $\emptyset_{p,m} = 1$, then $\mathsf{P}(x_i = \emptyset \mid y_{m,i} = p, x_1 \ldots x_{i-1})$ is just the probability that all other components are in null states. Thus

$$\begin{aligned} F_{p,m,i} &= \frac{1}{C_i} \mathsf{P}(x_i = \emptyset, y_{m,i} = p \mid x_1 \ldots x_{i-1}) \\ &= \frac{1}{C_i} \emptyset_{p,m} \tilde{F}_{p,m,i} \prod_{l \neq m} \tilde{F}_{\emptyset,l,i} \\ &= \emptyset_{p,m} \frac{\tilde{F}_{p,m,i}}{\tilde{F}_{\emptyset,m,i}} \end{aligned} \tag{4.43}$$

2. $x_i \neq \emptyset, \not{\emptyset}$

   Here, C_i is the probability that one component outputs the observed value x_i, while all the other components are in null states.

$$C_i = \sum_m \sum_p \mathsf{P}_{p,m}(x_i) \tilde{F}_{p,m,i} \prod_{l \neq m} \tilde{F}_{\emptyset,l,i} \tag{4.44}$$

   $\mathsf{P}(x_i \mid y_{m,i} = p, x_1 \ldots x_{i-1})$ is straightforward if (p, m) is not null; being $\mathsf{P}_{p,m}(x_i)$ times the the probability that all other components are in null states. If, on the other hand, $\emptyset_{p,m} = 1$, then the conditional probability is given by the probability that exactly one of the remaining components outputs the value x_i.

$$F_{p,m,i} = \frac{1}{C_i} \tilde{F}_{p,m,i} \left((1 - \emptyset_{p,m}) \mathsf{P}_{p,m}(x_i) \prod_l \tilde{F}_{\emptyset,l,i} + \emptyset_{p,m} \sum_{l \neq m} \sum_p \mathsf{P}_{p,l}(x_i) \tilde{F}_{p,l,i} \prod_{k \neq l,m} \tilde{F}_{\emptyset,k,i} \right) \tag{4.45}$$

3. $x_i = \not{\emptyset}$

In this case, C_i is the probability that at least two components are in a non-null state

$$C_i = 1 - \left(\prod_m F_{\emptyset,m,i} \right) - \left(\sum_m (1 - F_{\emptyset,m,i}) \prod_{l \neq m} F_{\emptyset,m,i} \right) \tag{4.46}$$

The expression for $F_{p,m,i}$ is notationally cumbersome, so we will not write it explicitly. Instead, we note that $\mathsf{P}\left(x_i \mid y_{m,i} = p, x_1 \ldots x_{i-1}\right)$ is the probability that at least one other component is non-null if $\emptyset_{p,m} = 0$ and that at least two other components are non-null if $\emptyset_{p,m} = 1$. Both of these probabilities are found in a form similar to that of C_i, above.

Once the terms $F_{i,m}$ and $B_{i,m}$ have been calculated, the state and transition estimates are derived using (4.28) applied to each component in turn.

Consequences of the factorial approximation

To what extent does the factorial constraint of the coupled forward–backward algorithm affect the eventual parameter estimates? We may can make two separate arguments for robustness of the estimates to error.

First, it might be feared that, since the terms F and B are calculated recursively and since there is an error in each calculation, the estimated value and the true value would progressively diverge over time. This is not the case. Boyen and Koller (1999) have examined factorial approximations such as the present one in the context of general dynamic probabilistic networks. They argue that the approximation error does not grow over time because two forces oppose the growth. First, the incorporation of observed data tends to drive the approximated distribution towards the correct one. Second, the randomization due to the stochastic transition from the $(i-1)$th step to the ith tends to broaden both the correct distribution and the approximate one, which also has the effect of bringing them closer together. In other words, $T_m F_{m,i-1}$ may be closer to the true $\mathsf{P}\left(y_{m,i} \mid x_i \ldots x_{i-1}\right)$ than $F_{m,i-1}$ is to $\mathsf{P}\left(y_{m,i-1} \mid x_i \ldots x_{i-1}\right)$. Intuitively, we may think of each random transition contributing to a "forgetting" of the old, incorrect, distribution.

To these arguments we can add a third, peculiar to the current model. When the observation $x_i = \emptyset$, our forward and backward steps are correct. Recall from the discussion of the forward–backward algorithm for sparse HMMs that after a substantial stetch of null observations, F_i (B_i) is relatively independent of its value at the beginning (end) of the segment. Thus, in the mixture, whenever we enounter a stretch of null observations we tend to reset the forward–backward estimates to their correct values.

Second, even if the errors in the state and transition estimates are typically large, it is possible that their effect on parameter estimates derived through EM may be small. Constrained E-steps of the sort we perform here were discussed briefly in section 1.8. There it was pointed out that

generalized EM using a constrained optimization of the latent variable distribution will eventually yield the correct maximum-likelihood parameter estimates if and only if the conditional distribution at the optimum $\mathsf{P}_{\theta^*}(\mathcal{Y} \mid \mathcal{X})$ satisfies the constraint. In the present case, this will be true if, at the optimal parameter values, only one component is likely to be in a non-null state at each time-step where $x_i \neq \emptyset, \not{c}$. In other words, all observed data can be assigned with high likelihood to only one component. If, on the other hand, two different components claim equal responsibility for the point, then the factored distribution will assign a probability close to 0.25 that they were both in non-null states, whereas the correct joint probability would be 0 (if they were both in non-null states a collision would have been observed). Futhermore, provided that most data are well assigned in this way, the above arguments suggest that a small number of ambiguous points will not have a profound effect on the estimates associated with the others. Thus, in well clustered data, the approximation has little effect on the eventual estimates, even if, in intermediate steps of EM, it is inaccurate. Note that "well-clustered" here does not necessarily mean that the output distributions are well separated. Each data point must be assigned to a single component, either because only that component has an output distribution which assigns it high likelihood or because its temporal relationship to nearby points marks it as arising from a particular model.

4.4.3 Parameter re-estimation

The M-step requires optimization of the expected log-likelihood (4.34) with respect to the parameters, with the estimates $s^n_{p,m,i}$ and $t^n_{pq,m,i}$ fixed at the values derived from the E-step. The expression of (4.34) contains separate additive terms for each component model; as a result, it can be ompti-mized with respect to the parameters of each SHMM independently. The part that involves the mth model is

$$Q^n_m(T_m, \{\theta_{p,m}\}) = \sum_i \sum_{p,q} t^n_{pq,m,i} \log T_{pq,m} + \sum_i \sum_p s^n_{p,m,i} \mathcal{E}_{x_{m,i} | z_{p,m,i}=1, \mathcal{X}, \theta^{n-1}} \left[\log \mathsf{P}_{p,m}(x_{m,i})\right] \tag{4.47}$$

Optimization with respect to $T_{pq,m}$ can clearly proceed exactly as in the standard case, and so we obtain

$$T^n_{pq,m} = \frac{\sum_{i=1}^{N} t^n_{pq,m,i}}{\sum_{i=0}^{N-1} s^n_{q,m,i}} \tag{4.48}$$

Re-estimation of the output distribution parameters $\theta_{p,m}$ is almost the same as in the standard Baum–Welch algorithm. It is still the case that the different output distributions can be optimized independently. For states with null output distributions, of course, there are no parameters to fit.

For non-null distributions, we recall the result of (4.36) and find that

$$\theta^n_{p,m} = \underset{\theta_{p,m}}{\operatorname{argmax}} \left(\sum_{i:x_i \neq \emptyset, q} s^n_{p,m,i} \log \mathsf{P}_{\theta_{p,m}}(x_i) - \sum_{i:x_i = q} s^n_{p,m,i} \mathsf{H}[\mathsf{P}_{p,m}] \right) \tag{4.49}$$

(Note that if $x_i = \emptyset$ and (p, m) is not a null state, $s^n_{p,m,i}$ must be 0, and so we can ignore the corresponding terms). Thus, the parameters are fit to the observed non-null and non-collision data, weighted by the state estimates as usual, but with an additional entropy penalty on the likelihood which weighted by the sum of the state estimates for collision time-steps. In practice, if the number of collisions is small relative to the total number of non-null observations, we can often neglect this term.