CHAPTER

9 Context-Dependent Classification

9.1 INTRODUCTION

The classification tasks considered so far have assumed that no relation exists among the various classes. In other words, having obtained a feature vector $\boldsymbol{x}$ from a class ω_i, the next feature vector could belong to any other class. In this chapter we will remove this assumption, and we will assume that the various classes are closely related. That is, successive feature vectors are not independent. Under such an assumption, classifying each feature vector separately from the others obviously has no meaning. The class to which a feature vector is assigned depends (a) on its own value, (b) on the values of the other feature vectors, and (c) on the existing relation among the various classes. Such problems appear in various applications such as communications, speech recognition, and image processing.

In the context-free classification, our starting point was the Bayesian classifier. In other words, a feature vector was classified to a class ω_i if

$$P(\omega_i|\boldsymbol{x}) > P(\omega_j|\boldsymbol{x}), \quad \forall j \neq i$$

The Bayesian point of view will also be adopted here. However, the dependence among the various classes sets demands for a more general formulation of the problem. The mutual information that resides within the feature vectors requires the classification to be performed using *all* feature vectors simultaneously and also to be arranged in the same sequence in which they occurred from the experiments. For this reason, in this chapter we will refer to the feature vectors as *observations* occurring in sequence, one after the other, with $\boldsymbol{x}_1$ being the first and $\boldsymbol{x}_N$ the last from a set of N observations.

9.2 THE BAYES CLASSIFIER

Let $X : \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ be a sequence of N (feature vectors) observations and $\omega_i, i = 1, 2, \ldots, M$, the classes in which these vectors must be classified. Let

$\Omega_i : \omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_N}$ be one of the possible sequences of these classes corresponding to the observation sequence, with $i_k \in \{1, 2, \ldots, M\}$ for $k = 1, 2, \ldots, N$. The total number of these class sequences Ω_i is M^N, that is, the number of possible *ordered* combinations of M distinct objects taken in groups of N. *Our classification task is to decide to which class sequence Ω_i a sequence of observations X corresponds*. This is equivalent to appending $\boldsymbol{x}_1$ to class ω_{i_1}, $\boldsymbol{x}_2$ to ω_{i_2}, and so on. A way to approach the problem is to view each specific sequence X as an (extended) feature vector and $\Omega_i, i = 1, 2, \ldots, M^N$, as the available classes. Having observed a specific X, the Bayes rule assigns it to Ω_i for which

$$P(\Omega_i|X) > P(\Omega_j|X), \quad \forall i \neq j \tag{9.1}$$

and following our already familiar arguments, this is equivalent to

$$P(\Omega_i)p(X|\Omega_i) > P(\Omega_j)p(X|\Omega_j), \quad \forall i \neq j \tag{9.2}$$

In the following we will investigate the specific form that Eq. (9.2) takes for some typical class dependence models.

9.3 MARKOV CHAIN MODELS

One of the most widely used models describing the underlying class dependence is the Markov chain rule. If $\omega_{i_1}, \omega_{i_2}, \ldots$ is a sequence of classes, then the Markov model assumes that

$$P(\omega_{i_k}|\omega_{i_{k-1}}, \omega_{i_{k-2}}, \ldots, \omega_{i_1}) = P(\omega_{i_k}|\omega_{i_{k-1}}) \tag{9.3}$$

The meaning of this is that the class dependence is limited only within two successive classes. This type of model is also called a first-order Markov model, to distinguish it from obvious generalizations (second, third, etc.). In other words, given that the observations $\boldsymbol{x}_{k-1}, \boldsymbol{x}_{k-2}, \ldots, \boldsymbol{x}_1$ belong to classes $\omega_{i_{k-1}}, \omega_{i_{k-2}}, \ldots, \omega_{i_1}$, respectively, the probability of the observation $\boldsymbol{x}_k$, at stage k, belonging to class ω_{i_k} *depends only on the class from which observation $\boldsymbol{x}_{k-1}$, at stage $k-1$, has occurred*. Now combining (9.3) with the probability chain rule [Papo 91, p. 192],

$$\begin{aligned} P(\Omega_i) &\equiv P(\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_N}) \\ &= P(\omega_{i_N}|\omega_{i_{N-1}}, \ldots, \omega_{i_1})P(\omega_{i_{N-1}}|\omega_{i_{N-2}}, \ldots, \omega_{i_1}) \ldots P(\omega_{i_1}) \end{aligned}$$

we obtain

$$P(\Omega_i) = P(\omega_{i_1}) \prod_{k=2}^{N} P(\omega_{i_k}|\omega_{i_{k-1}}) \tag{9.4}$$

where $P(\omega_{i_1})$ is the prior probability for class $\omega_{i_1}, i_1 \in \{1, 2, \ldots, M\}$, to occur. Furthermore, two commonly adopted assumptions are that (a) given the sequence of classes, the observations are statistically independent, and (b) the probability density function in one class does not depend on the other classes. *That is,*

dependence exists only on the sequence in which classes occur, but within a class observations "obey" the class' own rules. This assumption implies that

$$p(X|\Omega_i) = \prod_{k=1}^{N} p(\boldsymbol{x}_k|\omega_{i_k}) \tag{9.5}$$

Combining Eqs. (9.4) and (9.5), the Bayes rule for Markovian models becomes equivalent to the statement:

Statement

Having observed the sequence of feature vectors $X : \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, classify them in the respective sequence of classes $\Omega_i : \omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_N}$, so that the quantity

$$p(X|\Omega_i)P(\Omega_i) = P(\omega_{i_1})p(\boldsymbol{x}_1|\omega_{i_1}) \prod_{k=2}^{N} P(\omega_{i_k}|\omega_{i_{k-1}})p(\boldsymbol{x}_k|\omega_{i_k}) \tag{9.6}$$

becomes maximum.

As we have already stated, searching for this maximum requires the computation of Eq. (9.6) for each of the $\Omega_i, i = 1, 2, \ldots, M^N$. This amounts to a total of $O(NM^N)$ multiplications, which is usually a large number indeed. However, this direct computation is a brute-force task. Let us take for example two sequences Ω_i and Ω_j, which we assume differ only in the last class, that is, $\omega_{i_k} = \omega_{j_k}, k = 1, 2, \ldots, N-1$ and $\omega_{i_N} \neq \omega_{j_N}$. It does not take much "scientific thought" to realize that the computation of (9.6) for these two sequences shares all multiplications (which need not be repeated) but one. Furthermore, closer observation of (9.6) reveals that it has a rich computational structure, which can be exploited in order to maximize it in a much more efficient way. This now becomes our next concern.

9.4 THE VITERBI ALGORITHM

Figure 9.1 shows a diagram with N dot columns, where each dot in a column represents one of the M possible classes, $\omega_1, \omega_2, \ldots, \omega_M$. Successive columns correspond to successive observations $\boldsymbol{x}_k, k = 1, 2, \ldots, N$. The arrows determine transitions from one class to another, as observation vectors are obtained in sequence. Thus, *each of the class sequences Ω_i corresponds to a specific path of successive transitions*. Each transition from one class ω_i to another ω_j is characterized by a *fixed* probability $P(\omega_j|\omega_i)$, which is assumed to be known for the adopted model of class dependence. Furthermore, we assume that these probabilities are the same for all successive stages k. That is, the probabilities depend only on the respective class transitions and not on the stage at which they occur. We will further assume that the conditional probability densities $p(\boldsymbol{x}|\omega_i), i = 1, 2, \ldots, M$, are also known to the model. The task of maximizing (9.6) can now be stated as follows. Given a sequence of observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, find the path of successive (class) transitions that

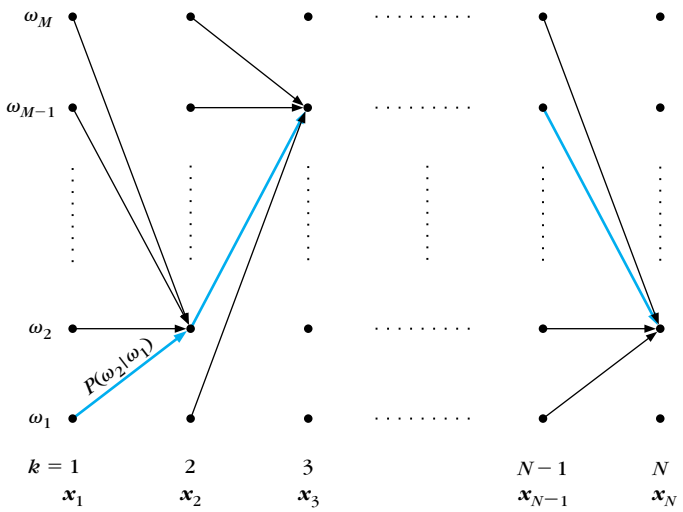

FIGURE 9.1

Trellis diagram for the Viterbi algorithm. The red line denotes the optimal path. The classes along this optimal path will be the ones in which the respective observations are classified.

maximizes (9.6) (e.g., red line in the figure). *The classes along this optimal path will be the ones in which the respective observations are classified*. To search for the optimal path we have to associate a cost with each of the transitions, in agreement with the cost function given in (9.6). A careful look at (9.6) suggests adopting

$$\hat{d}(\omega_{i_k}, \omega_{i_{k-1}}) = P(\omega_{i_k}|\omega_{i_{k-1}})p(\boldsymbol{x}_k|\omega_{i_k}) \tag{9.7}$$

as the cost associated with a transition of a path i from node (class) $\omega_{i_{k-1}}$, at stage $k-1$, to node ω_{i_k}, at stage k *and* at the same time observation $\boldsymbol{x}_k$ occurring. The initial condition for $k = 1$ is given by

$$\hat{d}(\omega_{i_1}, \omega_{i_0}) = P(\omega_{i_1})p(\boldsymbol{x}_1|\omega_{i_1})$$

Using this notation, the overall cost to be optimized becomes

$$\hat{D} \equiv \prod_{k=1}^{N} \hat{d}(\omega_{i_k}, \omega_{i_{k-1}}) \tag{9.8}$$

It will not come as a surprise to us if instead of (9.8) one chooses to maximize

$$\begin{aligned} \ln(\hat{D}) &= \sum_{k=1}^{N} \ln \hat{d}(\omega_{i_k}, \omega_{i_{k-1}}) \\ &\equiv \sum_{k=1}^{N} d(\omega_{i_k}, \omega_{i_{k-1}}) \equiv D \end{aligned} \tag{9.9}$$

where $d(\cdot,\cdot) \equiv \ln \hat{d}(\cdot,\cdot)$. Looking carefully at (9.9) or (9.8), it will not take us long to realize that Bellman's principle can again offer us the means for efficient optimization. Indeed, let us define, in accordance with D, the cost for reaching class ω_{i_k} at stage k via a path i as

$$D(\omega_{i_k}) = \sum_{r=1}^{k} d(\omega_{i_r}, \omega_{i_{r-1}}) \tag{9.10}$$

Then, Bellman's principle states that

$$D_{\max}(\omega_{i_k}) = \max_{i_{k-1}} [D_{\max}(\omega_{i_{k-1}}) + d(\omega_{i_k}, \omega_{i_{k-1}})], \quad i_k, i_{k-1} = 1, 2, \ldots, M \tag{9.11}$$

with

$$D_{\max}(\omega_{i_0}) = 0 \tag{9.12}$$

It is now straightforward to see that the optimal path, which leads to the maximum D in (9.9), is the one that ends at the final stage N in the class $\omega_{i_N}^*$ for which

$$\omega_{i_N}^* = \arg\max_{\omega_{i_N}} D_{\max}(\omega_{i_N}) \tag{9.13}$$

Going back to Figure 9.1, we see that at each stage $k, k = 1, 2, \ldots, N$, there are M possible transitions to each of the nodes ω_{i_k}. The recursive relation in (9.11) suggests that in searching for the maximum we need only keep one of these transitions for every node, *the one that leads to the maximum respective cost* $D_{\max}(\omega_{i_k})$ (red lines in the figure). *Hence, at each stage there are only M surviving paths*. Therefore, the number of operations is M for each node, thus M^2 for all the nodes at each stage, and NM^2 in total. The latter number compares very favorably with the NM^N of the "brute-force task." The resulting algorithm is known as the Viterbi algorithm. This dynamic programming algorithm was first suggested for optimal decoding in convolutional codes in communications [Vite 67].

In previous chapters, we stated that for a number of reasons an alternative to the optimal Bayes classifier can be used. No doubt context-dependent classification can also be emancipated from its Bayesian root. This can easily be achieved by adopting different transition costs $d(\omega_{i_k}, \omega_{i_{k-1}})$, which are not necessarily related to probability densities. In the following sections we will present two typical application areas of the Viterbi algorithm.

Example 9.1

In this example we apply the Viterbi algorithm to compute the optimal paths up to stage $k = 4$, once observation x_4 has been received. Assume that $x_4 = 1.2$ and that the observations reside in the one-dimensional space. Let the task involve three classes, namely, ω_1, ω_2, ω_3. We will further assume that the optimal paths up to stage $k = 3$ have been computed and

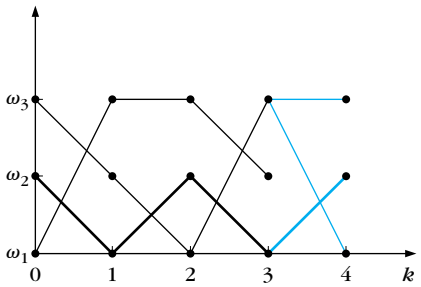

FIGURE 9.2

Optimal paths for the grid corresponding to Example 9.1. The red lines correspond to the extensions of the optimal paths from stage $k = 3$ to stage $k = 4$.

Table 9.1 Transition Costs Between Nodes for Example 9.1

Classes	$\omega_{i_k} = \omega_1$	$\omega_{i_k} = \omega_2$	$\omega_{i_k} = \omega_3$
$\omega_{i_{k-1}} = \omega_1$	0.1	0.7	0.2
$\omega_{i_{k-1}} = \omega_2$	0.4	0.3	0.3
$\omega_{i_{k-1}} = \omega_3$	0.3	0.1	0.6

are shown in black lines in Figure 9.2. Let the optimal costs associated with each class at stage $k = 3$ be equal to

$$D(\omega_1) = -0.5, \quad D(\omega_2) = -0.6, \quad D(\omega_3) = -0.2 \tag{9.14}$$

All the values are negative, since, as Eqs. (9.9) and (9.10) suggest, costs result by summing up logarithms of probability products. The previous costs are assumed known and they have been computed based on the initial class probability values and (a) the transition probability costs among the three classes, which are given in Table 9.1 and (b) the values of the received observations, x_0, x_1, x_2, x_3. We have further assumed that the probability density function describing the emission of an observation from each one of the classes follows a Gaussian distribution

$$p(x|\omega_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$$

where $\mu_1 = 1.0$ and $\sigma_1^2 = 0.03$, $\mu_2 = 1.5$ and $\sigma_2^2 = 0.02$, $\mu_3 = 0.5$ and $\sigma_3^2 = 0.01$.

We will first compute the optimal path reaching class ω_1 at stage $k = 4$. According to Eqs. (9.11), (9.10), and (9.7), the following calculations are in order:

$$\ln p(x_4 = 1.2|\omega_{i_4} = \omega_1) = -0.1578$$

Total cost for the transition from $\omega_{i_3} = \omega_1$ to $\omega_{i_4} = \omega_1$ is equal to
$-0.5 + \ln(0.1) - 0.1578 = -2.9604$.

Total cost for the transition from $\omega_{i_3} = \omega_2$ to $\omega_{i_4} = \omega_1$ is equal to
$$-0.6 + \ln(0.4) - 0.1578 = -1.6741.$$

Total cost for the transition from $\omega_{i_3} = \omega_3$ to $\omega_{i_4} = \omega_1$ is equal to
$$-0.2 + \ln(0.3) - 0.1578 = -1.5617.$$

Hence, the optimal path reaching class ω_1 at stage $k = 4$ is through ω_3 at stage $k = 3$.

For the transitions to ω_2 at $k = 4$, we have

$$\ln p(x_4 = 1.2|\omega_{i_4} = \omega_2) = -0.2591$$

and the respective values for the paths reaching class ω_2 from classes ω_1, ω_2 and ω_3 at $k = 3$ are -1.1158, -2.0631, -2.7617. Thus the optimal path reaching ω_2 at $k = 4$ is through ω_1 at $k = 3$.

Finally, the respective values for the paths reaching ω_3 at $k = 4$ are

$$\ln p(x_4 = 1.2|\omega_{i_4} = \omega_3) = -2.2176$$

and -4.3271, -4.0216, -2.9285. As a result, the best path reaching node ω_3 at stage $k = 4$ goes through class ω_3 at stage $k = 3$ (self-transition).

If $k = 4$ is the final stage, that is, only four observations are available, then the optimal path, denoted by a bold line in Figure 9.2, is the one ending at stage ω_2 with an overall cost equal to -1.1158. Going backwards along the optimal path (backtracking), we assign: x_4 to ω_2, x_3 to ω_1, x_2 to ω_2, x_1 to ω_1 and x_0 to ω_2.

9.5 CHANNEL EQUALIZATION

Channel equalization is the task of recovering a transmitted sequence of information bits I_k (e.g., 1 or 0) after they have been corrupted by the transmission channel and noise sources. The samples received at the receiver end are, thus, given by

$$x_k = f(I_k, I_{k-1}, \ldots, I_{k-n+1}) + \eta_k \tag{9.15}$$

where the function $f(\cdot)$ represents the action of the channel and η_k the noise sequence. The channel contribution to the overall corruption is called the *intersymbol interference (ISI)* and it spans n successive information bits. The equalizer is the *inverse* system whose task is to provide decisions $\hat{I}_k$ about the transmitted information bits I_k, based on l successively received samples $[x_k, x_{k-1}, \ldots, x_{k-l+1}] \equiv \boldsymbol{x}_k^T$. Usually, a delay r must be used in order to accommodate the (possible) noncausal nature of the inverse system. In such cases the decisions made at time k correspond to the I_{k-r} transmitted information bit (Figure 9.3). A simple example will reveal to us how the equalization problem comes under the umbrella of a Markovian context-dependent classification task. Assume a simple linear channel

$$x_k = 0.5I_k + I_{k-1} + \eta_k \tag{9.16}$$

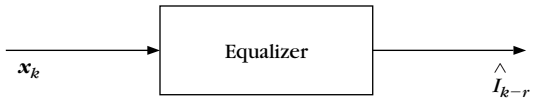

FIGURE 9.3
Block diagram of an equalizer.

For $l = 2$, successively received samples are combined in vectors in the two-dimensional space, that is,

$$\boldsymbol{x}_k^T = [x_k, x_{k-1}]$$

Let us further assume that there are N such observation vectors available. From (9.16) it is readily seen that each $\boldsymbol{x}_k, k = 1, 2, \ldots, N$, depends on the values of three successive information bits, namely I_k, I_{k-1}, I_{k-2}. Neglecting the effects of noise, the possible values of the received samples x_k are given in Table 9.1, together with the respective information bits. Figure 9.4a shows the geometry in the two-dimensional space with (x_k, x_{k-1}) on its axis. When the effect of noise is taken into account, the received vectors are clustered around these points (for example, shaded area). For the specific channel of (9.16) there are eight possible clusters, $\omega_i, i = 1, 2, \ldots, 8$ (Table 9.1). Clusters (red) around "+" correspond to $I_k = 1$ and those (gray) around "o" to $I_k = 0$. In general, the total number of clusters for a binary information sequence is $m = 2^{n+l-1}$. On the reception of each $\boldsymbol{x}_k = [x_k, x_{k-1}]^T$, the equalizer has to decide whether the corresponding transmitted information bit I_k was either a "1" (i.e., class "A") or a "0" (class "B"). In other words, this is nothing other than a two-class classification problem (M class for the M-ary case), and each class consists of a *union of clusters*. Thus, various techniques, from those we have already studied in previous chapters, can be used. A simple way, which was followed in [Theo 95], consists of two steps. During the training period a sequence of known information bits is transmitted, and the representative center $\boldsymbol{\mu}_i$ for each of the clusters is computed by a simple averaging of all the vectors $\boldsymbol{x}_k$ belonging to the respective cluster. This is possible during the *training period*, since the transmitted information bits are known, and thus we know to which of the clusters each received $\boldsymbol{x}_k$ belongs. For example, in the case of Table 9.1 if the sequence of transmitted bits is $(I_k = 1, I_{k-1} = 0, I_{k-2} = 1)$, then the received $\boldsymbol{x}_k$ belongs to ω_6. At the same time, the clusters are labeled as "1" or "0" depending on the value of the I_k bit. At the so-called *decision directed mode* the transmitted information bits are unknown, and the decision about the transmitted I_k is based on which cluster ("1" or "0" label) the received vector $\boldsymbol{x}_k$ is closest to. For this purpose a metric is adopted to define distance. The Euclidean distance of the received vector $\boldsymbol{x}_k$ from the representatives $\boldsymbol{\mu}_i$ of the clusters is an obvious candidate. Although such an equalizer can result in reasonable performance, measured in *bit error rate (BER)* (the percentage of information bits wrongly identified), there is still a great deal of information that has not been exploited. Let us search for it!

Table 9.2 Received Samples and Respective Information Bits for the Channel of Eq. (9.16)

I_k	I_{k-1}	I_{k-2}	x_k	x_{k-1}	Cluster
0	0	0	0	0	ω_1
0	0	1	0	1	ω_2
0	1	0	1	0.5	ω_3
0	1	1	1	1.5	ω_4
1	0	0	0.5	0	ω_5
1	0	1	0.5	1	ω_6
1	1	0	1.5	0.5	ω_7
1	1	1	1.5	1.5	ω_8

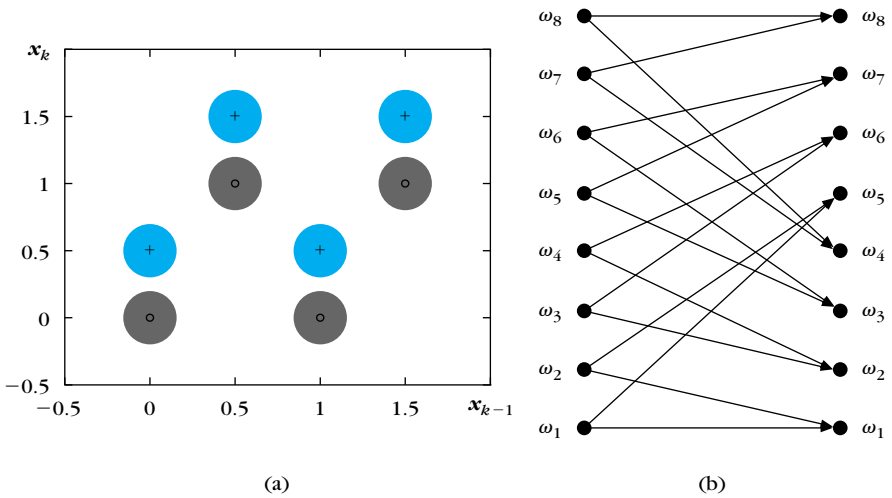

FIGURE 9.4

Plot (a) of the eight possible clusters associated with the channel of Eq. (9.16) and (b) the allowable transitions among them.

From the definition of ISI in (9.15) we know that the channel spans a number of successive information bits. *Thus, only certain transitions among clusters are possible*. Indeed, let us assume, for example, that at time k the transmitted information bits (I_k, I_{k-1}, I_{k-2}) were $(0, 0, 1)$; hence, the corresponding observation vector $\boldsymbol{x}_k$ belongs to cluster ω_2. The next received observation vector $\boldsymbol{x}_{k+1}$ will depend on the triple (I_{k+1}, I_k, I_{k-1}) bits, which can be either $(1, 0, 0)$ or $(0, 0, 0)$. Thus, $\boldsymbol{x}_{k+1}$

will belong either to ω_5 or to ω_1. That is, there are only two possible transitions from ω_2. Figure 9.4b shows the possible transitions among the various clusters. Assuming equiprobable information bits, then from Figure 9.4b we can easily conclude that all *allowable* transitions have probability 0.5, that is,

$$P(\omega_1|\omega_1) = 0.5 = P(\omega_5|\omega_1)$$

and the rest are zero (not allowable). We now have all the ingredients to define the equalization problem as a context-dependent classification task.

Given N observation vectors $\boldsymbol{x}_k, k = 1, 2, \ldots, N$, classify them in a sequence of clusters $\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_N}$. This automatically classifies each $\boldsymbol{x}_k$ in one of the two classes "A" and "B," which is equivalent to deciding that I_k is 1 or 0. For this goal a cost function has to be adopted. In [Theo 95, Geor 97] the cost $d(\omega_{i_k}, \omega_{i_{k-1}})$ in (9.9) used in the Viterbi algorithm, for the allowable transitions, was taken to be

$$d(\omega_{i_k}, \omega_{i_{k-1}}) = d_{\omega_{i_k}}(\boldsymbol{x}_k) \tag{9.17}$$

where $d_{\omega_{i_k}}(\boldsymbol{x}_k)$ is the distance of $\boldsymbol{x}_k$ from the representative of the ω_{i_k} cluster. This can be either the Euclidean

$$d_{\omega_{i_k}}(\boldsymbol{x}_k) = \|\boldsymbol{x}_k - \boldsymbol{\mu}_{i_k}\| \tag{9.18}$$

or the Mahalanobis distance

$$d_{\omega_{i_k}}(\boldsymbol{x}_k) = \left((\boldsymbol{x}_k - \boldsymbol{\mu}_{i_k})^T \Sigma_{i_k}^{-1} (\boldsymbol{x}_k - \boldsymbol{\mu}_{i_k})\right)^{1/2} \tag{9.19}$$

The covariance matrices Σ_{i_k}, describing the distribution of the observation vectors around the respective cluster representatives, are learned during the training period together with the cluster representatives $\boldsymbol{\mu}_i$. The Viterbi algorithm is then used to obtain the optimal overall minimum distance sequence, and recursion (9.11) is modified for the search of the minimum.

An alternative way to define the cost function is to consider observations as scalars x_k, that is, the received samples ($l = 1$), and make the following assumptions:

- Successive noise samples are statistically independent and Gaussian.
- The channel impulse response is known or can be estimated. In practice, a specific channel model is assumed, that is, $\hat{f}(\cdot)$, and its parameters are estimated via an optimization method, for example, least squares [Proa 89].

Under the preceding assumptions, the cost for the allowable state transitions in (9.9) becomes

$$\begin{aligned} d(\omega_{i_k}, \omega_{i_{k-1}}) &= \ln p(x_k|\omega_{i_k}) \equiv \ln(p(\eta_k)) \\ &= -(x_k - \hat{f}(I_k, \ldots, I_{k-n+1}))^2 \end{aligned} \tag{9.20}$$

where η_k is the respective Gaussian distributed noise sample. Obviously, in (9.20) the constants in the Gaussian density function have been omitted. If the Gaussian and independence assumptions are valid, this is obviously a Bayesian optimal

classification to the clusters (from which the "0" and "1" classes result). However, if this is not true, the cost in (9.20) is no longer the optimal choice. This is, for example, the case when the so-called cochannel (nonwhite) interference is present. In such cases, the cluster-based approach is justifiable, and indeed it leads to equalizers with more robust performance [Geor 97]. Furthermore, the fact that in the clustering approach no channel estimation is required can be very attractive in a number of cases, where nonlinear channels are involved and their estimation is not a straightforward task [Theo 95]. In [Kops 03] equalization is performed in the one-dimensional space, that is, $l = 1$. Although this increases the probability of having clusters with different labels to overlap, this is not crucial to the performance, since the Viterbi algorithm has the power to detect the correct label, by exploiting the history in the path. Furthermore, it is pointed out that one needs not determine directly all the 2^n cluster representatives; it suffices to learn, during the training phase, only n of the clusters, and the rest can be obtained by simple arithmetic operations. This is achieved by exploiting the mechanism underlying the cluster formation and the associated symmetries. Both of these observations have a substantial impact on reducing the computational complexity as well as the required length of the training sequence.

The discussion so far has been based on the development of a trellis diagram associated with the transitions among clusters, and the goal has been to unravel the optimal path, using the Viterbi algorithm. However, although this came as a natural consequence of the context-dependent Bayesian classifier, it turns out this is not the most efficient way from a computational point of view. From Figure 9.4b one can easily observe that pairs of clusters jump to the same clusters after transition. For example, the allowable transitions from ω_1 and ω_2 are the same and lead to either ω_1 or ω_5. The same is true for ω_3 and ω_4, and so on. This is because the allowable transitions are determined by the $n + l - 2$ most recent bits. For the example of Figure 9.4, transitions are determined by the pair (I_k, I_{k-1}), which, however, is shared by two clusters, that is, (I_k, I_{k-1}, I_{k-2}), depending on the value of I_{k-2} if it is 0 or 1. The pair (I_k, I_{k-1}) is known as the *state* at time k. This is because, knowing the state at time k and the transmitted bit I_{k+1} at time $k + 1$, we can determine the next state (I_{k+1}, I_k) at time $k + 1$, as is the case in the finite state machines. Since transitions are determined by the states, one can construct a trellis diagram based on the states instead of on clusters. For the example of Figure 9.5, where eight clusters are present, there is a total of four states, that is, $s_1 : (0, 0)$, $s_2 : (0, 1)$, $s_3 : (1, 0)$, $s_4 : (1, 1)$. Figure 9.5 shows these states and the allowable transitions among them. Each transition is associated with one bit, which is the current transmitted bit. Obviously, there is a close relationship between states and clusters. If we know the state transition, that is, $(I_k, I_{k-1}) \rightarrow (I_{k+1}, I_k)$, then the current cluster at time $k + 1$ will be determined by the corresponding values of (I_{k+1}, I_k, I_{k-1}), and this automatically determines the cost of the respective transition, for example, (9.18) or (9.19). Hence the estimates of the transmitted bits are obtained from the sequence of bits along the optimal path, that is, the one with the minimum total cost, in the state trellis diagram instead of the larger cluster trellis diagram.

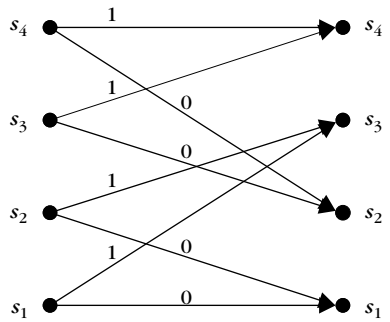

FIGURE 9.5

Plot of the four states associated with the channel of Eq. (9.16) and a two-dimensional equalizer showing the allowable transitions among them.

9.6 HIDDEN MARKOV MODELS

In the channel equalization application of the previous section, the states of the Markov chain were *observable*. That is, given the $l + n - 1$ most recent information bits (i.e., I_k, I_{k-1}, I_{k-2}, in the given example), the state to which the corresponding observation vector $\boldsymbol{x}_k$ belongs is readily known. Thus, during the training period these states can be "labeled," and we can estimate their associated parameters (i.e., the related clusters). In this section, we will be concerned with systems where the states cannot be directly observed. The observations will be considered as the result of an action associated with each state and which is described by a set of probabilistic functions. Moreover, the sequence in which the different states are visited by successive observations is itself the result of another stochastic process, *which is hidden to us*, and the associated parameters describing it can only be inferred by the set of the received observations.

These types of Markov models are known as *hidden Markov models* (HMMs). Let us consider some simple examples of such processes inspired by the well-known coin-tossing problem. We will assume that in all experiments the coin tossing takes place behind a curtain, and all that is disclosed to us is the outcome of each experiment. That is, each time an experiment is performed we cannot know the specific coin (in the case of multiple coins) whose tossing resulted in the current observation (heads or tails). Thus, a crucial part of the probabilistic process is hidden to us.

In the first experiment, a single coin is tossed to produce a sequence of heads (H) and tails (T). This experiment is characterized by a single parameter indicating the propensity of the coin for landing heads. This is quantified by the probability $P(H)$ of H ($P(T) = 1 - P(H)$). A straightforward modeling of this statistical process is to associate one state with the outcome H and one with the outcome T. Hence, this is another example of a process with observable states, since states coincide with the

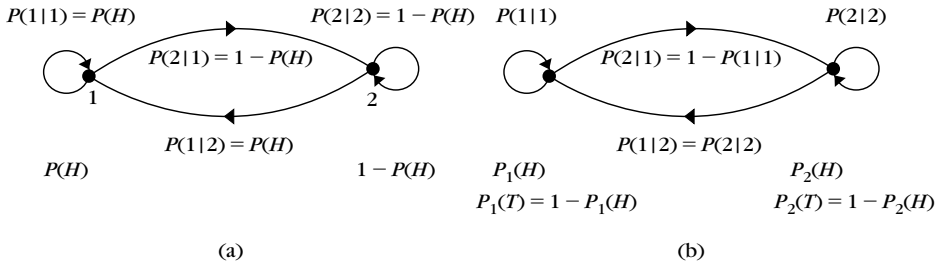

FIGURE 9.6

Markov models for hidden coin-tossing experiments: (a) single coin and (b) two coins.

observations. Figure 9.6a illustrates the underlying generation mechanism of the sequence of observations. $P(i|j)$ denotes the transition probability from state s_j to state s_i, once the coin has been tossed and an observation has been made available to us. For simplicity, states are shown simply as j and i, respectively. For this specific example, state $i = 1$ represents H and state $j = 2$ represents T. Also, for this case, all transition probabilities are expressed in terms of one parameter; that is, $P(H)$. For example, assume that the coin is in state $i = 1$ (H). Tossing the coin again it can either result in H (the coin stays in the same state and $P(1|1) = P(H)$) or it can jump into the other state resulting in T $(P(2|1) = P(T) = 1 - P(H))$.

In the second experiment, we assume that two coins are used behind the curtain. Although, again, the sequence of observations consists of a random succession of heads or tails, it is apparent that this process cannot be sufficiently modeled by a single parameter. To model the process we assume two states, corresponding to the two coins. The model is shown in Figure 9.6b. Two sets of parameters are involved. One of them consists of the probabilities $P_1(H)$ and $P_2(H)$; that is, the probabilities of H for coins 1 and 2, respectively. The other set of parameters are the transition probabilities, $P(i|j), i, j = 1, 2$. For example, $P(1|2)$ is the probability that the current observation (which can be either H or T) is the outcome of an experiment performed using coin 1 (state $i = 1$) and that the previous observation was the result of tossing coin 2 (state $j = 2$). Self-transition probabilities, for example, $P(1|1)$, mean that the same coin (1) is tossed twice and the process remains in the same state ($i = 1$) for two successive times. Taking into account that probabilities of an event add to one, two of the transition parameters are sufficient, and this amounts to a total of four parameters (i.e., $P_1(H)$, $P_2(H)$, $P(1|1)$, $P(2|2)$). It is important to point out that the states of this process are not observable, since we have no access to the information related to which coin is tossed each time.

Figure 9.7 shows the Markov model for the case of tossing three coins behind the curtain. Nine parameters are now required to describe the process; that is, the probabilities $P_i(H)$, $i = 1, 2, 3$, one for each coin and six transition probabilities (the number of transition probabilities is nine but there are three constraints;

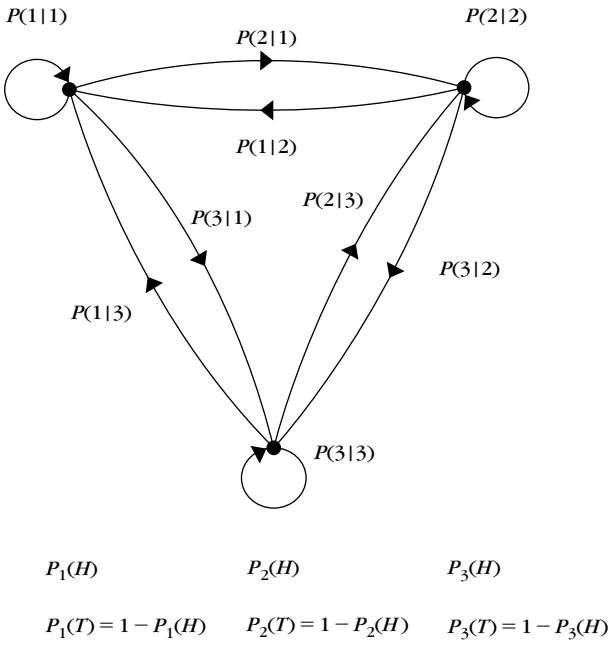

FIGURE 9.7

Markov model for three hidden coins.

i.e., $\sum_{i=1}^{3} P(i|j) = 1, j = 1, 2, 3$). The case of three coins is another example of a probabilistic process with hidden states.

As we will soon see, a major task associated with HMM is first to adopt a model for the underlying process that produces the sequence of observations and then to estimate the unknown set of parameters based on these observations. For the tossing coins examples, these parameters are the transition probabilities as well as the head or tail probabilities for each of the coins. No doubt, adopting the right model is crucial. If, for example, the head or tail observations were produced by tossing two coins and we selected, wrongly, a three-coin model, the resulting estimates of the parameters would lead to an overall poor modeling of the received observations.

In general, an HMM is a type of stochastic modeling appropriate for nonstationary stochastic sequences, with statistical properties that undergo distinct random transitions among a set of different stationary processes. In other words, an HMM models a sequence of observations as a *piecewise stationary process*. Such models have been used extensively in speech recognition to model speech utterances [Bake 75, Jeli 76]. An utterance may be a spoken word, part of a word, or even a complete sentence or a paragraph. The statistical properties of the speech signal within an utterance undergo a series of transitions. For example, a word consists of

subword portions of *voiced* (vowels) and *unvoiced* (consonants) sounds. These are characterized by distinctly different statistical properties, which are in turn reflected in transitions of the speech signal from one statistic to another. Handwriting recognition [Chen 95, Vlon 92, Agaz 93, ElYa 99, Aric 02, Ramd 03], texture classification [Chen 95a, Wu 96], blind equalization [Anto 97, Kale 94, Geor 98], musical pattern recognition [Pikr 06] are some other example applications in which the power of HMM modeling has been successfully exploited.

An HMM model is basically a *stochastic finite state automaton*, which generates an observation string, that is, the sequence of observation vectors, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$. Thus, an HMM model consists of a number of, say K, states and the observation string is produced as a result of successive transitions from one state i to another j. We will adopt the so-called Moore machine model, according to which observations are produced as emissions from the states upon *arrival* (of the transition) at each state.

Figure 9.8 shows a typical diagram of an HMM of three states, where arrows indicate transitions. Such a model could correspond to a short word with three different stationary parts. The model provides information about the successive transitions between the states ($P(i|j), i, j = 1, 2, 3$—temporal modeling of the spoken word) and also about the stationary statistics underlying each state ($p(\boldsymbol{x}|i), i = 1, 2, 3$). This type of HMM model is also known as left to right, because transitions to states with a smaller index are not allowed. Other models also do exist [Rabi 89]. In practice, the states correspond to certain physical characteristics, such as distinct sounds. In speech recognition, the number of states depends on the expected number of such sound phenomena (phonemes)[1] within one word. Actually, a number of states (typically three or four) are used for each phoneme. The average number of observations, resulting from various versions of a spoken word,

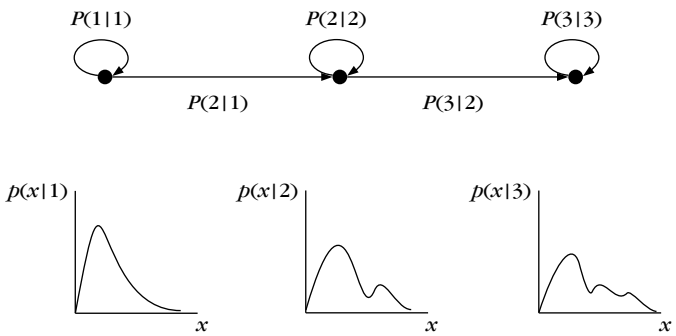

FIGURE 9.8

Model parameters describing a three-state hidden Markov model.

[1] A phoneme is a basic sound unit corresponding to a unique set of articulatory gestures, characterizing the vocal tract articulators for speech sound production [Dell 93].

can also be used as an indication of the number of required states. However, the exact number of states is often the result of experimentation and cannot be determined accurately *a priori*. In blind equalization, the states are associated with the number of clusters formed by the received data [Geor 98]. In character recognition tasks, the states may correspond to line or arc segments in the character [Vlon 92].

In the sequel we will assume that we are given a set of M known reference patterns, and our goal is to *recognize* which one of them an unknown test pattern matches best. This problem was studied in the previous chapter from a template matching (deterministic) perspective. A different (stochastic) path will be taken here. Specifically, we will assume that each known (reference) pattern is described via an HMM model. That is, each of the M patterns is characterized by the following set of parameters:

- The number K_s of the states, $s = 1, 2, \ldots, M$.
- The probability densities $p(\boldsymbol{x}|j), j = 1, 2, \ldots, K_s$, describing the distribution of the observations emitted from state j.
- The transition probabilities $P(i|j), i, j = 1, 2, \ldots, K_s$, among the various states. Some of them can be zero.
- The probabilities $P(i), i = 1, 2, \ldots, K_s$, of the initial state.

Although this is quite a general description of an HMM model, it is worth pointing out that variations are also possible. For example, sometimes the place of the self-transition probability ($P(i|i)$) is taken by the state duration probability distribution, which describes the number of successive stages for which the model stays in state i (Section 9.7). In some other cases, a model for the generation mechanism of the observations is adopted—for example, an autoregressive model [Pori 82] or even a time-varying one that models nonstationary state statistics [Deng 94]. In the sequel we will adhere to the foregoing model. Our problem now consists of two major tasks. One is the *training* for each of the HMM models, that is, the computation of the parameters just listed. The other is the task of *recognition*. That is, once the HMM parameters of the reference models are known, how do we decide which reference model the unknown pattern matches best? We will start with the latter task.

Recognition

Any path method

To start with, we will treat each of the M reference HMM models as a distinct class. The sequence of observations X is *the result of emissions, due to transitions among the different states of the respective model*. The problem then becomes a typical classification task. Given the sequence of N observations $X: \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, resulting from the unknown pattern, decide to which class it belongs. The Bayesian classifier decides in favor of pattern $\mathcal{S}^*$ for which

$$\mathcal{S}^* = \arg\max_{\mathcal{S}} P(\mathcal{S}|X), \quad \text{that is, over all the models} \tag{9.21}$$

and for equiprobable reference models (classes) this is equivalent to

$$\mathcal{S}^* = \arg\max_{\mathcal{S}} p(X|\mathcal{S}) \tag{9.22}$$

where for convenience we have used $\mathcal{S}$ to denote the set of parameters describing each HMM model, that is,

$$\mathcal{S} = \{P(i|j), p(\boldsymbol{x}|i),\ P(i), K_s\}$$

For each model $\mathcal{S}$ there is more than one possible set of successive state transitions Ω_i, each having probability of occurrence $P(\Omega_i|\mathcal{S})$. Thus, recalling the known rule for probabilities, we can write [Papo 91]

$$p(X|\mathcal{S}) = \sum_i p(X, \Omega_i|\mathcal{S}) = \sum_i p(X|\Omega_i, \mathcal{S})P(\Omega_i|\mathcal{S}) \tag{9.23}$$

In order to find the maximum $p(X|\mathcal{S})$ over all possible models, the quantity in (9.23) has to be computed for each of the M reference models. Its efficient computation can be achieved via an approach similar to the Viterbi algorithm. Indeed, the only difference between (9.23) and (9.6) is that, instead of simply searching for the maximum over all possible state sequences Ω_i, (9.23) requires summing up the respective values for each of them. To this end, let us define as $\alpha(i_k)$ the probability density of the joint event: (a) a path is at state $i_k (i_k \in \{1, 2, \ldots, K_s\})$ at stage k *and* (b) observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{k-1}$ have been emitted at the previous stages *and* (c) observation $\boldsymbol{x}_k$ is emitted from the state i_k at stage k. From the definition of $\alpha(i_k)$ the following recursive relation is easily understood:

$$\begin{aligned}\alpha(i_{k+1}) &\equiv p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k+1}, i_{k+1}|\mathcal{S}) \\ &= \sum_{i_k} \alpha(i_k)P(i_{k+1}|i_k)p(\boldsymbol{x}_{k+1}|i_{k+1}), \quad k = 1, 2, \ldots, N-1\end{aligned} \tag{9.24}$$

with

$$\alpha(i_1) = P(i_1)p(\boldsymbol{x}_1|i_1)$$

The product $P(i_{k+1}|i_k)p(\boldsymbol{x}_{k+1}|i_{k+1})$ in (9.24) provides the local information for the last transition, and $\alpha(i_k)$ is the information accumulated from the path history up to stage k. As is apparent from its definition, $\alpha(i_k)$ does not depend on the subsequent observations $\boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_N$. The definition of a joint probability density function, which depends on *all* available observations, is also possible and will be used later on. To this end, let us define $\beta(i_k)$ as the probability density function of the event: observations $\boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_N$ occur at stages $k+1, \ldots, N$, given that at stage k the path is at state i_k. Then after a little thought we conclude that $\beta(i_k)$ obeys the following recursion:

$$\begin{aligned}\beta(i_k) &\equiv p(\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+2}, \ldots, \boldsymbol{x}_N|i_k, \mathcal{S}) \\ &= \sum_{i_{k+1}} \beta(i_{k+1})P(i_{k+1}|i_k)p(\boldsymbol{x}_{k+1}|i_{k+1}), \quad k = N-1, \ldots, 1\end{aligned} \tag{9.25}$$

where by definition

$$\beta(i_N) = 1, i_N \in \{1, 2, \ldots, K_s\} \tag{9.26}$$

Thus, the probability density of the joint event: (a) a path is at state i_k at stage k *and* (b) $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ have been observed, is given by

$$\begin{aligned}\gamma(i_k) &\equiv p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, i_k|\mathcal{S}) \\ &= p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, i_k|\mathcal{S}) p(\boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_N|i_k, \mathcal{S}) \\ &= \alpha(i_k)\beta(i_k)\end{aligned} \tag{9.27}$$

where the assumption about the observations' independence has been employed. Equation (9.27) also justifies the choice $\beta(i_N) = 1, i_N \in \{1, 2, \ldots, K_s\}$.

Let us now return to our original goal of computing the maximum $p(X|\mathcal{S})$. Equation (9.24) suggests that we can write Eq. (9.23) as

$$p(X|\mathcal{S}) = \sum_{i_N=1}^{K_s} \alpha(i_N) \tag{9.28}$$

For the computation of (9.28), we need to compute all $\alpha(i_k)$, for $k = 1, 2, \ldots, N$. This is efficiently achieved using the diagram of Figure 9.9. Each node corresponds to a stage k and a state $i_k, i_k = 1, 2, \ldots, K_s$. For each of the nodes the density

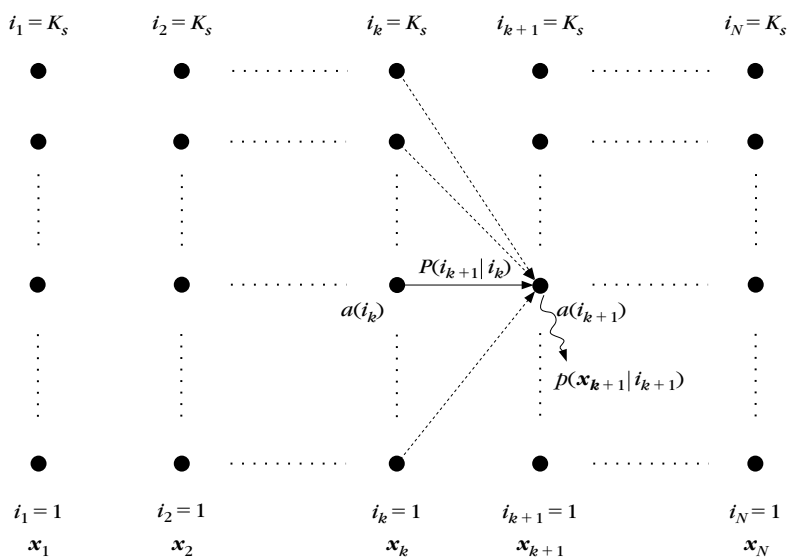

FIGURE 9.9

Diagram showing the computational flow for the any path method.

$\alpha(i_k)$ from Eq. (9.24) is computed. Thus, the number of computations is of the order of NK_s^2. The resulting algorithm is known as the *any path* method, since all paths participate in the final cost. The computation of (9.28) is performed for each of the M models, and the unknown string pattern of the observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ is classified to the reference model $\mathcal{S}$ for which $p(X|\mathcal{S})$ becomes maximum.

Best Path Method

An alternative, suboptimal, approach is the so-called *best path* method. According to this method, for a given observation sequence X, we compute the most probable (best) path of states sequence *for each* of the reference models. The task now becomes that of (9.1), and the search for each of the optima can be achieved efficiently via the Viterbi algorithm, with the cost D given as in (9.9),

$$D = \sum_{k=1}^{N} d(i_k, i_{k-1}) \tag{9.29}$$

$$d(i_k, i_{k-1}) = \ln P(i_k|i_{k-1}) + \ln p(\boldsymbol{x}_k|i_k)$$

In other words, for each of the models we compute the maximum of $P(\Omega_i)\, p(X|\Omega_i) = p(\Omega_i, X)$. Hence, the summation in (9.23) is replaced by a maximum operation. The unknown pattern is classified to that reference model $\mathcal{S}$ for which the resulting optimal cost D is maximum.

Training

Training is a more difficult task. The states now are not observable, and a direct training approach, such as the one used in Section 9.5, cannot be adopted. The parameters that define each HMM model $\mathcal{S}$ can *only* be inferred from the available observations.

One way to achieve this goal is to estimate the unknown parameters, so that the output for each model, (9.29) or (9.28), becomes maximum *for a training set of observations known to belong to the model*. This is an optimization task with a nonlinear cost function, and it is carried out iteratively. To this end, assumptions about the probability density functions $p(\boldsymbol{x}|i)$ are required. The procedure can be simplified if one assumes that the observations $\boldsymbol{x}_k$ can take only discrete values. In such cases probability density functions $p(\boldsymbol{x}|i)$ become probabilities, $P(\boldsymbol{x}|i)$.

Discrete Observation HMM Models

We will assume that the training observation string $\boldsymbol{x}_k, k = 1, 2, \ldots, N$, consists of quantized vectors. In practice, this is achieved via vector quantization techniques, to be discussed later in Chapter 14. Hence, each observation vector can take one only out of L possible distinct values in the l-dimensional space. Thus, observations

can be described as integers $r, r = 1, 2, \ldots, L$. The steps for each of the two methods, that is, any path and best path, are as following:

Baum–Welch Reestimation

The "output" quantity in the any path procedure is $p(X|\mathcal{S})$. Thus, estimating the parameters of the model $\mathcal{S}$ so that $p(X|\mathcal{S})$ is a maximum is nothing but a *maximum likelihood* parameter estimation procedure. Before going into the discussion of the iteration steps some definitions are needed.

Definitions

- $\xi_k(i,j,X|\mathcal{S}) \equiv$ the probability of the joint event: (a) a path passes through state i at stage k *and* (b) through state j at the next stage $k+1$ *and* (c) the model generates the available sequence of observations X, given the parameters of the model $\mathcal{S}$.
- $\gamma_k(i|X,\mathcal{S}) \equiv$ the probability of the event: a path passes through state i at stage k *given* the model and the available observation sequence.

From these definitions, it is not difficult to show that

$$\xi_k(i,j) \equiv \xi_k(i,j|X,\mathcal{S}) = \frac{\xi_k(i,j,X|\mathcal{S})}{P(X|\mathcal{S})} \tag{9.30}$$

Mobilizing the definitions in Eqs. (9.24) and (9.25), Eq. (9.30) becomes

$$\xi_k(i,j) = \frac{\alpha(i_k = i)P(j|i)P(\boldsymbol{x}_{k+1}|j)\beta(i_{k+1} = j)}{P(X|\mathcal{S})} \tag{9.31}$$

where $\alpha(i_k = i)$ accounts for the path history terminating at stage k and state i. $\beta(i_{k+1} = j)$ accounts for the future of the path, which at stage $k+1$ is at state j and then evolves *unconstrained* until the end. The product $P(j|i)P(\boldsymbol{x}_{k+1}|j)$ accounts for the local activity at stage k. The other quantity of interest is given by

$$\gamma_k(i) \equiv \gamma_k(i|X,\mathcal{S}) = \frac{\alpha(i_k = i)\beta(i_k = i)}{P(X|\mathcal{S})} \tag{9.32}$$

From the foregoing it is not difficult to see that

- $\sum_{k=1}^{N} \gamma_k(i)$ can be regarded as the expected (over the number of stages) number of times state i occurs, given the model $\mathcal{S}$ and the observation sequence X. When the upper index in the summation is $N-1$, this quantity is the expected number of transitions from state i.
- $\sum_{k=1}^{N-1} \xi_k(i,j)$ can be regarded as the expected number of transitions from state i to state j, given the model and the observation sequence.

The preceding definitions lead us to adopt the following (re)estimation formulas as reasonable estimates of the unknown model parameters.

$$\bar{P}(j|i) = \frac{\sum_{k=1}^{N-1} \xi_k(i,j)}{\sum_{k=1}^{N-1} \gamma_k(i)} \tag{9.33}$$

$$\bar{P}_{\boldsymbol{x}}(r|i) = \frac{\sum_{(k=1 \text{ and } \boldsymbol{x} \to r)}^{N} \gamma_k(i)}{\sum_{k=1}^{N} \gamma_k(i)} \tag{9.34}$$

$$\bar{P}(i) = \gamma_1(i) \tag{9.35}$$

The numerator in (9.34) sums only those of $\gamma_k(i)$ for which the corresponding observation $\boldsymbol{x}_k$ takes the rth discrete value. The iterative algorithm can now be expressed in terms of the following steps:

Iterations

- Initial conditions: Assume initial conditions for the unknown quantities. Compute $P(X|\mathcal{S})$.
- Step 1: From the current estimates of the model parameters reestimate the new model $\bar{\mathcal{S}}$ via Eqs. (9.33) to (9.35).
- Step 2: Compute $P(X \mid \bar{\mathcal{S}})$. If $P(X \mid \bar{\mathcal{S}}) - P(X \mid \mathcal{S}) > \epsilon$ set $\mathcal{S} = \bar{\mathcal{S}}$ and go to step 1. Otherwise stop.

Remarks

- Each iteration improves the model $\bar{\mathcal{S}}$; that is, it is true that $P(X|\bar{\mathcal{S}}) \geq P(X|\mathcal{S})$.
- The algorithm may lead to local maxima; see, for example, [Baum 67, Baum 68, Baum 70]. This is why the algorithm in practice runs a number of times, starting from different initial conditions, in order to find a favorable local maximum for $P(X|\mathcal{S})$. Other computational issues, including parallelism and memory requirements, are treated in [Turi 98].
- The Baum–Welch algorithm is basically an implementation of the EM algorithm, which was introduced in Chapter 2. Indeed, a little thought reveals that the ML estimation of the HMM parameters is a typical ML problem with an incomplete data set, that is, the unobserved states (e.g., [Moon 96, Diga 93]). A generalization of the method that allows multiple observation training sequences is given in [Li 00]. Other, gradient-based, optimizing techniques for the estimation of the unknown parameters have also been suggested and used [Levi 83].
- Practical implementation issues:
  1. *Scaling:* The probabilities $\alpha(i_k), \beta(i_k)$ are obviously less than one, and their values tend to zero very fast as the number of terms in the products (9.24) and (9.25) increases. In practice, the dynamic range of their computed values may exceed the precision range of the computer, so

appropriate scaling is required. A basic procedure is to scale $\alpha(i_k)$ in proportion to the number of stages. If the same scaling factor is used for the $\beta(i_k)$, then on taking their product in the recursions the effect of the scaling cancels out [Levi 83, Rabi 89] (Problem 9.4).

2. *Initial conditions:* This is an omnipresent problem in all iterative optimization algorithms. Usually, the unknown parameters are initialized randomly, subject, of course, to the constraints of the problem. That is, if some transitions are not allowed, the corresponding probabilities are set to zero, and also probabilities must add to one.

3. *Insufficient training data:* Generally, a large amount of training data is necessary to learn the HMM parameters. The observation sequence must be sufficiently long with respect to the number of states of the HMM model. This will guarantee that all state transitions will appear a sufficient number of times, so that the reestimation algorithm learns their respective parameters. If this not the case, a number of techniques have been devised to cope with the issue. For a more detailed treatment the reader may consult [Rabi 89, Dell 93] and the references therein.

Viterbi Reestimation

In the speech literature the algorithm is also known as the *segmental k-means training* algorithm [Rabi 89]. It is related to the best path method.

Definition 1.

- $n_{i|j} \equiv$ *number of transitions from state j to state i.*
- $n_{|j} \equiv$ *number of transitions originated from state j.*
- $n_{i|} \equiv$ *number of transitions terminated at state i.*
- $n(r|i) \equiv$ *number of times observation* $r \in \{1, 2, \ldots, L\}$ *occurs jointly with state i.*

Iterations

- Initial conditions: Assume the initial estimates of the unknown parameters. Obtain the best path and compute D.
- Step 1: From the available best path, reestimate the new model parameters as:

$$\bar{P}(i|j) = \frac{n_{i|j}}{n_{|j}}$$

$$\bar{P}\boldsymbol{x}(r|i) = \frac{n(r|i)}{n_{i|}}$$

- Step 2: For the new model parameters obtain the best path and compute the corresponding overall cost $\bar{D}$. Compare it with the cost D of the previous iteration. If $\bar{D} - D > \epsilon$ set $D = \bar{D}$ and go to step 1. Otherwise stop.

Symbol $\bar{P}\boldsymbol{x}(r|i)$ is the current iteration estimate of the probability of emitting from state i the rth value from the available palette of the L possible vectors. The preceding algorithm has assumed that the initial state is known; thus no estimation of the respective probabilities is needed. This is, for example, true for left-to-right models, such as the one shown in Figure 9.8. The Viterbi reestimation algorithm can be shown to converge to a proper characterization of the underlying observations [Fu 82, Lee 72].

Continuous Observation HMM

The discrete observation modeling of originally continuous variables suffers from a serious drawback. During the (vector) quantization stage of the signal (e.g., speech segment), a severe loss of information about the original waveform may occur, which can seriously degrade the performance of the recognizer. The alternative is to work with continuous observation modeling, albeit at the expense of higher complexity. This approach requires modeling of the probability densities $p(\boldsymbol{x}|i)$, prior to estimation. Once these have been estimated, the recognition problem evolves along the same lines as with the discrete observation case. The difference exists only in the training task. One way to approach the problem is to assume a parametric model for the probability density function and then use reestimation procedures to compute the unknown model parameters. As we have already discussed in Chapter 2, a very general parameterization of the probability density function is via mixture modeling, that is,

$$p(\boldsymbol{x}|i) = \sum_{m=1}^{L} c_{im} F(\boldsymbol{x}, \boldsymbol{\mu}_{im}, \Sigma_{im}) \tag{9.36}$$

where $F(\cdot, \cdot, \cdot)$ is a density function and $\boldsymbol{\mu}_{im}, \Sigma_{im}$ are the mean vector and the covariance matrix of the mth mixture. We will adhere to Gaussian functions, which are usually employed in practice. The mixture coefficients c_{im} have to satisfy the constraint

$$\sum_{m=1}^{L} c_{im} = 1, \quad 1 \le i \le K_s$$

so that

$$\int_{-\infty}^{+\infty} p(\boldsymbol{x}|i)\, d\boldsymbol{x} = 1, \quad 1 \le i \le K_s$$

Following arguments similar to those used to reestimate the parameters in the discrete observation HMM case, the following reestimation formulas are obtained [Lipo 82, Juan 85, Juan 86].

$$\bar{c}_{im} = \frac{\sum_{k=1}^{N} \gamma_k(i, m)}{\sum_{k=1}^{N} \sum_{r=1}^{L} \gamma_k(i, r)} \tag{9.37}$$

$$\bar{\boldsymbol{\mu}}_{im} = \frac{\sum_{k=1}^{N} \gamma_k(i, m) \boldsymbol{x}_k}{\sum_{k=1}^{N} \gamma_k(i, m)} \tag{9.38}$$

$$\bar{\Sigma}_{im} = \frac{\sum_{k=1}^{N} \gamma_k(i,m)(\boldsymbol{x}_k - \bar{\boldsymbol{\mu}}_{im})(\boldsymbol{x}_k - \bar{\boldsymbol{\mu}}_{im})^T}{\sum_{k=1}^{N} \gamma_k(i,m)} \tag{9.39}$$

The term $\gamma_k(i,m)$ is the probability density of being at state i and stage k with the mth mixture component accounting for $\boldsymbol{x}_k$, that is,

$$\gamma_k(i,m) = \frac{\alpha(i_k = i)\beta(i_k = i)}{\sum_{i_k=1}^{K_s} \alpha(i_k)\beta(i_k)} \times \frac{c_{im} F(\boldsymbol{x}_k, \boldsymbol{\mu}_{im}, \Sigma_{im})}{\sum_{r=1}^{L} c_{ir} F(\boldsymbol{x}_k, \boldsymbol{\mu}_{ir}, \Sigma_{ir})} \tag{9.40}$$

where c_{im} is the ratio of the expected number of times the system is at state i using the mth mixture component to the overall expected number of times the system is at state i. Similar interpretations can be made for the other formulas too.

When the Viterbi method is employed, reestimation of the parameters is based on averages computed across the best path. For example, for mixture modeling using a single Gaussian ($L = 1$) we get

$$\boldsymbol{\mu}_i = \frac{1}{N_i} \sum_{k=1}^{N} \boldsymbol{x}_k \delta_{ik}$$

$$\Sigma_i = \frac{1}{N_i} \sum_{k=1}^{N} (\boldsymbol{x}_k - \boldsymbol{\mu}_i)(\boldsymbol{x}_k - \boldsymbol{\mu}_i)^T \delta_{ik}$$

where $\delta_{ik} = 1$ for the stages where the path goes through state i and is zero otherwise, and N_i is the respective number of times the path passes through state i.

Remarks

- The algorithms just described estimate the unknown parameters using all the available observations simultaneously. An alternative path, of major practical importance, is to employ adaptive techniques in which new information can be incorporated to adapt an already trained model, without it being necessary to retrain it with all previously used data. It is generally accepted that speaker-dependent recognizers outperform speaker-independent systems, as long as sufficient training data are available. Thus, a long-standing idea is to use speaker-independent recognizers, trained with enough data on a multispeaker platform, and then adapt the model parameters to fit a specific speaker (and/or acoustic environment). This can be achieved by using the minimum number of data from the new speaker. Both batch and sequential schemes have been suggested. Some examples of such learning procedures are given in [Lee 91, Diga 95, Legg 95, Huo 95, Huo 97, Diga 93, Wang 01].
- A drawback of the modeling in (9.36) is that a mixture model is adopted for each of the states. This makes the number of parameters to be estimated rather high. Thus, for a given size of training data, it affects the robustness of the parameter estimation. To alleviate such problems and decrease the number of unknown parameters, so-called tied-mixture densities modeling has been

suggested, where the same Gaussian densities are shared across the mixtures of all the states [Bell 90] or groups of states [Diga 96, Kim 95, Gale 99, Gu 02].

- The Baum–Welch algorithm is an iterative procedure to maximize the likelihood function with respect to the unknown parameters. MAP procedures incorporating prior statistical information have also been proposed, and enhanced performance has been reported [Gauv 94]. An alternative is to optimize with respect to all the unknown parameters, instead of optimizing each HMM model separately, as was the case with ML earlier. The goal of such an optimization approach is to enhance the discrimination capabilities of the models, see, for example, [He 08]. Maximizing the mutual information [Bahl 86] or minimizing the cross-entropy [Ephr 89] and, more recently, the classification error rate by using either a smooth version of it [Juan 92, Juan 97] or the deterministic annealing technique [Rao 01] or controlling the influence of the outliers [Arsl 99] are examples of approaches that enhance performance at the expense of complexity.

  More recently, [Li 04] suggested the use of a deterministic annealing technique that allows one to adapt the number of states during training. This can offer some advantages for those cases where the number of states cannot be accurately predetermined.

- HMM are graphical models and they belong to a class of Bayesian networks (discussed in Chapter 2) known as dynamic Bayesian networks [Neap 04].

9.7 HMM WITH STATE DURATION MODELING

Hidden Markov modeling, as we have approached it so far, falls short of expectations in many cases in practice. Experimental evidence has identified a serious shortcoming associated with the use of the self-transition probabilities, $P(i|i)$, as parameters in the standard HMMs. This is related to the exponential modeling of the state duration probability, $P_i(d)$, that such a modeling implies, where d is the successive number of times the model remains in state i. Indeed, given $P(i|i)$ the probability of a path leaving current state i is equal to $1 - P(i|i)$. Hence, the probability of being in state i for d successive instants (i.e., $d - 1$ self-transitions, and emission of d consecutive observations from state i) is given by

$$P_i(d) = (P(i|i))^{d-1} (1 - P(i|i)) \tag{9.41}$$

For many cases (e.g., for a number of audio signals), such an exponential state duration modeling is inappropriate. To alleviate this drawback, it has been suggested to substitute the self-transition probability, $P(i|i)$, by an explicit variable state duration

probability $P_i(d)$ in the set of unknown HMM parameters [Ferg 80]. Thus, under this new setting, the set of the unknown parameters defining an HMM consists of

- The number K_s of the states.
- The probability densities $p(\boldsymbol{x}|j)$ (they become probabilities for the case of discrete observation models, i.e., $\boldsymbol{x} \in \{1, 2, \ldots, L\}$).
- The state transition probabilities $P(i|j),\ i,j = 1, 2, \ldots, K_s$.
- The state duration probabilities, $P_i(d),\ i = 1, 2, \ldots, K_s,\ 1 \leq d \leq D$.
- The probabilities $P(i),\ i = 1, 2, \ldots, K_s$, of the initial state.

Observe that a maximum allowable state duration D has been adopted. Thus, the model $\mathcal{S}$ can now be written as

$$\mathcal{S} = \{P(i|j), P_i(d), p(\boldsymbol{x}|i), P(i), K_s\}$$

Our goal remains the computation of the maximum of $P(X|\mathcal{S})$ in (9.23). To achieve this in an efficient way, we have to modify the set of auxiliary variables used with the standard HMM so that we can adapt to the needs of the new parameterization. To this end, define $\alpha_k(i)$ to be the probability density of the joint event: (a) a stay at state i ends at stage k *and* (b) observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ have been emitted up to stage k. That is,

$$\alpha_k(i) = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k,\ \text{state } i \text{ ends at stage } k|\mathcal{S}) \tag{9.42}$$

Note the slightly different notation used here, compared to (9.24), to emphasize the different meaning of the involved variables. Since the stay at state i ends at stage k, the next state, at stage $k+1$, can take any value *except* i; that is, $i_{k+1} \neq i$. Furthermore, looking at the path history up to stage k, there are various ways to reach state $i_k = i$. One is to jump to $i_k = i$ from an $i_{k-1} \neq i$, and this suggests that only one symbol is emitted from state i. The probability of this event depends on the value of $P_i(d = 1)$, since we know that the path will depart from state i at $k+1$. The second possibility is the path to be at state i at stage $k-1$ and remain there for two successive stages (i.e., $i_{k-1} = i_k = i$). The probability of such an event is equal to $P_i(d = 2)$. This rationale can be pushed backward $D-1$ steps prior to k (i.e., $i_{k-D+1} = \cdots = i_k = i$), and the probability of this event is given by $P_i(D)$. This, of course, can be applied to all stages prior to k (i.e., $1, 2, \ldots, k-1$). From this discussion, it is not difficult to write the counterpart of recursion (9.24) as

$$\alpha_k(i) = \sum_{(j=1,\, j\neq i)}^{K_s} \sum_{d=1}^{D} \alpha_{k-d}(j) P(i|j) P_i(d) \prod_{m=k-d+1}^{k} p(\boldsymbol{x}_m|i) \tag{9.43}$$

where, once more, statistical independence between observations has been assumed. Initialization of (9.43) requires the following computations as it can easily be understood from the respective definitions:

$$\alpha_1(j) = P(j) P_j(1) p(\boldsymbol{x}_1|j)$$

For $k = 2, 3, \ldots, D$ and $j = 1, 2, \ldots, K_s$,

$$\alpha_k(j) = P(j)P_j(k)\prod_{m=1}^{k} p(\boldsymbol{x}_m|j) + \sum_{(r=1,\ r\neq j)}^{K_s}\sum_{d=1}^{k-1}\alpha_{k-d}(r)P(j|r)P_j(d)\prod_{m=k-d+1}^{k} p(\boldsymbol{x}_m|j)$$

As was the case with the standard form of HMMs during the recognition phase, the desired quantity $p(X|\mathcal{S})$ can now be obtained from

$$p(X|\mathcal{S}) = \sum_{i=1}^{K_s}\alpha_N(i) \tag{9.44}$$

which is easily understood from the respective definitions and can efficiently be obtained by the repeated computation of Eq. (9.43) over all ks and is.

For the training phase, in order to derive reestimation formulas for the set of the unknown parameters $(P(i|j), P(i), P_i(d))$ and for a given number of states K_s the following auxiliary variables need to be defined [Rabi 93]. Variable $\alpha_k^*(i)$ is the probability density (probability for the discrete observations case) of the joint event: (a) a path starts its stay at state i at stage $k+1$ *and* (b) observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ have been emitted. That is,

$$\alpha_k^*(i) = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k,\ \text{state } i \text{ starts at stage } k+1|\mathcal{S})$$

From the respective definitions, the following are easily established.

$$\alpha_k^*(i) = \sum_{j=1,\ j\neq i}^{K_s}\alpha_k(j)P(i|j) \tag{9.45}$$

$$\alpha_k(i) = \sum_{d=1}^{D}\alpha_{k-d}^*(i)P_i(d)\prod_{m=k-d+1}^{k} p(\boldsymbol{x}_m|i) \tag{9.46}$$

In addition, let $\beta_k(i)$ be the *conditional* probability density of the event: observations $\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+2}, \ldots, \boldsymbol{x}_N$ have been observed, given that the path *ends* at state i and at stage k. That is,

$$\beta_k(i) = p(\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+2}, \ldots, \boldsymbol{x}_N|\text{the path ends its stay at state } i \text{ at stage } k, \mathcal{S})$$

Also, $\beta_k^*(i)$ is the conditional probability density of the event: observations $\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+2}, \ldots, \boldsymbol{x}_N$ have been observed, given that the path *starts* its stay at state i at stage $k+1$. That is,

$$\beta_k^*(i) = p(\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+2}, \ldots, \boldsymbol{x}_N|\text{the path starts its stay at state } i \text{ and stage } k+1, \mathcal{S})$$

The previous definitions allow us to write

$$\beta_k(i) = \sum_{j=1, j\neq i}^{K_s}\beta_k^*(j)P(j|i) \tag{9.47}$$

$$\beta_k^*(i) = \sum_{d=1}^{D} \beta_{k+d}(i) P_i(d) \prod_{m=k+1}^{k+d} p(\boldsymbol{x}_m|i) \tag{9.48}$$

Here, via the definitions, the following initial conditions hold (combined with (9.47)):

$$\beta_N(i) = 1, \quad i = 1, 2, \ldots, K_s \tag{9.49}$$

and

$$\beta_k^*(i) = \sum_{d=1}^{N-k} \beta_{k+d}(i) P_i(d) \prod_{k+1}^{k+d} p(\boldsymbol{x}_m|i), \quad k = N-1, \ldots, N-D \tag{9.50}$$

Based on the previous auxiliary variables and the derived relationships, the following reestimation formulas for the set of unknown parameters are obtained for the *discrete observations* case ($\boldsymbol{x}_k \to r \in \{1, 2, \ldots, L\}$).

$$\bar{P}(i) = \frac{P(i)\beta_0^*(i)}{P(X|\mathcal{S})} \tag{9.51}$$

$$\bar{P}(j|i) = \frac{\sum_{k=1}^{N-1} \alpha_k(i) P(j|i) \beta_k^*(j)}{\sum_{j=1}^{K_s} \sum_{k=1}^{N-1} \alpha_k(i) P(j|i) \beta_k^*(j)} \tag{9.52}$$

$$\bar{P}_{\boldsymbol{x}}(r|i) = \frac{\sum_{k=1, \boldsymbol{x}_k \to r}^{N} \left(\sum_{m<k} \alpha_m^*(i) \beta_m^*(i) - \sum_{m<k} \alpha_m(i) \beta_m(i) \right)}{\sum_{r=1}^{L} \sum_{k=1, \boldsymbol{x}_k \to r}^{N} \left(\sum_{m<k} \alpha_m^*(i) \beta_m^*(i) - \sum_{m<k} \alpha_m(i) \beta_m(i) \right)} \tag{9.53}$$

$$\bar{P}_i(d) = \frac{\sum_{k=1}^{N-d} \alpha_k^*(i) P_i(d) \beta_{k+d}(i) \prod_{m=k+1}^{k+d} P(\boldsymbol{x}_m|i)}{\sum_{d=1}^{D} \left(\sum_{k=1}^{N-d} \alpha_k^*(i) P_i(d) \beta_{k+d}(i) \prod_{m=k+1}^{k+d} P(\boldsymbol{x}_m|i) \right)} \tag{9.54}$$

Equation (9.51) is straightforward from the definitions and is an implication of the Bayes theorem. Equation (9.52) is the total number of path transitions from state i to state j along all the stages, divided by the total number of transitions that occur from state i. Equation (9.54) is the ratio of the number of times the path starts its stay at state i with duration d, divided by the number of times state i occurs with any duration.

Equation (9.53) needs a bit more elaboration. The numerator is the number of times observation $\boldsymbol{x}_k \in \{1, 2, \ldots, L\}$ is emitted from state i. To be simultaneously at state i and stage k means that a path can either start its stay at state i at stage k or may have started to be at this state at a previous stage (i.e., at $k-1, k-2, \ldots$) and remain there for a corresponding number of successive instants. However, there is a finite probability for a path to start being at state i at a stage earlier than k but not to survive long enough at this state for us to have the chance to "meet" it there at

stage k. The term $\alpha^*_m(i)\beta^*_m(i)$ is the probability that a path starts its stay at state i at some stage $m+1$, and the term $\alpha_{m+1}(i)\beta_{m+1}(i)$ is the probability that a path ends its stay at state i at stage $m+1$. The first summation is the total probability that a path starts its stay at state i at any stage up to k. The second summation is the total probability that a path ends its stay at i at any stage prior to k. Hence, the subtraction of the two summation terms gives the probability of having a path through i at stage k, for the *given observation sequence*. The denominator is the same quantity, but the summation is over all times that state i is visited by a path, regardless of the emitted observation.

Adopting HMM with an explicit state duration probability modeling improves the performance in many recognition tasks compared to the standard HMM. The cost one pays for such an improvement is the increased computational complexity. The storage requirements are increased by an order of D and the computational cost by an order of D^2. Besides it, the state duration model requires D more parameters to be estimated, in addition to those in the standard HMM. This, unavoidably, leads to a demand for longer training sequences to safeguard enough data for the accurate estimation of all unknown parameters. Some of these problems can be minimized by adopting a parametric model for $P_i(d)$, so that the number of unknown parameters is reduced to the number of parameters describing the parametric probability function. To this end, Gaussian, Poisson, and Gamma distribution models have been used [Levi 86, Russ 85]. In [Chie 03] the case of adapting the parameters of the adopted parametric state duration model is treated, to fit nonstationary speech variations for large vocabulary continuous speech recognition.

Best Path Method

For the recognition phase, given a trained HMM and an observation sequence X, the goal now becomes to compute the probability of the most probable path of states sequence. However, this can no longer be achieved by employing the Viterbi algorithm in the form given in Section 9.4. To adapt to the needs of the new parameterization, imposed by the explicit time duration modeling, we have to attack the problem in a slightly different way. For the computation of the best path, up to a node corresponding to stage k and state i, there are now two types of competing paths: (a) paths that end their stay in a state j *different* to i at stage $k-1$ and jump to the node (k,i) and (b) paths that end their stay in a $j \neq i$ state at previous stages, $k-2,\ldots,k-D$, and then jump to state i and remain there for $2,\ldots,D$, time instants, respectively. Let $a_k(i)$ be the optimal cost up to stage k and state i. According to Bellman's principle, for the computation of $a_k(i)$ the following are valid:

- For paths through $(k-1,j)$, $j = 1,2,\ldots,K_s$, $j \neq i$ and then jumping to i,

$$a_k(i) = a_{k-1}(j)P(i|j)p(\boldsymbol{x}_k|i)P_i(1), \quad j \neq i \tag{9.55}$$

- For paths through nodes $(k-d, j)$, $j \neq i$, $d = 2, \ldots, D$, that jump to i and remain there for d successive instants,

$$a_k(i) = a_{k-d}(j)P(i|j)P_i(d)\prod_{m=k-d+1}^{k} p(\boldsymbol{x}_m|i) \tag{9.56}$$

Thus, the optimal cost associated with node (k, i) results from

$$a_k(i) = \max_{1\leq d\leq D, 1\leq j\leq K_s, j\neq i}[\delta_k(j, d, i)] \tag{9.57}$$

$$\delta_k(j, d, i) = a_{k-d}(j)P(i|j)P_i(d)\prod_{m=k-d+1}^{k} p(\boldsymbol{x}_m|i) \tag{9.58}$$

Equations (9.57) and (9.58) hold for $k > D$. For $k \leq D$, initialization of (9.57) and (9.58) requires the following computations:

$$a_1(i) = P(i)P_i(1)p(\boldsymbol{x}_1|i), \quad i = 1, \ldots, K_s$$

For $k = 2, 3, \ldots, D$,

$$a_k(i) = \max\left\{P(i)P_i(k)\prod_{m=1}^{k} p(\boldsymbol{x}_m|i), \delta_k(j, d, i)\right\}, \quad 1 \leq d < k,\ 1 \leq j \leq K_s,\ j \neq i$$

According to the previous definitions, it turns out that the optimal path is the one ending at state i, where

$$a_N(i) = \arg\max_{1\leq j\leq K_s} a_N(j)$$

Equations (9.57) and (9.58) suggest that for $k > D$ there exist $(K_s \times D - D)$ candidate arguments, $\delta_k(j, d, i)$ for the maximization of each quantity $a_k(i)$.

Reestimation Equations for the Best Path Method

The reestimation formulas for the best path method require maintaining counters to track state transitions, symbol emissions from the individual states, and state durations. For example, the state transition probability $P(i|j)$ is reestimated by counting the number of times the transition from state j to state i appears along the optimal path (computed at the current iteration) and dividing it by the total number of times transitions from state j to any other state are detected. This approach results in the following formulas:

$$\bar{P}(i|j) = \frac{\text{number of transitions from state } j \text{ to state } i}{\text{total number of transitions from state } j}, \quad i \neq j$$

$$\bar{P}_{\boldsymbol{x}}(r|i) = \frac{\text{number of times } \boldsymbol{x} \to r \text{ was emitted from state } i}{\text{total number of observations at state } i}, \quad r = 1, 2, \ldots, L$$

$$\bar{P}_i(d) = \frac{\text{number of times } d \text{ successive observations are emitted from state } i}{\text{total time spent at state } i}$$

A variant of the best path state duration HMM modeling is proposed in [Pikr 06], which has been developed to fit the needs of the music recognition/classification task. To this end, the cost function is modified to account for possible errors that are usually encountered in practice—that is, errors in fundamental frequency estimation or variations among recordings of the same music item due to different instrument players.

Segment Modeling

Although HMM modeling is one of the most powerful and widely used techniques in recognition, it is not without its shortcomings. One of its principal limitations is the required assumption of independence among the observations (conditioned on the state sequence). In fact, this is not true for most of the cases. Another limitation is the rather weak state duration modeling achieved by standard HMM modeling. To overcome these limitations, a number of schemes have been suggested. Such an example is the state duration HMM modeling. More recently, an effort was made to present a variety of such schemes in a unified framework, under the notion of *segment modeling*. Here only the basic definitions will be reviewed.

In HMM modeling on the arrival of a transition at a state, *a single* observation (corresponding to a single frame of the original speech samples) is assumed to be emitted and the fundamental observation distribution is *on the frame level*, that is, $p(\boldsymbol{x}|i)$. In contrast, in segment modeling a *segment* X_r^d consisting of d frames, $X_r^d = [\boldsymbol{x}_r, \ldots, \boldsymbol{x}_{r+d-1}]$, is assumed to be emitted upon the arrival at a state. Here d is a random variable itself. The fundamental distribution is now at the segment level, that is, $p(X_r^d|i, d)$. A schematic representation is given in Figure 9.10. The parameters describing a segment model are (a) the number of states; (b) their transition modeling parameters; (c) the joint probability density function for the segment distribution, given the duration d; and (d) the duration probability $P(d|i)$. Training of these parameters follows generalizations of the Baum–Welch and Viterbi schemes. A more detailed treatment of the topic is beyond our scope, and the interested reader may consult, for example, [Oste 96, Russ 97, Gold 99] and the references therein.

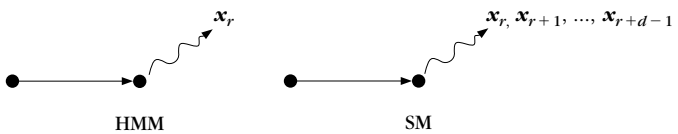

FIGURE 9.10

HMM and segment modeling (SM) for the emission of observations upon arrival of a transition at a state.

9.8 TRAINING MARKOV MODELS VIA NEURAL NETWORKS

The training phase of an HMM-based recognition system is entirely dedicated to the learning of probabilities (and densities). In Chapter 3 we have seen that a *supervised* classifier optimized via certain criteria, such as least squares, can approximate posterior class probabilities. This is the kickoff point for our current concern. States can be treated as classes, and a multilayer perceptron can be used as a nonlinear classifier. The observations feed the input nodes, and the network has as many outputs as the states. Training can be done via the backpropagation algorithm, and the desired responses will be 1 at the true state output and 0 at the others (see Chapter 4). It is now straightforward to see that the outputs of the NN will sufficiently approximate the posterior probabilities $P(i|\boldsymbol{x})$. These can then be changed to $p(\boldsymbol{x}|i)$, as required in the recognition phase of a Markov model–based recognizer, via the Bayes rule

$$p(\boldsymbol{x}|i) = \frac{P(i|\boldsymbol{x})p(\boldsymbol{x})}{P(i)}$$

where the state priors $P(i)$ are determined from their relative occurrence frequencies and $p(\boldsymbol{x})$ is constant for all states during recognition.

In following this procedure, we have made a crucial assumption. That is, the states are treated here as *being observable*, and during training we know the specific state from which each observation originates. For example, in speech recognition this is possible by associating each phoneme in a spoken word with a state. Here lies a disadvantage of this approach, since accurate segmentation of the speech signal is required and the boundaries are not always well defined. This is not the case in the HMM approach, where it is left to the algorithm to decide optimally about the state boundaries. A scheme for unified training of HMM/MLP that avoids the segmentation problem has been suggested [Koni 96]. Let us now turn to the benefits of bringing neural networks into the scene.

We have already mentioned that a major disadvantage of the standard HMM is the assumption of independence among the observations. Using a multilayer perceptron, the underlying statistical dependence can easily be accommodated. Figure 9.11 shows a possible way. Together with the "current" observation vector $\boldsymbol{x}_k$, p "past" as well as p "future" ones appear simultaneously at the input nodes. Thus, the input nodes amount to $(2p + 1)l$, with l being the dimension of the observation vectors. During training, the desired response will be 1 at the output node, corresponding to the state that "gives birth" to the "current" vector. We say that the network is trained with *contextual input information*. Further data dependence can also be accommodated by providing the input with information about the previous state in the sequence. During recognition, this is provided via an output feedback, shown in the figure by the dotted lines [Bourl 90]. Obviously, in such a configuration the output nodes of the network compute the conditional state probabilities $P(i_k|X_{k-p}^{k+p}, i_{k-1})$, where X_{k-p}^{k+p} denotes the contextual input information ranging from $\boldsymbol{x}_{k-p}$ to $\boldsymbol{x}_{k+p}$.

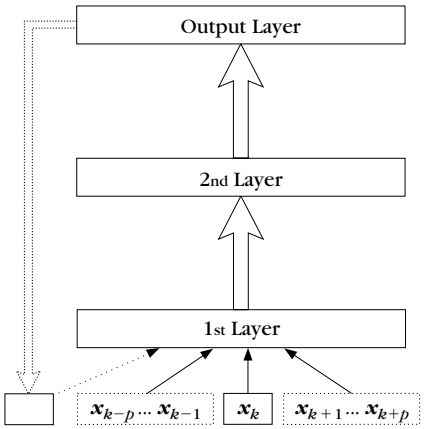

FIGURE 9.11

A multilayer perceptron architecture for training the parameters of a Markov model.

Having these probabilities at our disposal, a number of new "opportunities" open to us. Let us, for example, return to our original goal in (9.1) and treat states as classes. By the chain rule we have

$$\begin{aligned} P(\Omega_i|X) &\equiv P(i_1, i_2, \ldots, i_N|X) \\ &= P(i_N|i_{N-1}, \ldots, i_1, X) \ldots P(i_{N-1}|i_{N-2}, \ldots, i_1, X) P(i_1|X) \end{aligned} \tag{9.59}$$

Taking into account the Markovian property of the state dependence and relaxing a bit the conditional constraint on the observations, this can be written as

$$P(\Omega_i|X) = \prod_k P(i_k|X_{k-p}^{k+p}, i_{k-1}) \tag{9.60}$$

with some appropriate initial conditions. Computing its maximum can easily be done via dynamic programming arguments. A number of other alternatives are also possible; see for example [Bourl 90, Bourl 94, Morg 95] for a more detailed discussion of the topic. In [Pikr 06a] the use of Bayesian networks is suggested in place of neural networks.

Finally, it must be emphasized that in order to obtain good probability estimates the size of the multilayer perceptron must be large enough to have good approximating capabilities. This, of course, requires increased computational resources for the training. Furthermore, the incorporation of the contextual information imposes its own demands on large networks. Another point is that the approximation of probabilities by the network is valid at the global minimum of the minimized cost function, at least in theory. Practical issues affecting the overall performance of such an approach are reported in [Spec 94].

9.9 A DISCUSSION OF MARKOV RANDOM FIELDS

So far, our concern with context-dependent classification has been limited to the one-dimensional case. The current subsection is focused on the related two-dimensional generalizations. That is, observations will be treated as two-dimensional sequences $X(i,j)$. Such problems result in image processing, and observations can be, for example, the gray levels of the image array pixels. No doubt, complications arise, and our aim here is to provide the basic definitions and directions and not a detailed treatment of the topic.

Let us assume that we are given an array of observations X:$X(i,j)$, $i = 0, 1, \ldots, N_x - 1, j = 0, 1, \ldots, N_y - 1$, and a corresponding array of classes/states Ω: ω_{ij}, where each ω_{ij} can take one of M values. Once more our objective is, given the array of the observations, to estimate the corresponding values of the state array Ω so that

$$p(X|\Omega)P(\Omega) \text{ is maximum} \tag{9.61}$$

Within the scope of context-dependent classification the values of the elements of Ω will be assumed to be mutually dependent. Furthermore, we will assume that the range of this dependence is limited within a neighborhood. This brings us to the notion of Markov random fields (MRFs) defined in Chapter 7. Thus, for each (i,j) element of the array Ω a respective *neighborhood* $\mathcal{N}_{ij}$ is defined so that

- $\omega_{ij} \notin \mathcal{N}_{ij}$
- $\omega_{ij} \in \mathcal{N}_{kl} \Longleftrightarrow \omega_{kl} \in \mathcal{N}_{ij}$

In words, the (i,j) element does not belong to its own set of neighbors, and if ω_{ij} is a neighbor of ω_{kl}, then ω_{kl} is also a neighbor of ω_{ij}. The Markov property is then defined as

$$P(\omega_{ij}|\bar{\Omega}_{ij}) = P(\omega_{ij}|\mathcal{N}_{ij}) \tag{9.62}$$

where $\bar{\Omega}_{ij}$ includes all the elements of Ω except the (i,j) one. Figure 9.12 gives a typical example of a neighborhood with eight neighbor pixels. Equation (9.62) is a generalization of (9.3). In the one-dimensional case the ordering of the sequence led to the relation (9.4). Unfortunately, this *sequence ordering does not generalize in a natural way* to the two-dimensional case and imposes limitations on the involvement of the computationally elegant dynamic programming techniques [Hans 82].

A seminal paper that had an impact on the use of MRF modeling in image processing and analysis was that of Geman and Geman [Gema 84]. They built upon the important Hammersley–Clifford theorem, which establishes the *equivalence* between Markov random fields and Gibbs distributions [Besa 74]. Thus, we can talk of *Gibbs random fields* (GRFs). A Gibbs conditional probability is of the form

$$P(\omega_{ij}|\mathcal{N}_{ij}) = \frac{1}{Z}\exp\left(-\frac{1}{T}\sum_{k} F_k(\mathcal{C}_k(i,j))\right) \tag{9.63}$$

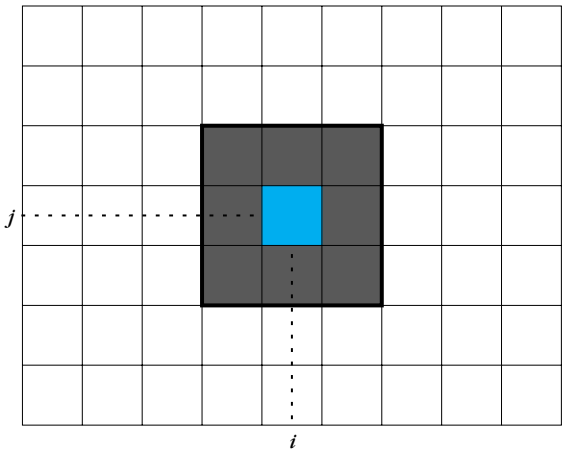

FIGURE 9.12

Example of a neighborhood involving eight neighbors of the (i,j) element (red).

where Z is a normalizing constant so that probabilities sum up to 1, T is a parameter, and $F_k(\cdot)$ are functions of the states of the pixels in the *cliques* $\mathcal{C}_k(i,j)$. A clique consists of either a single pixel or a set of pixels, which are neighbors of each other, with respect to the type of the chosen neighborhood. Figure 9.13 shows two cases of neighborhoods and the corresponding sets of cliques. A typical example of the exponent function in (9.63) for the four neighbors case is

$$-\frac{1}{T}\omega_{ij}\left(\alpha_1 + \alpha_2(\omega_{i-1,j} + \omega_{i+1,j}) + \alpha_2(\omega_{i,j-1} + \omega_{i,j+1})\right)$$

where the a_i's are constants.

It turns out that the joint probability $P(\Omega)$ for the Gibbsian model is

$$P(\Omega) = \exp\left(-\frac{U(\Omega)}{T}\right) \tag{9.64}$$

where

$$U(\Omega) = \sum_{ij}\sum_{k} F_k(\mathcal{C}_k(i,j)) \tag{9.65}$$

that is, the sum of the functions over all possible cliques associated with the neighborhood. In many cases, the posterior probability $P(\Omega|X)$, which is to be maximized (i.e., (9.61)) also turns out to be Gibbsian. Such cases result, for example, if the regions in the image are themselves generated by Markov (e.g., Gaussian two-dimensional AR) processes [Deri 86, Chel 85]. *Simulating annealing* techniques can then be employed to obtain the required maximum [Gema 84].

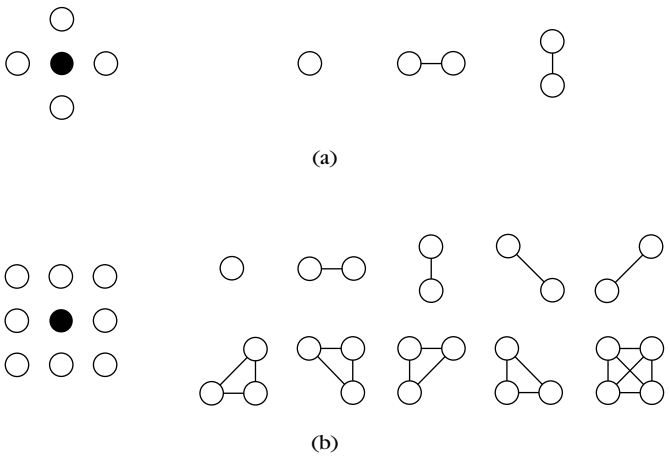

FIGURE 9.13

Two examples of neighborhoods with the corresponding cliques.

Hidden Markov generalizations to the two-dimensional plane have also been considered [Povl 95]. The idea here is to build a *pseudolikelihood function* starting from the local state transition probabilities, using Besag's method for coding the image in mutually independent pixel sets [Besa 74]. An alternative EM formulation of the problem was given in [Zhan 94]. Finally, another direction is the combination of Markov random fields and multiresolution analysis. At the subsampling stage, the Markov property is lost and suitable models are derived for the coarser resolutions. For more details the reader may consult, for example, [Laks 93, Bell 94, Kris 97] and the references therein.

9.10 PROBLEMS

9.1 Assume an HMM model with K states and an observation string X of N continuous observations. Assume that the pdf in each state is described by a Gaussian with known diagonal covariance matrix and unknown mean values. Using the EM algorithm, derive the reestimation recursions.
Hint: Form the complete data set as $Y = (X, \Omega)$, where Ω is the set of the states.

9.2 If the self-transition probability of a state is $P(i|i)$, then the probability of the model being at state i for d successive stages is given by $P_i(d) = (P(i|i))^{d-1}(1 - P(i|i))$. Show that the average duration for staying in state i is equal to $\bar{d} = \frac{1}{1-P(i|i)}$.

9.3 In practice, a number Q of different versions of the spoken word are used for training, each resulting in a sequence of observations X_m of length N_m,

$m = 1, 2, \ldots, Q$. Then comment on the following reestimation formulas:

$$\bar{P}(j|i) = \frac{\sum_{m=1}^{Q} \frac{1}{P(X_m|\mathcal{S})} \sum_{k=1}^{N_m-1} \xi_k(i,j,X_m|S)}{\sum_{m=1}^{Q} \frac{1}{P(X_m|\mathcal{S})} \sum_{k=1}^{N_m-1} \alpha^m(i_k = i)\beta^m(i_k = i)}$$

$$\bar{P}_{\boldsymbol{x}}(r|i) = \frac{\sum_{m=1}^{Q} \frac{1}{P(X_m|\mathcal{S})} \sum_{(k=1 \text{ and } \boldsymbol{x}\to r)}^{N_m} \alpha^m(i_k = i)\beta^m(i_k = i)}{\sum_{m=1}^{Q} \frac{1}{P(X_m|\mathcal{S})} \sum_{k=1}^{N_m} \alpha^m(i_k = i)\beta^m(i_k = i)}$$

where superscript m refers to the mth observation sequence, and $\xi(i,j,X|\mathcal{S})$ is defined in (9.30).

9.4 Rederive recursions (9.33) and (9.34) in terms of the scaled versions of α, β

$$\hat{\alpha}(i_k = i) = \frac{1}{c_k}\alpha(i_k = i), \quad \hat{\beta}(i_{k+1} = i) = c_k\beta(i_{k+1} = i)$$

where $c_k = \sum_{i_k=1}^{K_s} \alpha(i_k)$.

9.5 Assume that the HMM models are not equiprobable and let Λ be the set of all the unknown parameters for the M available models. Assume now that a training string X is known to correspond to the model $\mathcal{S}_r$. However, during training, maximization of $P(\mathcal{S}_r|X, \Lambda)$ is done with respect to all the parameters and not only those of the specific model. Show that this optimization leads to a maximum ratio between the contribution $p(X|\mathcal{S}_r, \Lambda)P(\mathcal{S}_r)$ of the correct model and $\sum_{s\neq r} p(X|\mathcal{S}_s, \Lambda)P(\mathcal{S}_s)$ of the incorrect models. That is, optimization with respect to all the parameters offers maximum discrimination power.

MATLAB PROGRAMS AND EXERCISES

Computer Programs

9.1 *Recognition score for HMMs using the Baum-Welch method.* Write a MATLAB function named *Baum_Welch_Do_HMM* that takes as input: (a) a column vector of initial state probabilities *pi_init*, (b) the transition matrix A, whose (i,j) element is the probability of transition from state i to state j, (c) the matrix of the emission probabilities B, whose (i,j) element is the probability to emit the ith alphabet symbol from state j, and (d) a row vector O, which contains a sequence of the code numbers of discrete symbols. It returns the score produced when the HMM, defined by *pi_init*, A, B, is applied to the sequence of symbols contained in O. Assume that if the alphabet symbols are, say, $s_1, s_2, \ldots, s_q$, the corresponding code numbers are $1, 2, \ldots, q$.

Solution

In order to avoid underflow problems, in the following implementation the score is computed as the log product of scaling factors.

```
function matching_prob=Baum_Welch_Do_HMM(pi_init,A,B,O)
  %Initialization
  T=length(O);
  [M,N]=size(B);
  alpha(:,1)=pi_init .* B(O(1),:)';
  c(1)=1/(sum(alpha(:,1)));
  alpha(:,1)=c(1)*alpha(:,1);
  for t=2:T
    for i=1:N
      alpha(i,t)=sum((alpha(:,t-1).* A(:,i)) * B(O(t),i));
    end
    c(t)=1/(sum(alpha(:,t)));
    alpha(:,t)=c(t)*alpha(:,t);
  end
  matching_prob=-sum(log10(c));
```

9.2 ***Viterbi method for Discrete Observation HMMs.*** Write a MATLAB function named *Viterbi_Do_HMM* that takes the same inputs as *Baum_Welch_Do_HMM* and returns (a) the best-state sequence and (b) the respective probability produced when the HMM, defined by *pi_init*, *A* and *B*, is applied to *O*, using the Viterbi method.

Solution

In the following implementation, the trellis diagram is constructed first, and then the best state sequence is extracted using the *back_traking* function defined in the computer programs' section of Chapter 8.

```
function [matching_prob,best_path]=Viterbi_Do_HMM...
  (pi_init,A,B,O)
  %Initialization
  T=length(O);
  [M,N]=size(B);
  pi_init(find(pi_init==0))=-inf;
  pi_init(find(pi_init>0))=log10(pi_init(find(pi_init>0)));
  A(find(A==0))=-inf;
  A(find(A>0))=log10(A(find(A>0)));
  B(find(B==0))=-inf;
  B(find(B>0))=log10(B(find(B>0)));
  % First observation
  alpha(:,1)=pi_init + B(O(1),:)';
  pred(:,1)=zeros(N,1);
  % Construct the trellis diagram
  for t=2:T
```

```
    for i=1:N
      temp=alpha(:,t-1)+A(:,i)+B(O(t),i);
      [alpha(i,t),ind]=max(temp);
      pred(i,t)=ind+sqrt(-1)*(t-1);
    end
  end
  [matching_prob,winner_ind]=max(alpha(:,T));
  best_path=back_tracking(pred,winner_ind,T);
```

9.3 *Viterbi method for Continuous Observation HMMs.* Under the hypothesis that the emission pdfs $p(\boldsymbol{x}|i)$ are Gaussians, write a MATLAB function named *Viterbi_Co_HMM*, which takes as inputs: (a) a column vector of initial state probabilities *pi_init*, (b) the transition matrix A, (c) a row vector, m, whose ith element is the mean of the ith Gaussian emission pdf, (d) a row vector, *sigma*, containing the variances of the previous pdfs, and (e) a row vector O, which contains a feature sequence. It returns, (a) the best-state sequence and (b) the respective probability, produced when the HMM, defined by *pi_init*, A, m, *sigma*, is applied to O, using the Viterbi method.

Solution

```
function [matching_prob,best_path]=Viterbi_Co_HMM...
  (pi_init,A,m,sigma,O)
  %Initialization
  T=length(O);
  [N,N]=size(A);
  pi_init(find(pi_init==0))=-inf;
  pi_init(find(pi_init>0))=log10(pi_init(find(pi_init>0)));
  A(find(A==0))=-inf;
  A(find(A>0))=log10(A(find(A>0)));
  for i=1:N
  alpha(i,1)=pi_init(i)+log10(normpdf(O(1),m(i),sigma(1)));
  end
  pred(:,1)=zeros(N,1);
  % Construction of the trellis diagram
  for t=2:T
    for i=1:N
      temp=alpha(:,t-1)+A(:,i)+log10(normpdf(O(t),m(i),...
       sigma(i)));
      [alpha(i,t),ind]=max(temp);
      pred(i,t)=ind+sqrt(-1)*(t-1);
    end
  end
  [matching_prob,winner_ind]=max(alpha(:,T));
  best_path=back_tracking(pred,winner_ind,T);
```

Computer Experiments

9.1 Two coins are used for a coin-tossing experiment, that is, coin A and coin B. The probability that coin A returns heads is 0.6, and the respective probability for coin B is 0.4. An individual standing behind a curtain decides which coin to toss as follows: the first coin to be tossed is always coin A, the probability that coin A is re-tossed is 0.4, and similarly, the probability that coin B is re-tossed is 0.6. An observer can only have access to the outcome of the experiment, that is, the sequence of heads and tails that is produced. (a) Model the experiment by means of a HMM (i.e., define the vector of the initial state probabilities, the transition matrix and the matrix of the emission probabilities) and (b) use the *Baum_Welch_Do_HMM* function to compute the HMM score for the sequence of observations $\{H, H, T, H, T, T\}$ where H stands for heads and T stands for tails.
Hint: In defining the input sequence of symbols O for *Baum_Welch_Do_HMM* function, use "1" for "H" and "2" for "T".

9.2 For the HMM of the previous experiment, use the *Viterbi_Do_HMM* function to find the best state sequence and respective path probability, for the following observation sequences: $\{H, T, T, T, H\}$ and $\{T, T, T, H, H, H, H\}$.
Hint: In defining the input sequence of symbols O for *Viterbi_Do_HMM* function, use "1" for "H" and "2" for "T".

9.3 Assume that two number generators, Gaussian in nature, operate with mean values 0 and 5, respectively. The values for the respective standard deviations are 1 and 2. The following experiment is carried out behind a curtain: a person tosses a coin to decide which generator will be the first to emit a number. Heads has a probability of 0.4 and stands for generator A. Then the coin is tossed 8 times, and each time the coin decides which generator will emit the next number. An observer has only access to the outcome of the experiment, i.e., to the following sequence of numbers: $\{0.3, 0.4, 0.2, 2.1, 3.2, 5, 5.1, 5.2, 4.9\}$. (a) Model the experiment by means of a HMM that emits continuous observations and (b) use the *Viterbi_Co_HMM* function to compute the best-state sequence and the corresponding probability for the given sequence of numbers.

REFERENCES

[Agaz 93] Agazi O.E., Kuo S.S. "Hidden Markov model based optical character recognition in the presence of deterministic transformations," *Pattern Recognition*, Vol. 26, pp. 1813–1826, 1993.

[Anto 97] Anton-Haro C., Fonollosa J.A.R., Fonollosa J.R. "Blind channel estimation and data detection using HMM," *IEEE Transactions on Signal Processing*, Vol. 45(1), pp. 241–247, 1997.

[Aric 02] Arica N., Yarman-Vural F.T. "Optical character recognition for cursive handwriting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24(6), pp. 801–813, 2003

[Arsl 99] Arslan L., Hansen J.H.L. "Selective training for hidden Markov models with applications to speech classification," *IEEE Transactions on Speech and Audio Processing*, Vol. 7(1), pp. 46–54, 1999.

[Bahl 86] Bahl L.R., Brown B.F., Desouza P.V. "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, Vol. 1, pp. 872–875, Japan, 1986.

[Bake 75] Baker J. "The DRAGON system—an overview," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. 23(1), pp. 24–29, 1975.

[Baum 67] Baum L.E., Eagon J.A. "An inequality with applications to statistical prediction for functions of Markov processes and to a model for ecology," *Bulletin of the American Mathematical Society*, Vol. 73, pp. 360–362, 1967.

[Baum 70] Baum L.E., Petrie T., Soules G., Weiss N. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Annals of Mathematical Statistics*, Vol. 41, pp. 164–171, 1970.

[Baum 68] Baum L.E., Sell G.R. "Growth functions for transformations of manifolds," *Pacific Journal of Mathematics*, Vol. 27, pp. 211–227, 1968.

[Bell 90] Bellegarda J.R., Nahamoo D. "Tied mixture continuous parameter modeling for speech recognition," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. 38(12), pp. 2033–2045, 1990.

[Bell 94] Bello M.G. "A combined Markov random field and wave-packet approach to image segmentation," *IEEE Transactions on Image Processing*, Vol. 3(6), pp. 834–847, 1994.

[Besa 74] Besag J. "Spatial interaction and the statistical analysis of lattice systems," *J. Royal Stat. Soc. B*, Vol. 36(2), pp. 192–236, 1974.

[Bourl 94] Bourland H., Morgan N. *Connectionist Speech Recognition*. Kluwer Academic Publishers, 1994.

[Bourl 90] Bourland H., Wellekens C.J. "Links between Markov models and the multilayer perceptrons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12(12), pp. 1167–1178, 1990.

[Chel 85] Chellapa R., Kashyap R.L. "Texture synthesis using 2-D noncausal autoregressive models," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. 33(1), pp. 194–203, 1985.

[Chen 95a] Chen J.-L., Kundu A. "Unsupervised texture segmentation using multichannel decomposition and hidden Markov models," *IEEE Transactions on Image Processing*, Vol. 4(5), pp. 603–620, 1995.

[Chen 95] Chen M.Y., Kundu A., Srihari S.N. "Variable duration HMM and morphological segmentation for handwritten word recognition," *IEEE Transactions on Image Processing*, Vol. 4(12), pp. 1675–1689, 1995.

[Chie 03] Chien J.-T., Huang C.-H. "Bayesian learning of speech duration models," *IEEE Transactions on Speech and Audio Processing*, Vol. 11(6), pp. 558–567, 2003.

[Dell 93] Deller J., Proakis J., Hansen J. *Discrete Time Processing of Speech Signals*. Macmillan, 1993.

[Deng 94] Deng L.,Aksmanovic M. "Speaker-independent phonetic classification using HMM with mixtures of trend functions," *IEEE Transactions on Speech and Audio Processing*, Vol. 5(4), pp. 319–324, 1997.

[Deri 86] Derin H. "Segmentation of textured images using Gibb's random fields," *Computer Vision, Graphics, and Image Processing*, Vol. 35, pp. 72–98, 1986.

[Diga 99] Digalakis V. "Online adaptation of hidden Markov models using incremental estimation algorithms," *IEEE Transactions on Speech and Audio Processing*, Vol. 7(3), pp. 253–261, 1999.

[Diga 95] Digalakis V., Rtischef D., Neumeyer L.G. "Speaker adaptation using constrained estimation of Gaussian mixtures," *IEEE Transaction on Speech and Audio Processing*, Vol. 3(5), pp. 357–366, 1995.

[Diga 96] Digalakis V., Monaco P., Murveit H. "Genones: Generalized mixture tying in continuous HMM model-based speech recognizers," *IEEE Transactions on Speech and Audio Processing*, Vol. 4(4), pp. 281–289, 1996.

[Diga 93] Digalakis V., Rohlicek J.R., Ostendorf M. "ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition," *IEEE Transactions on Speech and Audio Processing*, Vol. 1(4), pp. 431–441, 1993.

[ElYa 99] El-Yacoubi A., Gilloux M., Sabourin R., Suen C.Y. "An HHM-based approach for off-line unconstrained handwritten word modeling and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21(8), pp. 752–760, 1999.

[Ephr 89] Ephraim Y., Dembo A., Rabiner L.R. "A minimum discrimination information approach to hidden Markov modelling," *IEEE Transactions on Information Theory*, Vol. 35, pp. 1001–1023, September 1989.

[Ferg 80] Ferguson J. D. "Hiden Markov analysis: An introduction," in *Hidden Markov Models for Speech*, Institute for Defence Analysis, Princeton university, 1980.

[Fu 82] Fu K.S. *Syntactic Pattern Recognition and Applications*, Prentice Hall, 1982.

[Gale 99] Gales M.J.F. "Semitied covariance matrices for hidden Markov models," *IEEE Transactions on Speech and Audio Processing*, Vol. 7(3), pp. 272–281, 1999.

[Gauv 94] Gauvain J.L., Lee C.H. "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Transactions on Speech and Audio Processing*, Vol. 2(2), pp. 291–299, 1994.

[Gema 84] Geman S., Geman D. "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6(6), pp. 721–741, 1984.

[Geor 97] Georgoulakis C.,Theodoridis S. "Efficient clustering techniques for channel equalization in hostile environments," *Signal Processing*, Vol. 58, pp. 153–164, 1997.

[Geor 98] Georgoulakis C., Theodoridis S. "Blind equalization for nonlinear channels via hidden Markov modeling," *Proceedings EUSIPCO-98*, Rhodes, Greece, 1998.

[Gold 99] Goldberger J., Burshtein D., Franco H. "Segmental modeling using a continuous mixture of noparametric models," *IEEE Transactions on Speech and Audio Processing*, Vol. 7(3), pp. 262–271, 1999.

[Gu 02] Gu L., Rose K. "Substate tying with combined parameter training and reduction in tied-mixture HMM design," *IEEE Transactions on Speech and Audio Processing*, Vol. 10(3), 2002.

[Hans 82] Hansen F.R., Elliot H. "Image segmentation using simple Markov field models," *Computer Graphics and Image Processing*, Vol. 20, pp. 101–132, 1982.

[He 08] He X., Deng L., Chou W. "Discriminative Learning in Sequential Pattern Recognition—A Unifying Review for Optimization-Oriented Speech Recognition," to appear *IEEE Signal Processing Magazine*, september 2008.

[Huo 95] Huo Q., Chan C., Lee C.H. "Bayesian adaptive learning of the parameters of hidden Markov model for speech recognition," *IEEE Transactions on Speech and Audio Processing*, Vol. 3(5), pp. 334–345, 1995.

[Huo 97] Huo Q., Lee C.H. "On-line adaptive learning of the continuous density HMM based on approximate recursive Bayes estimate," *IEEE Transactions on Speech and Audio Processing*, Vol. 5(2), pp. 161–173, 1997.

[Jeli 76] Jelinek F. "Continuous speech recognition by statistical methods," *Proceedings of the IEEE*, Vol. 64(4), pp. 532–555, 1976.

[Juan 85] Juang B.H. "Maximum likelihood estimation for mixture multivariate stochastic observations of Markov chains," *AT&T System Technical Journal*, Vol. 64, pp. 1235–1249, July–August 1985.

[Juan 97] Juang B.H., Chou W., Lee C.H. "Minimum classification error rate methods for speech recognition," *IEEE Transactions on Speech and Audio Processing*, Vol. 5(3), pp. 257–266, 1997.

[Juan 92] Juang B.H., Katagiri S. "Discriminative learning for minimum error classification," *IEEE Transactions on Signal Processing*, Vol. 40(12), pp. 3043–3054, 1992.

[Juan 86] Juang B.H., Levinson S.E., Sondhi M.M. "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Transactions on Information Theory*, Vol. IT-32, pp. 307–309, March 1986.

[Kale 94] Kaleh G.K., Vallet R. "Joint parameter estimation and symbol detection for linear and nonlinear unknown channels," *IEEE Transactions on Communications*, Vol. 42(7), pp. 2406–2414, 1994.

[Kim 95] Kim N.S., Un C.K. "On estimating robust probability distribution in HMM-based speech recognition," *IEEE Transactions on Speech and Audio Processing*, Vol. 3(4), pp. 279–286, 1995.

[Koni 96] Konig Y. "REMAP: Recursive estimation and maximization of a-posteriori probabilities in transition-based speech recognition," Ph.D. thesis, University of California at Berkeley, 1996.

[Kops 03] Kopsinis Y., Theodoridis S. "An efficient low-complexity technique for MLSE equalizers for linear and nonlinear channels," *IEEE Transactions on Signal Processing*, Vol. 51(12), pp. 3236–3249, 2003.

[Kris 97] Krishnamachari S., Chellappa R. "Multiresolution Gauss–Markov random field models for texture segmentation," *IEEE Transactions on Image Processing*, Vol. 6(2), pp. 251–268, 1997.

[Laks 93] Lakshmanan S., Derin H. "Gaussian Markov random fields at multiple resolutions," in *Markov Random Fields: Theory and Applications* (R. Chellappa, ed.), Academic Press, 1993.

[Lee 72] Lee C.H., Fu K.S. "A stochastic syntax analysis procedure and its application to pattern recognition," *IEEE Transactions on Computers*, Vol. 21, pp. 660–666, 1972.

[Lee 91] Lee C.H., Lin C.H., Juang B.H. "A study on speaker adaptation of the parameters of continuous density hidden Markov models," *IEEE Transactions on Signal Processing*, Vol. 39(4), pp. 806–815, 1991.

[Legg 95] Leggetter C.J., Woodland P.C. "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Comput. Speech Lang.*, Vol. 9, pp. 171–185, 1995.

[Levi 86] Levinson S.E. "Continuously variable duration HMMs for automatic speech recognition," *Computer Speech and Language*, Vol. 1, pp. 29–45, March 1986.

[Levi 83] Levinson S.E., Rabiner L.R., Sondhi M.M. "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell System Technical Journal*, Vol. 62(4), pp. 1035–1074, April 1983.

[Li 04] Li J., Wang J., Zhao Y., Yang Z. "Self adaptive design of hidden Markov models," *Pattern Recognition Letters*, Vol. 25, pp. 197–210, 2004.

[Li 00] Li X., Parizeau M., Plamondon R. "Training hidden Markov models with multiple observations-A combinatorial method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22(4), pp. 371–377, 2000.

[Lipo 82] Liporace L.A. "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Transactions on Information Theory*, Vol. IT-28(5), pp. 729–734, 1982.

[Moon 96] Moon T. "The expectation maximization algorithm," *Signal Processing Magazine*, Vol. 13(6), pp. 47–60, 1996.

[Morg 95] Morgan N. Boulard H. "Continuous speech recognition," *Signal Processing Magazine*, Vol. 12(3), pp. 25–42, 1995.

[Neap 04] Neapolitan R.D. *Learning Bayesian Networks*, Prentice Hall, Gliffs, N.J. 2004.

[Oste 96] Ostendorf M., Digalakis V., Kimball O. "From HMM's to segment models: A unified view of stochastic modeling for speech," *IEEE Transactions on Audio and Speech Processing*, Vol. 4(5), pp. 360–378, 1996.

[Papo 91] Papoulis A. *Probability Random Variables and Stochastic Processes*, 3rd ed., McGraw-Hill 1991.

[Pikr 06] Pikrakis A., Theodoridis S., Kamarotos D. "Classification of musical patterns using variable duration hidden Markov models," *IEEE Transactions on Speech and Audio Processing*, Vol. 14(5), pp. 1795–1807, 2006.

[Pikr06a] Pikrakis A., Gaunakopoulos T., Theodoridis S. "Speech/music discrimination for radio broadcasts using a hybrid HMM–Bayesiay network architecture," *Proceedings, EUSIPCO-Florence*, 2006.

[Pori 82] Poritz A.B. "Linear predictive HMM and the speech signal," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 1291–1294, Paris, 1982.

[Povl 95] Povlow B., Dunn S. "Texture classification using noncausal hidden Markov models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17(10), pp. 1010–1014, 1995.

[Proa 89] Proakis J. *Digital Communications*, 2nd ed., McGraw-Hill, 1989.

[Rabi 89] Rabiner L. "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of IEEE*, Vol. 77, pp. 257–285, February, 1989.

[Rabi 93] Rabiner L., Juang B.H. *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[Ramd 03] Ramdane S., Taconet B., Zahour A. "Classification of forms with handwritten fields by planar Markov models," *Pattern Recognition*, Vol. 36, pp. 1045–1060, 2003.

[Rao 01] Rao A.V., Rose K. "Deterministically annealed design of hidden Markov Model speech recognizers," *IEEE Transactions on Speech and Audio Precessing*, Vol. 9(2), pp. 111–127, 2001.

[Russ 97] Russell M., Holmes W. "Linear trajectory segmental HMM's," *IEEE Signal Processing Letters*, Vol. 4(3), pp. 72–75, 1997.

[Russ 85] Russell M.J., Moore R.K. "Explicit modeling of state occupancy in HMMs for automatic speech recognition," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 5–8, 1985.

[Spec 94] Special issue on neural networks for speech in *IEEE Transactions on Speech and Audio Processing*, Vol. 2(1), 1994.

[Theo 95] Theodoridis S., Cowan C.F.N., See C.M.S. "Schemes for equalization of communication channels with nonlinear impairments," *IEE Proceedings on Communications*, Vol. 142(3), pp. 165–171, 1995.

[Turi 98] Turin W. "Unidirectional and parallel Baum-Welch algorithms," *IEEE Transactions on Speech and Audio Processing*, Vol. 6(6), pp. 516–523, 1998.

[Vite 67] Viterbi A.J. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, Vol. 13, pp. 260–269, 1967.

[Vlon 92] Vlontzos J.A., Kung S.Y. "Hidden Markov models for character recognition," *IEEE Transactions on Image Processing*, Vol. 1(4), pp. 539–543, 1992.

[Wang 01] Wang S., Zhao Y. "Online Bayesian tree-structured transformation of HMM's with optimal model selection for speaker adaptation," *IEEE Transactions on Speech and Audio Processing*, Vol. 9(6), pp. 663–677, 2001.

[Wu 96] Wu W.R., Wei S.C. "Rotational and gray scale transform invariant texture classification using spiral resampling, subband decomposition, and hidden Markov model," *IEEE Transactions on Image Processing*, Vol. 5(10), pp. 1423–1435, 1996.

[Zhan 94] Zhang J., Modestino J.W., Langan D.A. "Maximum likelihood parameter estimation for unsupervised stochastic model based image segmentation," *IEEE Transactions on Image Processing*, Vol. 3(4), pp. 404–421, 1994.