defining the degree of fit between a word and some speech data, as an alternative to the 'cumulative distance' used in Chapter 8. This measure of degree of fit is based on the notion of **probability**, and the basic theory is explained in this chapter. For simplicity in introducing the concepts, the discussion in this chapter will continue to concentrate on words as the recognition unit. In practice, the majority of current recognition systems represent words as a sequence of **sub-word units**, but the underlying theory is not affected by the choice of unit. The use of sub-word units for recognition, together with other developments and elaborations of the basic statistical method will be explained in later chapters. In the following explanation, some elementary knowledge of statistics and probability theory is assumed, but only at a level which could easily be obtained by referring to a good introductory textbook (see Chapter 17 for some references).

9.2 INTRODUCTION TO HIDDEN MARKOV MODELS

Up to now we have considered choosing the best matching word by finding the template which gives the minimum cumulative 'distance' along the optimum matching path. An alternative approach is, for each possible word, to postulate some device, or **model**, which can generate patterns of features to represent the word. Every time the model for a particular word is activated, it will produce a set of feature vectors that represents an example of the word, and if the model is a good one, the statistics of a very large number of such sets of feature vectors will be similar to the statistics measured for human utterances of the word. The best matching word in a recognition task can be defined as the one whose model is most likely to produce the observed sequence of feature vectors. What we have to calculate for each word is thus not a 'distance' from a template, but the *a posteriori* probability that its model could have produced the observed set of feature vectors. We do not actually have to make the model produce the feature vectors, but we use the known properties of each model for the probability calculations. We will assume for the moment that the words are spoken in an 'isolated' manner, so that we know where the start and end of each word are, and the task is simply to identify the word. Extensions to sequences of words will be considered in Section 9.11.

We wish to calculate the *a posteriori* probability, $P(w|\boldsymbol{Y})$, of a particular word, w, having been uttered during the generation of a set of feature observations, $\boldsymbol{Y}$. We can use the model for w to calculate $P(\boldsymbol{Y}|w)$, which is the probability of $\boldsymbol{Y}$ conditioned on word w (sometimes referred to as the **likelihood** of w). To obtain $P(w|\boldsymbol{Y})$, however, we must also include the *a priori* probability of word w having been spoken. The relationship between these probabilities is given by Bayes' rule:

$$P(w|\boldsymbol{Y}) = \frac{P(\boldsymbol{Y}|w)P(w)}{P(\boldsymbol{Y})}. \tag{9.1}$$

This equation states that the probability of the word given the observations is equal to the probability of the observations given the word, multiplied by the probability of the word (irrespective of the observations), and divided by the probability of the observations. The probability, $P(\boldsymbol{Y})$, of a particular set of feature observations, $\boldsymbol{Y}$, does not depend on which word is being considered as a possible match, and therefore only acts as a scaling factor on the probabilities. Hence, if the

Ex.1019 / Page 31 of 102
AMAZON.COM, INC.

goal is to find the word w which maximizes $P(w|\boldsymbol{Y})$, the $P(\boldsymbol{Y})$ term can be ignored, because it does not affect the choice of word. If for the particular application all permitted words are equally likely, then the $P(w)$ term can also be ignored, so we merely have to choose the word model that maximizes the probability, $P(\boldsymbol{Y}|w)$, of producing the observed feature set, $\boldsymbol{Y}$. In practice for all but the simplest speech recognizers the probability of any particular word occurring will depend on many factors, and for large vocabularies it will depend on the statistics of word occurrence in the language. This aspect will be ignored in the current chapter, but will be considered further in Chapter 12.

The way we have already represented words as sequences of template frames gives us a starting point for the form of a possible model. Let the model for any word be capable of being in one of a sequence of **states**, each of which can be associated with one or more frames of the input. In general the model moves from one state to another at regular intervals of time equal to the frame interval of the acoustic analysis. However, we know that words can vary in timescale. In the asymmetrical DP algorithm mentioned in Chapter 8 (Figure 8.4(b), showing slopes of 0, 1 and 2) the timescale variability is achieved by repeating or skipping frames of the template. In our model this possibility can be represented in the sequence of states by allowing the model to stay in the same state for successive frame times, or to bypass the next state in the sequence. The form of this simple model is shown in Figure 9.1. In fact, if a word template has a sequence of very similar frames, such as might occur in a long vowel, it is permissible to reduce the number of states in the model by allowing it to stay in the same state for several successive frames.

The mathematics associated with a model such as the one shown in Figure 9.1 can be made more tractable by making certain simplifying assumptions. To be more specific, it is assumed that the output of the model is a **stochastic process** (i.e. its operation is governed completely by a set of probabilities), and that the probabilities of all its alternative actions at any time t depend only on the state it is in at that time, and not on the value of t. The current output of the model therefore depends on the identity of the current state, but is otherwise independent of the sequence of previous states that it has passed through to reach that state. Hence the model's operation is a **first-order Markov process**, and the sequence of states is a **first-order Markov chain**. Although the model structure shown in Figure 9.1 is quite appropriate for describing words that vary in timescale, the equations that represent the model's behaviour have exactly the same form in the more general case where transitions are allowed between all possible pairs of states.

At every frame time the model is able to change state, and will do so randomly in a way determined by a set of **transition probabilities** associated with the state it is currently in. By definition, the probabilities of all transitions from a

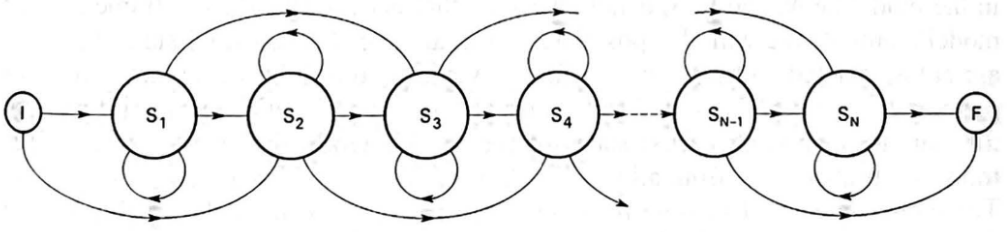


Figure 9.1. State transitions for a simple word model, from an initial state, I, to a final state, F.

Ex.1019 / Page 32 of 102
AMAZON.COM, INC.

state at any frame time must sum to 1, but the sum includes the probability of a transition that re-enters the same state. When the model is activated a sequence of feature vectors is emitted, in the same form as might be observed when a word is spoken. However, in the type of model considered here, observing the feature vectors does not completely determine what the state sequence is. In addition to its transition probabilities, each state also has associated with it a **probability density function (p.d.f.)** for the feature vectors. Each p.d.f. can be used to calculate the probability that any particular set of feature values could be emitted when the model is in the associated state. This probability is usually known as the **emission probability**. The actual values of the observed features are, therefore, probabilistic functions of the states, and the states themselves are hidden from the observer. For this reason this type of model is called a **hidden Markov model (HMM)**.

The emission p.d.f. for a state may be represented as a discrete distribution, with a probability specified separately for each possible feature vector. Alternatively, it is possible to use a parameterized continuous distribution, in which feature vector probabilities are defined by the parameters of the distribution. Although there are significant advantages, which will be explained in Section 9.7, in modelling feature probabilities as continuous functions, it will simplify the following explanation if we initially consider only discrete probability distributions.

9.3 PROBABILITY CALCULATIONS IN HIDDEN MARKOV MODELS

In order to explain the HMM probability calculations, we will need to introduce some symbolic notation to represent the different quantities which must be calculated. Notation of this type can be found in many publications on the subject of HMMs for ASR. Certain symbols have come to be conventionally associated with particular quantities, although there is still some variation in the details of the notation that is used. In choosing the notation for this book, our aims were to be consistent with what appears to be used the most often in the published literature, while also being conceptually as simple as possible.

We will start by assuming that we have already derived good estimates for the parameters of all the word models. (Parameter estimation will be discussed later in the chapter.) The recognition task is to determine the most probable word, given the observations (i.e. the word w for which $P(w|\boldsymbol{Y})$ is maximized). As explained in Section 9.2, we therefore need to calculate the likelihood of each model emitting the observed sequence of features (i.e. the value of $P(\boldsymbol{Y}|w)$ for each word w).

Considering a single model, an output representing a whole word arises from the model going through a sequence of states, equal in length to the number of observed feature vectors, T, that represents the word. Let the total number of states in the model be N, and let s_t denote the state that is occupied during frame t of the model's output. We will also postulate an initial state, I and a final state, F, which are not associated with any emitted feature vector and only have a restricted set of possible transitions. The initial state is used to specify transition probabilities from the start to all permitted first states of the model, while the final state provides transition probabilities from all possible last emitting states to the end of the word. The model must start in state I and end in state F, so in total the model will go through a sequence of $T+2$ states to generate T observations. The use of non-

Ex.1019 / Page 33 of 102
AMAZON.COM, INC.

emitting initial and final states provides a convenient method for modelling the fact that some states are more likely than others to be associated with the first and the last frame of the word respectively[1]. These compulsory special states will also be useful in later discussions requiring *sequences* of models.

The most widely used notation for the probability of a transition from state i to state j is a_{ij}. The emission probability of state j generating an observed feature vector $\boldsymbol{y}_t$ is usually denoted $b_j(\boldsymbol{y}_t)$.

We need to compute the probability of a given model producing the observed sequence of feature vectors, $\boldsymbol{y}_1$ to $\boldsymbol{y}_T$. We know that this sequence of observations must have been generated by a state sequence of length T (plus the special initial and final states) but, because the model is hidden, we do not know the identities of the states. Hence we need to consider all possible state sequences of length T. The probability of the model generating the observations can then be obtained by finding the **joint probability** of the observations and any one state sequence, and summing this quantity over all possible state sequences of the correct length:

$$P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T) = \sum_{\substack{\text{over all possible} \\ \text{state sequences} \\ \text{of length } T}} P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T, s_1, s_2, \cdots, s_T)$$

$$= \sum_{\substack{\text{over all possible} \\ \text{state sequences} \\ \text{of length } T}} P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T, | s_1, s_2, \cdots, s_T) P(s_1, s_2, \cdots, s_T), \qquad (9.2)$$

where, for notational convenience, in the equations we are omitting the dependence of all the probabilities on the identity of the model.

Now the probability of any particular state sequence is given by the product of the transition probabilities:

$$P(s_1, s_2, \cdots, s_T) = a_{Is_1} \left(\prod_{t=1}^{T-1} a_{s_t s_{t+1}} \right) a_{s_T F}, \qquad (9.3)$$

where $a_{s_t s_{t+1}}$ is the probability of a transition from the state occupied at frame t to the state at frame $t+1$; a_{Is_1} and $a_{s_T F}$ similarly define the transition probabilities from the initial state I and to the final state F. If we assume that the feature vectors are generated independently for each state, the probability of the observations given a particular state sequence of duration T is the product of the individual emission probabilities for the specified states:

$$P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T \mid s_1, s_2, \cdots, s_T) = \prod_{t=1}^{T} b_{s_t}(\boldsymbol{y}_t). \qquad (9.4)$$

---

[1] Some published descriptions of HMM theory do not include special initial and final states. Initial conditions are sometimes accommodated by a vector of probabilities for starting in each of the states (e.g. Levinson *et al.*, 1983), which has the same effect as the special initial state used here. For the last frame of the word, approaches include allowing the model to end in any state (e.g. Levinson *et al.*, 1983) or enforcing special conditions to only allow the model to end in certain states. The treatment of the first and last frames does not alter the basic form of the probability calculations, but it may affect the details of the expressions associated with the start and end of an utterance.

Ex.1019 / Page 34 of 102
AMAZON.COM, INC.

Thus the probability of the model emitting the complete observation sequence is:

$$P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T) = \sum_{\substack{\text{over all possible} \\ \text{state sequences} \\ \text{of length } T}} a_{Is_1} \left(\prod_{t=1}^{T-1} b_{s_t}(\boldsymbol{y}_t) a_{s_t s_{t+1}} \right) b_{s_T}(\boldsymbol{y}_T) a_{s_T F} \,. \quad (9.5)$$

Unless the model has a small number of states and T is small, there will be an astronomical number of possible state sequences, and it is completely impractical to make the calculations of Equation (9.5) directly for all sequences. One can, however, compute the probability indirectly by using a recurrence relationship. We will use the symbol $\alpha_j(t)$ to be the probability[2] of the model having produced the first t observed feature vectors and being in state j for frame t. The recurrence can be computed in terms of the values of $\alpha_i(t-1)$ for all possible previous states, i.

$$\alpha_j(t) = P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_t, s_t = j) \quad (9.6)$$

$$= \left(\sum_{i=1}^{N} \alpha_i(t-1) a_{ij} \right) b_j(\boldsymbol{y}_t) \quad \text{for } 1 < t \leq T \quad (9.7)$$

The value of $\alpha_j(1)$, for the first frame, is the product of the transition probability a_{Ij} from the initial state I, and the emission probability $b_j(\boldsymbol{y}_1)$.

$$\alpha_j(1) = a_{Ij} b_j(\boldsymbol{y}_1) \quad (9.8)$$

The value of $\alpha_j(T)$, for the last frame in the observation sequence, can be computed for any of the emitting states by repeated applications of Equation (9.7), starting from the result of Equation (9.8).

The total probability of the complete set of observations being produced by the model must also include the transition probabilities into the final state F. We will define this quantity as $\alpha_F(T)$, thus:

$$P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T) = \alpha_F(T) = \sum_{i=1}^{N} \alpha_i(T) a_{iF} \,. \quad (9.9)$$

Equation (9.9) gives the probability of the model generating the observed data, taking into account all possible sequences of states. This quantity represents the probability of the observations given the word model (the $P(\boldsymbol{Y}|w)$ term in Equation (9.1)). Incorporating the probability of the word, $P(w)$, gives a probability that is a scaled version of $P(w|\boldsymbol{Y})$, the probability of the word having been spoken. Provided that the model is a good representation of its intended word, this probability provides a useful measure which can be compared with the probability according to alternative word models in order to identify the most probable word.

---

[2] In the literature, this probability is almost universally represented by the symbol α. However, there is some variation in the way in which the α symbol is annotated to indicate dependence on state and time. In particular, several authors (e.g. Rabiner and Juang (1993)) have used $\alpha_t(j)$, whereas we have chosen $\alpha_j(t)$ (as used by Knill and Young (1997) for example). The same variation applies to the quantities β, γ and ξ, which will be introduced later. The differences are only notational and do not affect the meaning of the expressions, but when reading the literature it is important to be aware that such differences exist.

Ex.1019 / Page 35 of 102
AMAZON.COM, INC.

9.4 THE VITERBI ALGORITHM

The probability of the observations, given the model, is made up of contributions from a very large number of alternative state sequences. However, the probability distributions associated with the states will be such that the probability of the observed feature vectors having been produced by many of the state sequences will be microscopically small compared with the probabilities associated with other state sequences. One option is to ignore all but the single most probable state sequence. Equation (9.2) can be modified accordingly to give the probability, $\hat{P}$, of the observations for this most probable state sequence:

$$\hat{P}(y_1, y_2, \cdots, y_T) = \max_{\substack{\text{over all possible} \\ \text{state sequences} \\ \text{of length } T}} \left(P(y_1, y_2, \cdots, y_T, s_1, s_2, \cdots, s_T)\right). \qquad (9.10)$$

The probability associated with the most probable sequence of states can be calculated using the **Viterbi algorithm** (Viterbi, 1967), which is a dynamic programming algorithm applied to probabilities. Let us define a new probability, $\hat{\alpha}_j(t)$ as the probability of being in the j^{th} state, after having emitted the first t feature vectors and *having been through the most probable sequence* of $t-1$ preceding states in the process. Again we have a recurrence relation, equivalent to the one shown in Equation (9.7):

$$\hat{\alpha}_j(t) = \max_{\text{over } i} \left(\hat{\alpha}_i(t-1)a_{ij}\right) b_j(y_t) \quad \text{for } 1 < t \le T \ . \qquad (9.11)$$

The conditions for the first state are the same as for the total probability, which was given in Equation (9.8):

$$\hat{\alpha}_j(1) = \alpha_j(1) = a_{Ij} b_j(y_1) \ . \qquad (9.12)$$

Successive applications of Equation (9.11) will eventually yield the values for $\hat{\alpha}_j(T)$. Defining $\hat{\alpha}_F(T)$ as the probability of the full set of observations being given by the most probable sequence of states, its value is given by:

$$\hat{P}(y_1, y_2, \cdots, y_T) = \hat{\alpha}_F(T) = \max_{\text{over } i} \left(\hat{\alpha}_i(T) a_{iF}\right). \qquad (9.13)$$

The difference between the total probability and the probability given by the Viterbi algorithm depends on the magnitude of the contribution of the 'best' state sequence to the total probability summed over all possible sequences. If the feature-vector p.d.f.s of all states are substantially different from each other, the probability of the observations being produced by the best sequence might not be appreciably less than the total probability including all possible sequences. The difference between the total probability and the probability for the best sequence will, however, be larger if the best path includes several consecutive frames shared between a group of two or more states which have very similar p.d.f.s for the feature vectors. Then the probability of generating the observed feature vectors would be almost independent of how the model distributed its time between the states in this group. The total probability, which is the sum over all possible allocations of frames to states, could then be several times the probability for the

Ex.1019 / Page 36 of 102
AMAZON.COM, INC.

best sequence. This point will be considered again in Section 9.14. However, the design of models used in current recognizers is such that sequences of states with similar emission p.d.f.s generally do not occur. As a consequence, in spite of the theoretical disadvantage of ignoring all but the best path, in practice the differences in performance between the two methods are usually small. Some variant of the Viterbi algorithm is therefore usually adopted for decoding in practical speech recognizers, as using only the best path requires less computation. (There can also be considerable advantages for implementation, as will be discussed in Section 9.12.)

9.5 PARAMETER ESTIMATION FOR HIDDEN MARKOV MODELS

So far, we have considered the probability calculations required for recognition. We have assumed that the parameters of the models, i.e. the transition probabilities and emission p.d.f.s for all the states, are already set to their optimum values for modelling the statistics of a very large number of human utterances of all the words that are to be recognized. In the discussion which follows we will consider the problem of deriving suitable values for these parameters from a quantity of training data. We will assume for the moment that the body of training data is of sufficient size to represent the statistics of the population of possible utterances, and that we have sufficient computation available to perform the necessary operations.

The training problem can be formulated as one of determining the values of the HMM parameters in order to maximize the probability of the training data being generated by the models ($P(\boldsymbol{Y}|w)$ in Equation (9.1)). Because this conditional probability of the observations $\boldsymbol{Y}$ given word w is known as the 'likelihood' of the word w, the training criterion that maximizes this probability is referred to as **maximum likelihood** (other training criteria will be considered in Chapter 11). If we knew which frames of training data corresponded to which model states, then it would be straightforward to calculate a maximum-likelihood estimate of the probabilities associated with each state. The transition probabilities could be calculated from the statistics of the state sequences, and the emission probabilities from the statistics of the feature vectors associated with each state. However, the 'hidden' nature of the HMM states is such that the allocation of frames to states cannot be known. Therefore, although various heuristic methods can be formulated for analysing the training data to give rough estimates of suitable model parameters, there is no method of calculating the optimum values directly.

If, however, one has a set of rough estimates for all the parameters, it is possible to use their values in a procedure to compute new estimates for each parameter. This algorithm was developed by Baum and colleagues and published in a series of papers in the late 1960s and early 1970s. It has been proved by Baum (1972) that new parameter estimates derived in this way always produce a model that is at least as good as the old one in representing the data, and in general the new estimates give an improved model. If we iterate these operations a sufficiently large number of times the model will converge to a locally optimum solution. Unfortunately, it is generally believed that the number of possible local optima is so vast that the chance of finding the global optimum is negligible. However, it is unlikely that the global optimum would in practice be much better than a good local optimum, derived after initialization with suitable starting estimates for the models.

Ex.1019 / Page 37 of 102
AMAZON.COM, INC.

Baum's algorithm is an example of a general method which has come to be known as the **expectation-maximization (EM) algorithm** (Dempster *et al.*, 1977). The EM algorithm is applicable to a variety of situations in which the task is to estimate model parameters when the observable data are 'incomplete', in the sense that some information (in this case the state sequence) is missing.

The detailed mathematical proofs associated with the derivation of the re-estimation formulae for HMMs are beyond the scope of this book, although Chapter 17 gives some references. In the current chapter, we will describe the re-estimation calculations and give some intuitive explanation. The basic idea is to use some existing estimates for the model parameters to calculate the probability of being in each state at every frame time, given these current estimates of the model parameters *and* the training data. The probabilities of occupying the states can then be taken into account when gathering the statistics of state sequences and of feature vectors associated with the states, in order to obtain new estimates for the transition probabilities and for the emission probabilities respectively. In the re-estimation equations we will use a bar above the symbol to represent a re-estimated value, and the same symbol without the bar to indicate its previous value.

9.5.1 Forward and backward probabilities

Suppose for the moment that we have just a single example of a word, and that this example comprises the sequence of feature vectors $\boldsymbol{y}_1$ to $\boldsymbol{y}_T$. Also, assume that the word has been spoken in isolation and we know that $\boldsymbol{y}_1$ corresponds to the first frame of the word, with $\boldsymbol{y}_T$ representing the last frame. In Equation (9.7) we showed how to compute $\alpha_j(t)$, which is the probability of the model having emitted the first t observed feature vectors and being in state j. The values of $\alpha_j(t)$ are computed for successive frames in order, going **forward** from the beginning of the utterance. When estimating parameters for state j, we will need to know the probability of being in the state at time t, while the model is in the process of emitting *all* the feature vectors that make up the word. For this purpose we also need to compute $\beta_j(t)$, which is defined as the **backward** probability of emitting the remaining $T-t$ observed vectors that are needed to complete the word, given that the j^{th} state was occupied for frame t :

$$\beta_j(t) = P(\boldsymbol{y}_{t+1}, \boldsymbol{y}_{t+2}, \cdots, \boldsymbol{y}_T \mid s_t = j)\,. \tag{9.14}$$

When calculating the backward probabilities, it is necessary to start applying the recurrence from the end of the word and to work backwards through the sequence of frames. Each backward probability at time t is therefore derived from the backward probabilities at time $t+1$. Because the notation convention is to move from state i to state j, it is usual to specify the recurrence relationship for the backward probabilities with the i^{th} state occupied at time t. Thus the value of $\beta_i(t)$ is computed in terms of the values of $\beta_j(t+1)$ for all possible following states j :

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(\boldsymbol{y}_{t+1}) \beta_j(t+1) \qquad \text{for } T > t \geq 1\,. \tag{9.15}$$

Ex.1019 / Page 38 of 102
AMAZON.COM, INC.

In contrast to Equation (9.7), it will be noticed that Equation (9.15) does not include the emission probability for frame t. This difference in form between the definitions of $\alpha_i(t)$ and $\beta_i(t)$ is necessary because of the way we will combine these quantities in Equation (9.17).

The first application of Equation (9.15) uses the fact that the model must be in the final state, F, at the end of the word. At this point all features will have been emitted, so the value of $\beta_i(T)$ is just the probability of a transition from state i to state F:

$$\beta_i(T) = a_{iF} . \tag{9.16}$$

The probability of the model emitting the full set of T feature vectors and being in the j^{th} state for the t^{th} observed frame must be the product of the forward and backward probabilities for the given state and frame pair, thus:

$$P(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_T, s_t = j) = \alpha_j(t)\beta_j(t) . \tag{9.17}$$

Although it is not relevant to parameter re-estimation, it is interesting to note that, as the probability of generating the full set of feature vectors and being in state j for frame t is given by $\alpha_j(t)\beta_j(t)$, the probability of the observations irrespective of which state is occupied in frame t must be the value of this product summed over all states. We can write this probability as:

$$P(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_T) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t) \qquad \text{for any value of } t, \tag{9.18}$$

where here we use i as the state index for ease of comparison with Equation (9.9). Equation (9.18) is true for any value of the frame time, t, and Equation (9.9) is thus just a special case for the last frame, where $t = T$ and in consequence $\beta_i(T) = a_{iF}$.

9.5.2 Parameter re-estimation with forward and backward probabilities

In practice when training a set of models there would be several (say E) examples of each word, so the total number of feature vectors available is the sum of the numbers of frames for the individual examples. The re-estimation should use all the training examples with equal weight. For this purpose it is necessary to take into account that the current model would be expected to fit some examples better than others, and we need to prevent these examples from being given more weight in the re-estimation process. The simple product $\alpha_j(t)\beta_j(t)$ does not allow for these differences, as it represents the *joint* probability of being in state j at time t and generating a particular set of feature vectors representing one example. In order to be able to combine these quantities for different examples, we require the *conditional* probability of occupying state j given the feature vectors.

We will define a quantity $\gamma_j(t)$, which is the probability of being in state j for frame t, given the feature vectors for one example of the word. This quantity can be derived from $\alpha_j(t)\beta_j(t)$ using Bayes' rule, and it can be seen that the result involves simply normalizing $\alpha_j(t)\beta_j(t)$ by the probability of the model generating the observations.

Ex.1019 / Page 39 of 102
AMAZON.COM, INC.

$$\gamma_j(t) = P(s_t = j \mid \boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T) = \frac{P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T \mid s_t = j) P(s_t = j)}{P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T)}$$

$$= \frac{P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T, s_t = j)}{P(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T)} = \frac{\alpha_j(t)\beta_j(t)}{\alpha_F(T)} \tag{9.19}$$

The normalization by $\alpha_F(T)$ thus ensures that when there are several examples of the word, all frames of all examples will contribute equally to the re-estimation.

The probability, $b_j(\boldsymbol{k})$, of observing some particular feature vector, $\boldsymbol{k}$, when the model is in state j can be derived as the probability of the model being in state j and observing $\boldsymbol{k}$, divided by the probability of the model being in state j. In order to take into account the complete set of training examples of the word, we need to sum both the numerator and the denominator over all frames of all examples. Hence, assuming E examples of the word, the re-estimate for the emission probability is given by:

$$\bar{b}_j(\boldsymbol{k}) = \frac{\sum_{e=1}^{E} \sum_{\{t : \boldsymbol{y}_{te} = \boldsymbol{k}, t = 1,2,\ldots,T_e\}} \gamma_j(t,e)}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_j(t,e)}. \tag{9.20}$$

In Equation (9.20), quantities for the e^{th} example of the word are denoted by T_e for the number of frames in the example and $\boldsymbol{y}_{te}$ for the feature vector at the t^{th} frame of the example, with $\gamma_j(t,e)$ being used for the value of $\gamma_j(t)$ for the e^{th} example.

The denominator in Equation (9.20) is the sum of the individual probabilities of being in state j for each frame time, given the complete set of training data, and is sometimes referred to as the **state occupancy**. In some publications, the term **count** is also used when referring to this quantity. Although it is in fact a sum of probabilities, because it has been summed over the complete data set it is equivalent to the expected number, or count, of frames for which the state is occupied (although it will not in general be an integer number of frames).

In order to re-estimate the transition probabilities, we need to calculate the probability of a transition between any pair of states. This calculation is basically straightforward, but care needs to be taken to treat the start and end of the word correctly[3]. In the following explanation, transitions from the initial state and to the final state will be treated separately from transitions between emitting states.

Returning for the moment to considering only a single example of the word, let us define $\xi_{ij}(t)$ to be the probability that there is a transition from state i to state j at time t, given that the model generates the whole sequence of feature vectors representing the example of the word:

$$\xi_{ij}(t) = \frac{\alpha_i(t) a_{ij} b_j(\boldsymbol{y}_{t+1}) \beta_j(t+1)}{\alpha_F(T)} \quad \text{for } 1 \le t < T. \tag{9.21}$$

[3] The details of the equations given here apply to the use of special initial and final states and there will be slight differences if, for example, the model is allowed to end in any state (as in some publications).

Equation (9.21) can be applied to calculate the probability of a transition between any pair of emitting states at frame times starting from $t=1$ up until $t=T-1$. For the final frame, $t=T$, there cannot be a transition to another emitting state and the only possible transition is to the final state, F, with probability $\xi_{iF}(T)$, thus:

$$\xi_{iF}(T) = \frac{\alpha_i(T)a_{iF}}{\alpha_F(T)}. \tag{9.22}$$

For the initial state, we need to calculate the probability of a transition to each of the emitting states. This transition from the initial state is only possible at the start of the word, before any observations have been generated. If we regard this time as being $t=0$ then, given that the model must start in state I, another special instance of Equation (9.21) can be derived for all transitions out of state I, thus:

$$\xi_{Ij}(0) = \frac{a_{Ij}b_j(\mathbf{y}_1)\beta_j(1)}{\alpha_F(T)}. \tag{9.23}$$

The total probability of a transition between any pair of states i and j is obtained by summing the values of $\xi_{ij}(t)$ over all frames for which the relevant transition is possible. Dividing this quantity by the total probability γ_i of occupying state i gives the re-estimate for the transition probability a_{ij}. Assuming E examples of the word, for a transition between any two emitting states we have:

$$\bar{a}_{ij} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e-1}\xi_{ij}(t,e)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_i(t,e)} \quad \text{for } 1 \le i, j \le N, \tag{9.24}$$

where $\xi_{ij}(t, e)$ denotes the value of $\xi_{ij}(t)$ for the e^{th} training example. Note that the summation of $\xi_{ij}(t, e)$ over time only includes frames up until time $T_e - 1$. The last frame is not included as it cannot involve a transition to another emitting state, and so by definition the value of $\xi_{ij}(T, e)$ is zero for all pairs of emitting states.

Transitions from an emitting state to the final state F can only occur at time T_e and so the transition probability a_{iF} may be re-estimated as:

$$\bar{a}_{iF} = \frac{\sum_{e=1}^{E}\xi_{iF}(T_e,e)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_i(t,e)} \quad \text{for } 1 \le i \le N. \tag{9.25}$$

Transitions from the initial state I can only occur at the start (time $t=0$), when the model *must* be in state I, so $\gamma_I(0, e) = 1$ for all examples and hence:

$$\bar{a}_{Ij} = \frac{\sum_{e=1}^{E}\xi_{Ij}(0,e)}{E} \quad \text{for } 1 \le j \le N. \tag{9.26}$$

Ex.1019 / Page 41 of 102
AMAZON.COM, INC.

The use of forward and backward probabilities to re-estimate model parameters is usually known either as the **forward–backward algorithm** or as the **Baum–Welch algorithm**. The second name in "Baum–Welch" recognizes the fact that Lloyd Welch was working with Baum on this subject in the early 1960s.

After re-estimation using the Baum–Welch algorithm, the probability of the training data given the new set of models is guaranteed to be higher than the probability for the previous model set, except at the **critical point** at which a local optimum has been reached and therefore the models (and hence the probability) are unchanged. The procedure can thus be repeated in an iterative manner until the difference between the new and old probabilities is sufficiently small that the training process can be regarded as being close enough to its local optimum.

It can be seen from the expression of Equations (9.24), (9.25) and (9.26) using the quantities defined in Equations (9.21), (9.22) and (9.23) that, if any of the a_{ij} are initially given values of zero, their re-estimated values will also always be zero. Setting initial values of some transition probabilities to zero is thus a convenient way of constraining the structure of the word model to prevent it from producing intrinsically implausible state sequences. For example, it would not seem reasonable to allow the model to occupy a state early in the word, and then return to it after having been through several succeeding states. The sequence possibilities in Figure 9.1 are very limited, only allowing three non-zero values of a_{ij} for any state i, yet this structure is very plausible as a word model. Constraining the possible state sequences by setting most of the initial values of the transition probabilities to zero has the added benefit of greatly reducing the computation required for both recognition and training.

Model initialization issues, including the choice of initial conditions for the emission p.d.f.s, will be discussed in more detail later on in this chapter.

9.5.3 Viterbi training

It is also possible to re-estimate the model parameters using only the most likely path through the states, as given by the Viterbi algorithm. The calculations are substantially simplified by just considering a single path. For any frame of input data the probability of a state being occupied can only be unity or zero, depending on whether that state is on the path. The most likely path can be found by calculating the values of $\hat{\alpha}_j(t)$ for all states and frames to the end of the word using Equation (9.11), and then tracing back from the final state in the same way as for the DTW method described in Chapter 8. In contrast to Baum–Welch re-estimation, the backward probabilities are not required.

Having identified the most likely path, each input frame will have been allocated to a single state to provide a state-level segmentation of the training data. It will therefore be known which state produced each observed feature vector, and also which states preceded and followed each state along the path. For the re-estimation it is then only necessary, for all examples of each training word, to accumulate the statistics of the feature vectors that occur for each occupied state, and of the transitions between states along the most likely path. Using the identified path, there will need to be counts of the following events, totalled over all E examples of the word:

Ex.1019 / Page 42 of 102
AMAZON.COM, INC.

i. the number of frames for which each state gives rise to each of the possible feature vectors, with the count for state j and feature vector $\boldsymbol{k}$ being denoted by $n_j(\boldsymbol{y}_t = \boldsymbol{k})$;
ii. the number of frames for which a transition occurs between each pair of states, which for transitions between states i and j will be denoted by n_{ij};
iii. the number of occasions for which each state is occupied for the first frame of each example of the word, which for state j will be denoted by n_{1j};
iv. the number of occasions for which each state is occupied for the last frame of each example of the word, which for state i will be denoted by n_{iF};
v. the number of frames for which each state is occupied, which will be denoted by n_i and n_j for states i and j respectively.

The re-estimation formulae are then simply given by:

$$\bar{b}_j(\boldsymbol{k}) = \frac{n_j(\boldsymbol{y}_t = \boldsymbol{k})}{n_j}, \tag{9.27}$$

$$\bar{a}_{ij} = \frac{n_{ij}}{n_i} \quad \text{for all pairs of emitting states, } 1 \le i, j \le N, \tag{9.28}$$

$$\bar{a}_{iF} = \frac{n_{iF}}{n_i} \quad \text{for all } i \text{ such that } 1 \le i \le N, \tag{9.29}$$

$$\bar{a}_{1j} = \frac{n_{1j}}{E} \quad \text{for all } j \text{ such that } 1 \le j \le N. \tag{9.30}$$

Note that the above re-estimation equations for Viterbi training are in fact equivalent to the corresponding Baum–Welch equations (9.20, 9.24, 9.25, 9.26) with the values of all the frame-specific state occupancy probabilities ($\gamma_j(t, e)$, etc.) set either to one or to zero, depending on whether or not the relevant states are occupied at the given frame time. As with the Baum–Welch re-estimation, the Viterbi training procedure (determination of the most likely state sequence followed by estimation of the model parameters) can be applied in an iterative manner until the increase in the likelihood of the training data is arbitrarily small.

Because the contribution to the total probability is usually much greater for the most likely path than for all other paths, an iterative Viterbi training procedure usually gives similar models to those derived using the Baum–Welch recursions. However, the Viterbi method requires much less computation and it is therefore often (and successfully) adopted as an alternative to full Baum–Welch training.

9.6 VECTOR QUANTIZATION

In the discussion above it was assumed that the data used for training the models include a large enough number of words for reliable values to be obtained for all the parameters. For any statistical estimation to give sensible results it is obvious that the total number of data items must be significantly larger than the number of separate parameters to be estimated for the distribution. If the number of possible feature vectors is very large, as a result of many possible values for each of several individual features, many feature vectors will not occur at all in a manageable

Ex.1019 / Page 43 of 102
AMAZON.COM, INC.

amount of training data. In consequence all the generation probabilities for these feature vectors will be estimated as zero. If such a feature vector then occurred in the input during operational use of the recognizer, recognition would be impossible.

The multi-dimensional feature space for any practical method of speech analysis is not uniformly occupied. The types of spectrum cross-section that occur in speech signals cause certain regions of the feature space, for example those corresponding to the spectra of commonly occurring vowels and fricatives, to be highly used, and other regions to be only sparsely occupied. It is possible to make a useful approximation to the feature vectors that actually occur by choosing just a small subset of vectors, and replacing each measured vector by the one in the subset that is 'nearest' according to some suitable distance metric. This process of **vector quantization (VQ)** is also used in systems for efficient speech coding (see Section 4.3.5).

Setting up a vector quantizer usually involves first applying a **clustering** algorithm to group similar vectors together, then choosing a representative quantized vector for each cluster. The performance of such a quantizer depends on the number of different vectors and how they are chosen, but the details of these decisions are outside the scope of this book. It is, however, clear that if a fairly small **codebook** of vectors is chosen to represent the well-occupied parts of the feature space, all of these quantized vectors will occur frequently in a training database of moderate size. For each model state it will thus be possible to obtain good estimates for the probability of all feature vectors that are likely to occur.

Even after vector quantization, a fully trained model for a particular word will often have some feature vectors that are given zero probability for all states of the word. For example, the word "one" would not be expected to contain any examples of a feature representing the typical spectrum of an [s] sound. It is, however, important not to allow the probabilities to remain exactly at zero. Otherwise there is the danger of error on an input word that matches fairly well to the properties of one of the models except for just one non-typical frame that is represented by a zero-probability feature vector. In such a case the model will yield zero probability for that sequence of vectors, and the recognizer will therefore not be able to choose the correct word. A simple solution is to replace the zero value by a very small number. The model will then yield a low probability of generating the observed features, but if the rest of the word is sufficiently distinctive even this low value can be expected to be greater than the probability of generating the same set of features from any of the competing models. Better estimates for the probability of an unseen feature vector can be obtained by using a measure of distance from the vectors that are observed for the word, so that the unseen vector is given a higher probability if it is similar to those vectors which do occur in the training examples.

9.7 MULTI-VARIATE CONTINUOUS DISTRIBUTIONS

Vector quantization involves an approximation which unavoidably loses some information from the original data, and any method for estimating the probability of an unseen feature vector will inevitably be somewhat *ad hoc*. These limitations associated with discrete distributions can be overcome by representing the distribution of feature vectors by some suitable parametric description. Provided

Ex.1019 / Page 44 of 102
AMAZON.COM, INC.

that an appropriate parametric distribution can be found for describing the true distribution of the features, a useful estimate can be computed for the probability of *any* feature vector that may occur in the training and recognition processes.

Many natural processes involve variable quantities which approximate reasonably well to the **normal** (or **Gaussian**) distribution. The normal distribution has only two independently specifiable parameters, the **mean**, μ, and the **standard deviation**, σ. For a quantity x, the probability density, $\phi(x)$, is given by:

$$\phi(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right). \tag{9.31}$$

When quantities are distributed normally, this simple mathematical description of the distribution makes it possible to calculate the probability of the quantity lying in any range of values provided the mean and standard deviation of the distribution are known. To calculate the probability of one particular value (i.e. a measured acoustic feature vector) occurring, we need to consider the limiting case in which the size of the interval for the range of values is infinitesimally small.

The definition of the continuous probability density function, $\phi(x)$, of a variate, x, is such that the probability of an observation lying in an infinitesimal interval of size dx centred on x is $\phi(x)dx$, and is thus infinitesimally small. However, if continuous probability density functions are used instead of discrete probability distributions in the HMM equations given in Sections 9.3 to 9.5, the computation will still give the correct relative likelihoods of the different words, as the infinitesimal interval, dx, is common to all probability calculations. The probability of observing the features, $P(\boldsymbol{Y})$, independently of which word is spoken, is also affected in the same way by the size of dx. The probability of the word given the features is therefore still correctly given by the formula expressed in Equation (9.1), even if these probability densities are used instead of actual probabilities for $P(\boldsymbol{Y})$ and $P(\boldsymbol{Y}|w)$. Although their theoretical interpretations are different, it is thus equally suitable to use either discrete or continuous probability distributions in the calculations of word probability and in parameter re-estimation. In the following discussion of continuous distributions, it will be convenient to continue to use the term "probability" even where the quantities are, strictly speaking, probability densities.

9.8 USE OF NORMAL DISTRIBUTIONS WITH HMMS

It is obvious that many naturally occurring quantities are not normally distributed. For example, speech intensity measured over successive fixed time intervals of, say, 20 ms during continuous speech will certainly not approximate to a normal distribution because it clearly has a hard limit of zero during silences, will be low for much of the time during weak sounds, but will go up to quite high values during more intense vowels. The intensity on a logarithmic scale would have a more symmetrical distribution, which might be nearer to normal, but in this case the low-level end of the distribution will be very dependent on background noise level.

Normal distributions usually fit best to measurements which can be expected to have a preferred value, but where there are various chance factors that may cause

Ex.1019 / Page 45 of 102
AMAZON.COM, INC.

deviation either side of that value, with the probability progressively decreasing as the distance either side of the preferred value increases. Thus it might be reasonable to use a normal distribution to approximate a distribution of speech features which are derived from the same specific part of a specific word spoken in the same way by the same person. When it is assumed that features are normally distributed for each state of an HMM, the distributions are often termed **single Gaussian**.

When different speakers are combined in the same distribution the departures from normal will be greater, and for different regional accents there is a fairly high probability that the distribution will be multi-modal, and therefore much less suitable for modelling as a normal distribution. However, when multi-modal distributions are likely, as is the case with many current speech recognition systems, it is now almost universal to model the distributions with a weighted sum, or **mixture**, of several normal distributions with different means and variances (usually referred to as **Gaussian mixtures**). Provided that there is a sufficient number of mixture components, any shape of distribution can be approximated very closely. This characteristic of sums of Gaussian distributions, combined with the attractive mathematical properties of the Gaussian itself, is largely responsible for their widespread and successful use for describing emission probability distributions in HMM-based speech recognition systems.

The theory underlying the use of mixture distributions is a straightforward extension of the single-Gaussian case and will be discussed in Section 9.10, after first introducing the probability calculations and model parameter re-estimation equations using single Gaussian distributions.

9.8.1 Probability calculations

The features are multi-dimensional and so, in the case of single-Gaussian distributions, they will form a multi-variate normal distribution. In general the features may not vary independently, and their interdependence is specified by a **covariance matrix**. The entries along the main diagonal of this matrix represent the variance of each feature, while the remaining entries indicate the extent to which the separate feature distributions are correlated with each other.

Let us first consider the output probability $b_j(\boldsymbol{y})$ for the j^{th} state, where $\boldsymbol{y}$ is a single feature vector. Assume that the column vector $\boldsymbol{y}$ comprises K features, $y_1, y_2, \ldots, y_K$. Let $\boldsymbol{\mu}_j$ be the column vector of means, $\mu_{j1}, \mu_{j2}, \ldots, \mu_{jK}$, and $\boldsymbol{\Sigma}_j$ be the covariance matrix for the distribution of features associated with that state. The definition of the multi-variate normal distribution gives the output probability compactly in matrix notation:

$$b_j(\boldsymbol{y}) = \frac{1}{|\boldsymbol{\Sigma}_j|^{1/2}(2\pi)^{K/2}} \exp\left(\frac{-(\boldsymbol{y}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_j)}{2}\right), \tag{9.32}$$

where $|\boldsymbol{\Sigma}_j|$ is the determinant of $\boldsymbol{\Sigma}_j$ and $(\boldsymbol{y}-\boldsymbol{\mu}_j)^T$ is the transpose of $(\boldsymbol{y}-\boldsymbol{\mu}_j)$. In the special case when the features are uncorrelated, the covariance matrix becomes zero except along its main diagonal (and is therefore often referred to as a **diagonal**

Ex.1019 / Page 46 of 102
AMAZON.COM, INC.

covariance matrix). The probability of a feature vector then reduces to a product of probabilities given by the univariate distributions of the separate features:

$$b_j(\boldsymbol{y}) = \prod_{k=1}^{K} \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_k - \mu_{jk}}{\sigma_{jk}}\right)^2\right), \tag{9.33}$$

where y_k is the k^{th} feature of $\boldsymbol{y}$, and μ_{jk} and σ_{jk} are the mean and standard deviation of the distribution of the k^{th} feature for state j.

Equation (9.33) is evidently computationally simpler than Equation (9.32). The extent of the computational saving provides a strong motivation for choosing methods of speech analysis for which the features are substantially uncorrelated. Some of these methods will be described in Chapter 10. Most current speech recognition systems adopt such a method and use diagonal covariance matrices.

Having defined an expression for the emission probability in terms of the distribution parameters, recognition can be performed in the same way as when using discrete distributions. Thus, in the case of the Viterbi algorithm, the new definition of $b_j(\boldsymbol{y})$ is simply used in Equations (9.11) and (9.12).

9.8.2 Estimating the parameters of a normal distribution

When modelling emission probabilities with continuous distributions, the training task is to optimize the parameters of the feature distribution model, rather than the probabilities of particular feature vectors. If we had a set of T feature vectors that were known to correspond to state j, then the maximum-likelihood estimates for the parameters of a normal distribution are easily calculated. The mean vector $\hat{\boldsymbol{\mu}}_j$ is equal to the average of all the observed vectors (i.e. the sample mean), and the covariance matrix $\hat{\boldsymbol{\Sigma}}_j$ is obtained based on the deviation of each of the observed vectors from the estimated mean vector (i.e. the sample covariance matrix):

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}_t , \tag{9.34}$$

$$\hat{\boldsymbol{\Sigma}}_j = \frac{1}{T}\sum_{t=1}^{T} (\boldsymbol{y}_t - \hat{\boldsymbol{\mu}}_j)(\boldsymbol{y}_t - \hat{\boldsymbol{\mu}}_j)^T . \tag{9.35}$$

Obviously, in the case of HMMs, the state sequence is not known, but the standard methods for estimating mean and covariance given in Equations (9.34) and (9.35) can be extended for use in either Baum–Welch or Viterbi re-estimation procedures, as explained below.

9.8.3 Baum–Welch re-estimation

For the Baum–Welch algorithm, the parameters are re-estimated using contributions from all frames of all the E examples of the word in the training data.

Ex.1019 / Page 47 of 102
AMAZON.COM, INC.

Each contribution is weighted by the probability of being in the state at the relevant frame time, as given by Equation (9.19). Therefore the re-estimates of the mean vector μ_j and the covariance matrix Σ_j associated with state j are given by:

$$\bar{\mu}_j = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_j(t,e)\,\mathbf{y}_{te}}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_j(t,e)}, \tag{9.36}$$

$$\bar{\Sigma}_j = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_j(t,e)\,(\mathbf{y}_{te}-\bar{\mu}_j)(\mathbf{y}_{te}-\bar{\mu}_j)^T}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_j(t,e)}, \tag{9.37}$$

where $\mathbf{y}_{te}$ is the feature vector for the t^{th} frame of the e^{th} example of the word.

Just as for discrete emission p.d.f.s, it can be shown that iterative application of the above formulae leads to a locally optimum solution. Baum's (1972) analysis included a proof for univariate normal distributions, which was later generalized by Liporace (1982) to a wider class of distributions, including multi-variate normal distributions.

Note that Equation (9.37) for re-estimating the covariance matrix is based on deviation of observed vectors from the *re-estimated* mean vector $\bar{\mu}_j$. In practice, when accumulating the contributions for covariance re-estimation it is easier to use the current estimate μ_j instead of the (yet to be computed) new value $\bar{\mu}_j$. It is then straightforward to correct for the difference between the old and new mean values at the end of the calculation.

9.8.4 Viterbi training

For Viterbi re-estimation, the requirement is just to use the state-level segmentation obtained from the most likely path according to the current set of models as the basis for collecting the statistics needed to apply Equations (9.34) and (9.35) for each model state. The statistics for state j are therefore gathered over all examples of the word using all frames for which state j is occupied. Using s_{te} to denote the state occupied at frame t of example e, the re-estimation formulae are as follows:

$$\bar{\mu}_j = \frac{1}{n_j}\sum_{e=1}^{E}\sum_{t\ni s_{te}=j}\mathbf{y}_{te}, \tag{9.38}$$

$$\bar{\Sigma}_j = \frac{1}{n_j}\sum_{e=1}^{E}\sum_{t\ni s_{te}=j}(\mathbf{y}_{te}-\bar{\mu}_j)(\mathbf{y}_{te}-\bar{\mu}_j)^T, \tag{9.39}$$

where, as in Section 9.5.3, n_j is the number of frames for which state j is occupied.

Ex.1019 / Page 48 of 102
AMAZON.COM, INC.

9.9 MODEL INITIALIZATION

There must be enough states in the model to capture all the acoustically distinct regions in the word. For example, a typical word of two or three syllables could need around 10–20 states to model the acoustic structure adequately. Because the training process only finds a local optimum, the initial estimates for the model parameters can have a strong influence on the characteristics of the final set of trained models. It is very important to give careful consideration to how the model parameters, including both transition and emission probabilities, should be initialized before training. The trained model for each word needs to capture the spectral and temporal characteristics of all spoken utterances of that word while at the same time, in order to minimize recognition errors, it must be a constraining model which does not allow inappropriate sequences of states for the word.

In an HMM, the probability of a path through the model is computed on a frame-by-frame basis and therefore cannot take into account any of the previous states occupied other than the one at the immediately preceding time frame. Thus, if a model allows many different transitions from each state, recognition errors can result if a sequence of frames gives a good acoustic match even if the complete state sequence is very inappropriate for a genuine example of the word. Even the limited degree of flexibility included in the model structure shown in Figure 9.1 can cause problems if used throughout a word.

The dangers associated with allowing flexibility of transitions within a word model are such that most current uses of HMMs only allow a very restricted set of possible transitions, by initializing most of the transition probabilities to zero. A popular HMM structure for speech recognition uses a left-to-right topology with the probability of all transitions set to zero except those to the next state or returning to the current state (i.e. as for Figure 9.1 but omitting the 'skip' transitions). If this model structure is used to represent a word, the word will be modelled as a sequence of acoustic regions which can vary in duration but which must always all occur and always in the same fixed order. With this strong temporal constraint provided by the model structure, re-estimation (using either the Baum–Welch or the Viterbi approach) can give a useful local optimum even with a simple initialization approach for the emission probabilities. One popular strategy is to start with a uniform segmentation of each training example, with the number of segments being equal to the number of states in the model. This segmentation can then be used to compute the required statistics for each state emission probability, with the allowed transition probabilities of all emitting states initialized to identical values.

In the case of Baum–Welch training, an even simpler initialization strategy may be used for the parameters of discrete or normal distributions. For this method, sometimes called a **flat start** (Knill and Young, 1997), all emission p.d.f.s for all states are set to average values computed over the entire training set, in addition to using identical transition probabilities for a limited set of allowed transitions. Thus all permitted paths through the model start with equal probability and the training algorithm is left to optimize the parameters from this neutral starting position with constraints imposed by the model structure. This approach has been found to work well if there are several utterances for each model unit (e.g. Paul and Martin, 1988).

An important advantage of the initialization approaches described above is that the training process can be carried out completely automatically, without

Ex.1019 / Page 49 of 102
AMAZON.COM, INC.

requiring any pre-segmented data. Alternatively, if there are any data available for which suitable state boundaries are 'known' (for example, the boundaries could be marked by hand for a small subset of the training corpus), this segmentation can be used as the basis for initializing some or all of the model parameters.

If all models use a restricted structure that only allows transitions back to the same state or on to the next state, it is implicitly assumed that such a model structure is appropriate for representing all words. There are many cases in human language where pronunciation varies from occasion to occasion, even for one speaker. The variations may be at the phonemic level: for example, in the word "seven" many speakers often omit the vowel from the second syllable and terminate the word with a syllabic [n]. Allophonic variations can also occur: for example, in words ending in a stop consonant, the consonant may or may not be released. If a word with alternative pronunciations is represented by a single sequence of states with the model structure described above, some states will have to cope with the different pronunciations, and so their p.d.f.s will need to be multi-modal to model the distributions well. In these cases a normal distribution will not be suitable, and Gaussian mixtures will be essential for good modelling of the data.

A rather different approach is to explicitly model alternative pronunciations as alternative state sequences, using either whole-word or sub-word models. Initialization then involves choosing a constraining topology separately for each word model to take into account the possible phonetic structure of the word and its expected variation. It will thus be necessary to decide on the number of states required to represent each phonetic event and on the allowed transitions between the states, with state skips being allowed only where a particular phonetic event is sometimes omitted. For this approach to work it is essential that the emission p.d.f.s of the models are initialized with values roughly appropriate for the phonetic events expected for each state, because otherwise the training frames may not be allocated to the states in the intended way. Such models can be initialized by carefully hand-labelling a few examples of the training words in terms of state labels, and collecting the statistics of these data to initialize the emission p.d.f.s. However, the whole method requires a lot of skilled human intervention, and a simpler model topology is usually adopted, with any limitations in this approach being addressed by using Gaussian mixtures for the p.d.f.s. Methods used for including some simple provision for alternative pronunciations will be considered further in Chapter 12.

9.10 GAUSSIAN MIXTURES

9.10.1 Calculating emission probabilities

The expression for the emission probability defined in Equation (9.32) is easily extended to include a weighted sum of normal distributions, where each component distribution has a different mean and variance. We will use the notation $N(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ to represent the probability density of the observed vector $\boldsymbol{y}$ given a normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Thus, for an emission p.d.f. defined according to a Gaussian mixture distribution, the emission probability given by the m^{th} component for state j is:

$$b_{jm}(\boldsymbol{y}) = N\left(\boldsymbol{y}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}\right). \tag{9.40}$$

Ex.1019 / Page 50 of 102
AMAZON.COM, INC.

The total emission probability for a distribution with M components is defined as:

$$b_j(\boldsymbol{y}) = \sum_{m=1}^{M} c_{jm} b_{jm}(\boldsymbol{y}), \tag{9.41}$$

where c_{jm} denotes the weight of the m^{th} mixture component for state j. The mixture component weights can only take positive values, $c_{jm} \geq 0$, and must sum to 1:

$$\sum_{m=1}^{M} c_{jm} = 1 . \tag{9.42}$$

In the special case where there is only one mixture component, the emission probability specified by Equation (9.41) is defined in terms of a single Gaussian distribution with weight equal to 1 and is therefore equivalent to Equation (9.33).

Once the parameters of multiple-component mixture distributions have been trained, Equation (9.41) can be used as the basis for the recognition calculations in exactly the same way as with the simpler emission p.d.f.s that we have already discussed. Parameter estimation for mixture distributions requires more detailed consideration, and is discussed in the following sections. Firstly we will assume that initial estimates are available and address the re-estimation problem, before considering ways of obtaining suitable initial estimates in Section 9.10.4.

9.10.2 Baum–Welch re-estimation

Assuming that initial estimates are available for all the parameters of all the M components of a Gaussian mixture representing the emission p.d.f. for state j, Baum–Welch re-estimation can be used to find new estimates for these parameters, c_{jm}, $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$. When using Gaussian mixtures, the contribution from each observation $\boldsymbol{y}_t$ needs to be weighted by a probability that is specific to the mixture component m. By analogy with the quantity $\gamma_j(t)$ which was introduced in Section 9.5.2, let us define $\gamma_{jm}(t)$ to be the probability of being in state j at time t and using component m to generate $\boldsymbol{y}_t$, given that the model generates the whole sequence of T feature vectors representing an example of the word.

$$\gamma_{jm}(t) = \frac{\sum_{i=1}^{N} \alpha_i(t-1) a_{ij} c_{jm} b_{jm}(\boldsymbol{y}_t) \beta_j(t)}{\alpha_F(T)} \tag{9.43}$$

Now, if we have E examples of the word, summing the values of $\gamma_{jm}(t, e)$ over all frames of all examples gives the total probability for the m^{th} component of state j generating an observation. Dividing this quantity by the corresponding sum of $\gamma_j(t, e)$ terms gives the re-estimate for the mixture component weight c_{jm}:

$$\bar{c}_{jm} = \frac{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{jm}(t,e)}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_j(t,e)} . \tag{9.44}$$

Ex.1019 / Page 51 of 102
AMAZON.COM, INC.

The re-estimation equations for the mean vector and covariance matrix are the same as for the single-Gaussian case given in Equations (9.36) and (9.37), but using the component-specific state occupation probabilities $\gamma_{jm}(t,e)$:

$$\bar{\mu}_{jm} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{jm}(t,e)\mathbf{y}_{te}}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{jm}(t,e)}, \tag{9.45}$$

$$\bar{\Sigma}_{jm} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{jm}(t,e)(\mathbf{y}_{te}-\bar{\mu}_{jm})(\mathbf{y}_{te}-\bar{\mu}_{jm})^T}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{jm}(t,e)}. \tag{9.46}$$

Juang (1985) extended Liporace's (1982) analysis to show that iterative application of the re-estimation formulae leads to a locally optimum solution when emission p.d.f.s are defined in terms of sums of normal distributions.

9.10.3 Re-estimation using the most likely state sequence

The use of a Gaussian mixture to represent the HMM emission p.d.f. incorporates another 'hidden' element in the model, as it is not known from the observations which mixture component generated each observation. The probability calculation in Equation (9.41) uses the total probability taking into account all the mixture components that could have produced the observation. As a result the Baum–Welch re-estimation formulae in Equations (9.44) to (9.46) use probabilities not only of state occupancy but also of mixture components. The formulae can be simplified if the state sequence is known, but the situation is more complex than for the single-Gaussian case because the mixture components are still unknown. Thus the state sequence alone does not lead to an analytic solution for these emission p.d.f.s.

One option is to retain the EM algorithm for estimating the parameters of the mixture distribution. Equations (9.44) to (9.46) can be simplified accordingly: the summations are now just over those frames for which state j is occupied, and for each frame the component-dependent state occupation probability $\gamma_{jm}(t,e)$ simplifies to a component-dependent emission probability $c_{jm}b_{jm}(\mathbf{y}_{te})$. (The total state occupation probability $\gamma_j(t,e)$ is replaced by the emission probability $b_j(\mathbf{y}_{te})$.)

Alternatively, to estimate the distribution parameters without requiring an EM algorithm, each observation must be assigned to a single mixture component. This assignment can be achieved by using a clustering procedure to divide the observed feature vectors corresponding to any one model state into a number of groups equal to the number of mixture components for that state. ***K*-means clustering** is a well-established technique for dividing a set of vectors into a specified number of classes in order to locally minimize some within-class distance metric, and was originally applied to vector quantization (see Section 9.6). The term **segmental**

Ex.1019 / Page 52 of 102
AMAZON.COM, INC.

K-means is often used to refer to the use of K-means clustering in conjunction with a Viterbi alignment procedure to identify the state-level segmentation.

After clustering, each frame will be labelled, not only with the state that was occupied, but also with the mixture component that generated the observation. The re-estimation formula for the weight associated with the m^{th} mixture component of state j is then given by:

$$\bar{c}_{jm} = \frac{n_{jm}}{n_j}, \tag{9.47}$$

where n_{jm} represents the number of frames for which state j was occupied and mixture component m generated an observation. Using s_t to denote the state occupied and x_t to denote the mixture component used at time t, the re-estimation formulae for the mean feature vector and covariance matrix are straightforward extensions of the single-Gaussian case (Equations (9.38) and (9.39)), as follows:

$$\bar{\mu}_{jm} = \frac{1}{n_{jm}} \sum_{e=1}^{E} \sum_{t \ni s_t = j, x_t = m} y_{te}, \tag{9.48}$$

$$\bar{\Sigma}_{jm} = \frac{1}{n_{jm}} \sum_{e=1}^{E} \sum_{t \ni s_t = j, x_t = m} (y_{te} - \bar{\mu}_{jm})(y_{te} - \bar{\mu}_{jm})^T. \tag{9.49}$$

9.10.4 Initialization of Gaussian mixture distributions

The segmental K-means procedure outlined above uses an initial set of models to obtain the state-level segmentation, but does not rely on any existing estimates for the mixture components. It therefore provides a convenient method for initializing the parameters of HMMs using mixture distributions. If no models are available, the process can even be started from a uniform segmentation, as described in Section 9.9. Once initial estimates have been obtained for all the mixture components, the estimates can be refined using further iterations of the segmental K-means procedure. At this point the models could be used for recognition, but they can be trained further using full Baum–Welch re-estimation, or even using the EM algorithm to update the mixture parameters without changing the segmentation.

A segmental K-means procedure is often used to initialize mixture models prior to Baum–Welch training. However, this approach requires the number of mixture components to be decided in advance. An alternative is to start with trained single-Gaussian models and to incrementally increase the number of mixture components using a method often referred to as **mixture splitting**. Starting with a single Gaussian distribution for the emission p.d.f., a two-component mixture model is initialized by duplicating the parameters of the original distribution and perturbing the means by a small amount in opposite directions (typically ±0.2 standard deviations). The variances are left unchanged and the mixture weights are set to 0.5 for both components. The means, variances and mixture weights are all re-estimated, and the mixture-splitting procedure is then applied to the component with the largest weight (setting the weights of both new components to half the

Ex.1019 / Page 53 of 102
AMAZON.COM, INC.

value for the component from which they were derived). The model parameters are re-estimated again, and so on until the desired level of complexity is reached.

For a given number of mixture components, Young and Woodland (1993) reported that an iterative mixture-splitting training procedure with Baum–Welch re-estimation gave similar results to using segmental *K*-means followed by Baum–Welch training. However, a useful advantage of the mixture-splitting approach is that the number of mixture components can be chosen for each state individually according to some objective criterion based on how well the data are modelled. Examples of useful criteria for deciding on the number of components are the magnitude of the increase in training-data likelihood from adding a new component, or the quantity of training data available for the model concerned. This flexibility of mixture modelling is particularly beneficial for modelling large vocabularies; its use will be discussed further in Chapter 12.

9.10.5 Tied mixture distributions

Increasing the number of components used in a Gaussian mixture distribution allows for greater flexibility in the shapes of distributions that can be modelled, but a larger quantity of training data is required to ensure that the parameters are trained robustly. In any practical recognizer there are often only limited data available for training each model, which imposes limitations on the number of state-specific mixture components that can be included. However, similarities between different speech sounds are such that many of the component distributions will be similar for several different states. One straightforward way of taking advantage of these similarities to provide more data for training the model parameters is to use the same Gaussian distributions to represent *all* the states of all the models, with only the mixture weights being state-specific. Thus the distribution parameters are tied across the different states, and this type of model is often referred to as a **tied mixture** (Bellegarda and Nahamoo, 1990). The term **semi-continuous HMM** has also been used (Huang and Jack, 1989), because the one set of continuous distribution parameters for all states can be regarded as an alternative to the VQ-generated codebook used with discrete emission probabilities.

When using tied mixtures, the emission probability $b_j(\boldsymbol{y})$ for any one state j is calculated in the same way as for Equation (9.41), but although the mixture weights c_{jm} are state-specific, the $b_{jm}(\boldsymbol{y})$ terms will be the same for all states.

Using the new definition of the emission probability, re-estimation formulae can be derived for the mean $\boldsymbol{\mu}_m$ and covariance matrix $\boldsymbol{\Sigma}_m$ of the m^{th} component (the re-estimation of the mixture weights c_{jm} is unchanged). For example, tied-mixture versions of the Baum–Welch formulae in Equations (9.45) and (9.46) are as follows:

$$\bar{\boldsymbol{\mu}}_m = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\sum_{j=1}^{N}\gamma_{jm}(t,e)\boldsymbol{y}_{te}}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\sum_{j=1}^{N}\gamma_{jm}(t,e)}, \tag{9.50}$$

Ex.1019 / Page 54 of 102
AMAZON.COM, INC.

$$\bar{\Sigma}_m = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\sum_{j=1}^{N}\gamma_{jm}(t,e)(\boldsymbol{y}_{te}-\bar{\mu}_m)(\boldsymbol{y}_{te}-\bar{\mu}_m)^T}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\sum_{j=1}^{N}\gamma_{jm}(t,e)} . \quad (9.51)$$

Thus the only difference from the original (untied) mixture-distribution re-estimation formulae is that the contributions are now summed over all states as well as over all frames of all examples.

Tied mixtures have been used with some success to address the problems associated with training a large number of model parameters from a limited quantity of training data. However, although any practical system will have some model states which share many similarities, there will obviously be others which are quite different, and this characteristic will be reflected in the mixture weights for the different states. Thus rather more parameters are being tied together than is necessary, and such extensive tying may not be desirable for maximum discrimination. It is important to note that tied mixtures are just one example of the much more general concept of **parameter tying**, whereby any parameters of any model states can be tied together and the only effect on the re-estimation formulae is in the nature of the summations and in the indexing of the model parameters. The ability to tie together the parameters of HMM states is a significant factor in the success of current large-vocabulary speech recognition systems, and this use of tying is explained in Chapter 12.

9.11 EXTENSION OF STOCHASTIC MODELS TO WORD SEQUENCES

In the same way as was described for the dynamic programming methods in Chapter 8, HMMs extend easily to connected sequences of words. For recognition the word sequences can be represented by a higher-level model in which the states correspond to whole words, and the transition probabilities are the language-model probabilities (recognition using a language model will be discussed in Chapter 12).

In the case of recognition of isolated words we were not interested in the state sequences as such, but only in the likelihood of each word model emitting the observed feature vectors. When applying HMMs to connected words, however, we need to know the most likely sequence of words, so at the word level the Viterbi algorithm is necessary. The word boundary procedure is then exactly analogous to that described in Section 8.10, making use of the back-pointers to determine the word sequences.

The HMM training algorithms can also be used when the training data are spoken as natural connected sequences of words. It is not generally necessary to segment the data into the individual words prior to training. Instead, an embedded **training** approach can be used, whereby a composite model is obtained for the whole utterance by concatenating the required sequence of word models. This concatenation is very easy if special non-emitting initial and final states are used for the individual models, as it simply involves linking the final state of one word to the initial state of the next. The parameters of the composite model are trained

Ex.1019 / Page 55 of 102
AMAZON.COM, INC.

using the same procedure that would be carried out if this composite model represented a single word. If a state occurs more than once in the composite model (i.e. if the utterance contains more than one example of any particular word), all occurrences of that state will contribute to the parameter re-estimation. Provided that each word is spoken in a variety of different contexts, embedded training is very successful (at least for a constrained left-to-right model structure), even with the simplest 'flat' initialization procedure of setting the parameters of all models to the same values. The ability of the HMM training framework to automatically find the patterns in the data to associate with individual models is fundamental to the successful use of HMMs for substantial recognition tasks.

9.12 IMPLEMENTING PROBABILITY CALCULATIONS

The calculation of the forward and backward probabilities for sequences of feature vectors involves multiplication of a very large number of probability components, the majority of which are much less than 1. The results will in general have very low values, and mostly will be smaller than the minimum size of floating point number that can be held in any normal computer.

One solution to the number range problem is to check the probabilities at each stage of the recursion, and to multiply them by a scale factor that will bring the numbers back into the centre of the available range. However, scale factors must be noted and taken into account in estimating the relative likelihoods that each frame of feature vectors has been generated by each word model.

An alternative way of avoiding problems with numerical underflow is to represent all probabilities in logarithmic form, so that no explicit scaling is necessary. The following sections will discuss the implementation of HMM probability calculations using logarithms of probabilities.

9.12.1 Using the Viterbi algorithm with probabilities in logarithmic form

Because the logarithmic function is monotonic and increasing, the task of maximizing a probability can be achieved by maximizing its logarithm, and the main Viterbi probability calculation given in Equation (9.11) can therefore be replaced by:

$$\hat{\alpha}_j^L(t) = \max_{\text{over } i}\left(\hat{\alpha}_i^L(t-1) + a_{ij}^L\right) + b_j^L(\boldsymbol{y}_t) \ , \qquad (9.52)$$

where $\hat{\alpha}_j^L(t)$ is used for $\log(\hat{\alpha}_j(t))$, a_{ij}^L for $\log(a_{ij})$ and $b_j^L(\boldsymbol{y}_t)$ for $\log(b_j(\boldsymbol{y}_t))$.

When using discrete emission p.d.f.s with vector quantization, the calculation of Equation (9.52) is very straightforward and can easily be made very efficient: the quantities $\log(a_{ij})$ and $\log(b_j(\boldsymbol{y}_t))$ are fixed for given values of i, j and $\boldsymbol{y}_t$, and hence the logarithms need to be calculated just once and stored ready for use as required. The dynamic programming algorithm then only involves summations and comparisons, with no multiplications or logarithmic functions.

If normal distributions are used for the emission p.d.f.s, we must take the logarithm of the expression for the emission probability, but this is also

straightforward. For example, in the case of uncorrelated normal distributions for the individual features, taking logarithms[4] of Equation (9.33) gives:

$$b_j^L(\boldsymbol{y}_t) = \log(b_j(\boldsymbol{y}_t)) = -\frac{K}{2}\log(2\pi) - \sum_{k=1}^{K}\log(\sigma_{jk}) - \frac{1}{2}\sum_{k=1}^{K}\left(\frac{y_{kt}-\mu_{jk}}{\sigma_{jk}}\right)^2 \quad (9.53)$$

where we are now taking into account the fact that the observations are time-dependent, and are using the symbol y_{kt} to denote the k^{th} feature at the t^{th} frame.

Comparing Equation (9.53) with (9.33), it can be seen that use of logarithms has eliminated the need for the observation-dependent exponential operation, while the logarithmic terms are independent of the observed feature values and so can be pre-computed. Thus, while the computational load when using normal distributions is somewhat greater than for discrete emission p.d.f.s, the use of logarithms leads to a considerable computational saving as well as solving the number range problem.

9.12.2 Adding probabilities when they are in logarithmic form

When calculating emission probabilities using Gaussian mixture distributions, and for all calculations of forward and backward probabilities in Baum–Welch re-estimation, probabilities must be summed as well as multiplied and so the use of logarithms is more complicated. If we consider two probabilities, A and B, the task is to compute $\log(A + B)$ given $\log(A)$ and $\log(B)$. In theory, we could exponentiate both $\log(A)$ and $\log(B)$, add them and take the logarithm. However, aside from the computational issues, the exponential operation puts the probabilities back onto a linear scale and so presents problems for the wide range of probabilities that may be encountered. This difficulty can be addressed by first rewriting $\log(A + B)$ thus:

$$\log(A+B) = \log(A(1+B/A)) = \log(A) + \log(1+B/A)\,. \quad (9.54)$$

Assume that we have ordered the probabilities such that $A \geq B$. The issue is now one of evaluating the ratio B/A, which can be no greater than 1 and therefore the calculation will only present problems if this ratio is smaller than the smallest number which can be represented in the computer. This situation can only arise if B is so much smaller than A that it can safely be ignored by setting $\log(A + B) = \log(A)$. A procedure for finding $\log(A + B)$ is therefore as follows:

1. If $\log(B) > \log(A)$ then transpose $\log(A)$ and $\log(B)$.
2. Find $\log(B/A)$ by forming $\log(B) - \log(A)$. Store this value in C.
3. If $C <$ a suitable threshold, set $C = 0$.
4. Otherwise $C = \log(1 + \exp(C))$.
5. Add C to $\log(A)$.

The threshold in step 3 is used to prevent underflow when taking the exponential in step 4. The smallest value to which this threshold can be set is the logarithm of the smallest number that can be represented in the computer.

The procedure described above for performing probability calculations in logarithmic form is effective and widely used. However, whenever there is the need

[4] When using normal distributions, the calculations are simplest if natural logarithms are used, and the use of natural logarithms has been assumed in Equation (9.53).

Ex.1019 / Page 57 of 102
AMAZON.COM, INC.