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 kth feature of $\boldsymbol{y}$, and μ_{jk} and σ_{jk} are the mean and standard deviation of the distribution of the kth 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.

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 $\boldsymbol{\mu}_j$ and the covariance matrix $\boldsymbol{\Sigma}_j$ associated with state j are given by:

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

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

where $\boldsymbol{y}_{te}$ is the feature vector for the tth frame of the eth 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 $\bar{\boldsymbol{\mu}}_j$. In practice, when accumulating the contributions for covariance re-estimation it is easier to use the current estimate $\boldsymbol{\mu}_j$ instead of the (yet to be computed) new value $\bar{\boldsymbol{\mu}}_j$. 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:

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

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

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

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

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 mth component for state j is:

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

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 mth 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 mth 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}:

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

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 $\gamma_{jm}(t,e)$:

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

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

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 $c_{jm}b_{jm}(\boldsymbol{y}_{te})$. (The total state occupation probability $\gamma_j(t,e)$ is replaced by the emission probability $b_j(\boldsymbol{y}_{te})$.)

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**

***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 mth 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:

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

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

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

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 mth 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:

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

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

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

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:

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

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

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 $\log(b_j(y_t))$ are fixed for given values of i, j and 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 kth feature at the tth 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).

to add two probabilities, one exponential and one logarithmic operation are usually required. These operations can be avoided by using a method which allows the numbers to be added while in their logarithmic form[5]. Considering step 4 in the sequence of calculations described above, both the exponential and the logarithmic operation can be avoided by using a pre-computed look-up table to store the values of $\log(1 + B/A)$ in terms of $\log(B/A)$. Thus steps 3 and 4 can be replaced by a single table look-up operation, with $\log(B/A)$ as input (i.e. the value already stored in C at step 2). The output is $\log(1 + B/A)$, which can again be stored as the new value of C. Moderate accuracy in the value of $\log(A + B)$ can be achieved with a small look-up table. For example, a 1% accuracy for $A + B$ enables values of B/A of less than 0·01 to be ignored, and the look-up table for the larger values of B/A only needs entries for 115 equally spaced values of $\log(B/A)$.

If the above method is implemented using a suitable scale factor for the logarithms, it is then even possible to make all the probability calculations for recognition and parameter estimation using integer arithmetic on logarithmically coded numbers. No multiplications would be required with the VQ method, and no exponential functions would be needed when using Gaussian distributions. The 1% error proposed above should have very little effect on the re-estimation, but the error could easily be reduced if necessary by using a larger look-up table.

9.13 RELATIONSHIP BETWEEN DTW AND A SIMPLE HMM

It is interesting to compare the Markov probability calculation with the cumulative distance formula for a simple asymmetric dynamic programming algorithm in which each input frame occurs exactly once in the distance calculation. If the DP uses a squared Euclidean distance metric, the recognition process can be regarded as a special case of HMM Viterbi decoding, in which the word models have one state per template frame, and the features are assumed to be normally distributed with unit variance.

To clarify this relationship, we will return to the Viterbi calculation using logarithms of probabilities, given in Equation (9.52). The value of $b_j^L(\boldsymbol{y}_t)$ according to an uncorrelated normal distribution is given by Equation (9.53), where we are now assuming that $\sigma_{jk} = 1$ for all states j and for all features k. Hence

$$\sum_{k=1}^{K} \log(\sigma_{jk}) = 0 ,$$

and recursive calculation of $\hat{\alpha}_j^L(t)$ simplifies to:

$$\hat{\alpha}_j^L(t) = \max_{\text{over } i}\left(\hat{\alpha}_i^L(t-1) + a_{ij}^L\right) - \frac{K}{2}\log(2\pi) - \frac{1}{2}\sum_{k=1}^{K}\left(y_{kt} - \mu_{jk}\right)^2 . \qquad (9.55)$$

The term $K/2 \log(2\pi)$ is a constant, which will scale the likelihood calculation but will not affect the choice of optimal state sequence. Thus the only quantities that need be considered at each frame are the logarithms of the transition probabilities

[5] This method was described by Kingsbury and Rayner (1971) for a completely different application.

and the square of the Euclidean distance between the observed features at time t and the means for model state j.

As maximizing $\hat{\alpha}_j^L(t)$ is equivalent to minimizing $-\hat{\alpha}_j^L(t)$, this recognition task can be regarded as one of minimizing a distance comprising $-a_{ij}^L$, the negative logarithm of the transition probability (which must itself be positive), *plus* the squared Euclidean distance of the features from their model mean values. Thus we have a simple DP algorithm in which the $-a_{ij}^L$ terms are interpreted as timescale distortion penalties. Where only slopes of 0, 1, and 2 are permitted, as is the case for the HMM in Figure 9.1, the time distortion penalties for other values of slope are –log(0), and are therefore infinite.

9.14 STATE DURATIONAL CHARACTERISTICS OF HMMS

The probability of a model staying in the same state, i, for successive frames is determined only by the transition probability, a_{ii}. The expected number of frames it will stay in state i is $1/(1-a_{ii})$, so a value of $a_{ii} = 0{\cdot}9$ would be suitable for using one state to model, for example, a steady fricative sound whose expected duration is 10 frames. Although the expected total duration in state i in this case is 10 frames, the most likely duration is only one frame, with a probability of 0·1. The probabilities for longer durations decrease exponentially, as shown in trace (i) of Figure 9.2(b). This distribution is often referred to as a **geometric distribution** because the probabilities for successive numbers of frames form a geometric progression.

For any state representing a particular phonetic event, this type of duration distribution is obviously not sensible. For any such event there will be a most probable duration, with reducing probability for both shorter and longer durations. If many more states are available, the durational characteristics of the model can be improved, but only if a long steady region is modelled by a sequence of states with very similar feature p.d.f.s and the total likelihood method is used to calculate the word probability. For example, consider a group of four identical states with a repeat probability of 0·6, as shown in trace (ii) of Figure 9.2. The expected duration for the group is 10 frames, as it is for the single state shown in trace (i). However, in the case of the group of states, the variation of probability with duration is much more appropriate for speech sounds within a word. This more realistic distribution arises because, while there is only one possible way of going through the states in the minimum number of frames, there are more possible paths for longer frame sequences. However, the improved shape of duration distribution given by this state-splitting approach relies on using total likelihood probability calculations, whereas the Viterbi algorithm is generally used for recognition.

A simple method which can be used with the Viterbi algorithm involves merely imposing a minimum and a maximum duration on state occupancy. Such duration constraints can be achieved with an easy modification to the recognition algorithm, and can give worthwhile performance benefits. Many other methods have been proposed for improving the duration characteristics of HMMs, including some that model duration distributions of each state explicitly. These methods generally give greater benefits than simple duration constraints, but at the expense of more computation and some increase in the number of model parameters.

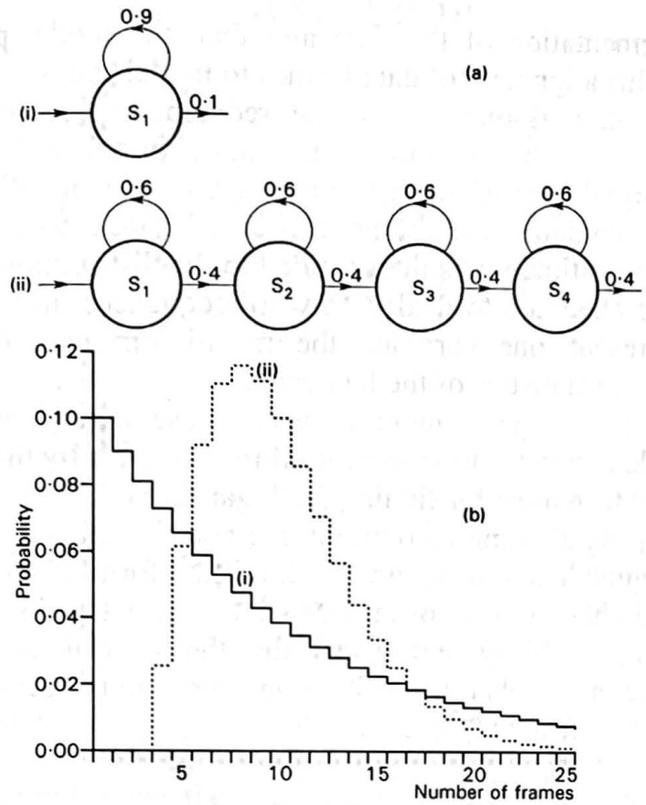


Figure 9.2 **(a)** Two arrangements of states, each with an expected occupation time of 10 frames. **(b)** Probability of occupancy of groups of states in the model sections shown in (a).

CHAPTER 9 SUMMARY

- The performance of pattern-matching speech recognizers is improved by representing typical characteristics of speech patterns in a way that also takes account of variability, which can be achieved by using a stochastic model of each word. Hidden Markov models (HMMs) represent each word as a sequence of states, with transition probabilities between each state and its permitted successors, and probability distributions defining the expected observed features for each state. A recursive formula can be used to calculate the probability that each word model will produce the observed data. The model with the highest probability is assumed to represent the correct word.
- Computation can be saved by using the Viterbi dynamic programming algorithm to calculate the probability of producing the data from only the most likely path through the states. This probability will always be less than the true probability, but the effect on recognition performance is usually very small and the Viterbi algorithm is generally adopted for HMM recognition.
- For each word model the transition probabilities and the probability distributions of the feature vectors can be found by the Baum–Welch re-estimation process. This process iteratively refines initial guesses to improve the model's representation of a set of training examples of the word, taking into account all possible paths through the states of the model.
- An alternative approach to estimating model parameters is to use a Viterbi training procedure, in which the initial guesses are used to find the most likely

state-level segmentation of the data and then the model parameters are re-estimated for this alignment of data frames to model states.

- Vector quantization is one method for reducing the set of possible feature vectors to a number for which robust training is possible. A parametric model, such as the normal (Gaussian) distribution or, more generally, a weighted sum (mixture) of Gaussian distributions, can also be used to describe the feature statistics. The re-estimation is then applied to the distribution parameters.
- HMMs can be extended to deal with word sequences, in which each state of the model represents one word, and the transition probabilities are determined by word sequence statistics of the language.
- One way of overcoming scaling problems because of very small numbers in the probability calculations is to represent all the numbers by their logarithms, and to use a special technique for finding the logarithm of the sum of two numbers.
- The dynamic programming recognition method described in Chapter 8 can be shown to be equivalent to using a very simplified form of HMM.
- The durational characteristic of an HMM state is determined only by the self-loop transition probability, and is such that the most likely duration is always only one frame and probabilities for longer durations decrease exponentially, so forming a geometric progression.

CHAPTER 9 EXERCISES

E9.1 What is the significance of the word 'hidden' in hidden Markov models?

E9.2 Why is it not necessary to explicitly consider all possible state sequences when calculating the probability of an HMM generating observed data?

E9.3 What is the essential difference between the Viterbi algorithm and the total likelihood method when calculating the probability of a word model generating observed data? What practical advantages can be gained by using the Viterbi algorithm for recognition?

E9.4 How can the form of an HMM be constrained by choice of initial parameters provided for re-estimation?

E9.5 What is the purpose of the 'vector quantization' sometimes used in HMMs?

E9.6 What are the benefits of using normal distributions to model feature statistics for HMMs? What are the limitations of simple normal distributions and how can these be overcome?

E9.7 How do the calculations required for Viterbi training differ from those for Baum–Welch re-estimation?

E9.8 What are the practical difficulties associated with implementing forward and backward probability calculations? What solutions are usually adopted?

E9.9 How can a simple HMM be interpreted as equivalent to a DTW recognizer?

E9.10 Why are the state durational characteristics of HMMs not very appropriate for modelling speech? What are the effects on duration characteristics if a single state is replaced by a sequence of several identical states?

CHAPTER 10

Introduction to Front-end Analysis for Automatic Speech Recognition

10.1 INTRODUCTION

The term "front-end analysis" refers to the first stage of ASR, whereby the input acoustic signal is converted to a sequence of acoustic **feature vectors**. As explained in Section 8.3, the short-term spectrum provides a convenient way of capturing the acoustic consequences of phonetic events. Ideally the method of front-end analysis should preserve all the perceptually important information for making phonetic distinctions, while not being sensitive to acoustic variations that are irrelevant phonetically. As a general policy for ASR, it seems desirable not to use features of the acoustic signal that are not used by human listeners, even if they are reliably present in human productions, because they may be distorted by the acoustic environment or electrical transmission path without causing the perceived speech quality to be impaired. Over the years many different front-ends have been tried, for use first with DTW recognizers and, more recently, with HMM systems. These front-ends vary in the extent to which they incorporate knowledge about human auditory perception, but currently the most successful analysis methods include at least some of the known properties of perception. These successful methods are, however, also characterized by a compatibility with the mathematical techniques that are generally used in HMM recognizers (as will be explained later). In this chapter we will introduce various aspects of front-end analysis for ASR.

10.2 PRE-EMPHASIS

The spectrum of voiced speech is characterized by a downward trend, whereby frequencies in the upper part of the spectrum are attenuated at about 6 dB/octave. This downward trend is due to a combination of the typical –12 dB/octave slope of the glottal source spectrum with the +6 dB/octave lift given by the radiation effect due to the lips (see Chapter 2). For the purpose of front-end analysis, it is common to compensate by applying a **pre-emphasis** of 6 dB/octave so that the analysed signal has a roughly flat spectral trend. This pre-emphasis is easily applied to the speech signal as the first processing stage. Although the above argument for pre-emphasis only applies to voiced regions, in practice it is usually applied throughout without causing any obvious problems for the analysis of voiceless regions.

10.3 FRAMES AND WINDOWING

Due to physical constraints, the vocal tract shape generally changes fairly slowly with time and tends to be fairly constant over short intervals (around 10–20 ms). A

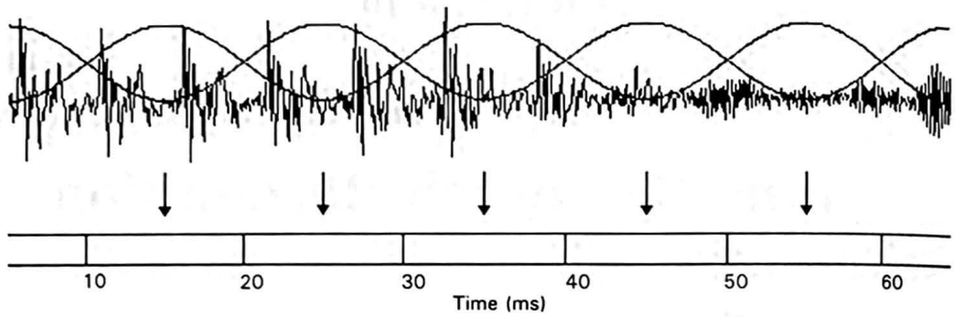


Figure 10.1 Analysis of a speech signal into a sequence of frames. This example shows a 20 ms Hanning window applied at 10 ms intervals to give a frame rate of 100 frames/s.

reasonable approximation is therefore to analyse the speech signal into a sequence of **frames**, where each frame is represented by a single feature vector describing the average spectrum for a short time interval.

Prior to any frequency analysis, each section of signal is multiplied by a tapered **window** (usually a **Hamming** or **Hanning** window). This type of windowing is necessary to reduce any discontinuities at the edges of the selected region, which would otherwise cause problems for the subsequent frequency analysis by introducing spurious high-frequency components into the spectrum. The length of each analysis window must be short enough to give the required time resolution, but on the other hand it cannot be too short if it is to provide adequate frequency resolution. In addition, because the analysis is normally performed at a fixed time interval, during voiced speech the window must be long enough so that it is not sensitive to exact position relative to the glottal cycle (i.e. there needs to always be at least one complete glottal cycle in the main part of the window). Long windows also have the advantage of smoothing out some of the random temporal variation that occurs in unvoiced sounds such as fricatives, but at the expense of blurring rapid events such as the releases of stop consonants. A common compromise is to use a 20–25 ms window applied at 10 ms intervals (giving a **frame rate** of 100 frames/s and an overlap between adjacent windows of about 50%), as shown in Figure 10.1.

10.4 FILTER BANKS, FOURIER ANALYSIS AND THE MEL SCALE

In Section 8.3 we introduced a speech signal representation using a filter bank with channels whose bandwidth and spacing increase with frequency (motivated by psychophysical studies of the frequency resolving power of the human ear). A convenient implementation of filter-bank analysis involves applying a Fourier transform. The output of the Fourier analysis will usually be at a finer frequency resolution than is required, especially at high frequencies. Thus the Fourier magnitudes are summed into a smaller number of channels, whose bandwidth and spacing conform to a perceptual scale such as the Bark or mel scale (see Section 3.5). Typically no more than 20 such channels are used for speech with a 4 kHz bandwidth, with a few additional channels being needed for higher-bandwidth signals. As already explained in Section 8.3, it is advantageous for the filter-bank output to represent power logarithmically, which reflects the phonetic

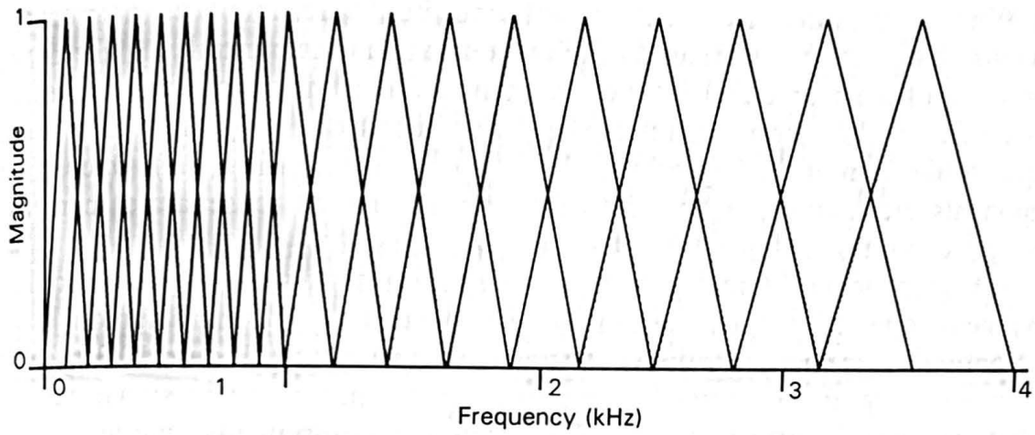


Figure 10.2 Triangular filters of the type suggested by Davis and Mermelstein (1980) for transforming the output of a Fourier transform onto a mel scale in both bandwidth and spacing.

significance of level variations and accords with evidence of a similar compressive non-linearity in auditory systems (see Chapter 3). A consequence of the logarithmic compression is that, when sampled from representative speech over a long period of time, the distribution of the energy in each of the channels tends to follow a Gaussian distribution, and is therefore compatible with any Gaussian assumptions that are made in the modelling.

Figure 10.2 shows a set of triangular 'filters' that can be used to compute a weighted sum of Fourier spectral components, so that the output of the process approximates to a mel scale. Here the centre frequencies of the filters are spaced equally (at intervals of 100 Hz) on a linear scale from 100 Hz to 1 kHz, and equally on a logarithmic scale above 1 kHz. (Other slightly different spacings are also often used.) Each filter's magnitude frequency response is triangular in shape, and is equal to unity at the centre frequency and decreases linearly to zero at the centre frequencies of the two adjacent filters. This configuration of mel filters, which is now very widely used in ASR, was suggested by Davis and Mermelstein (1980).

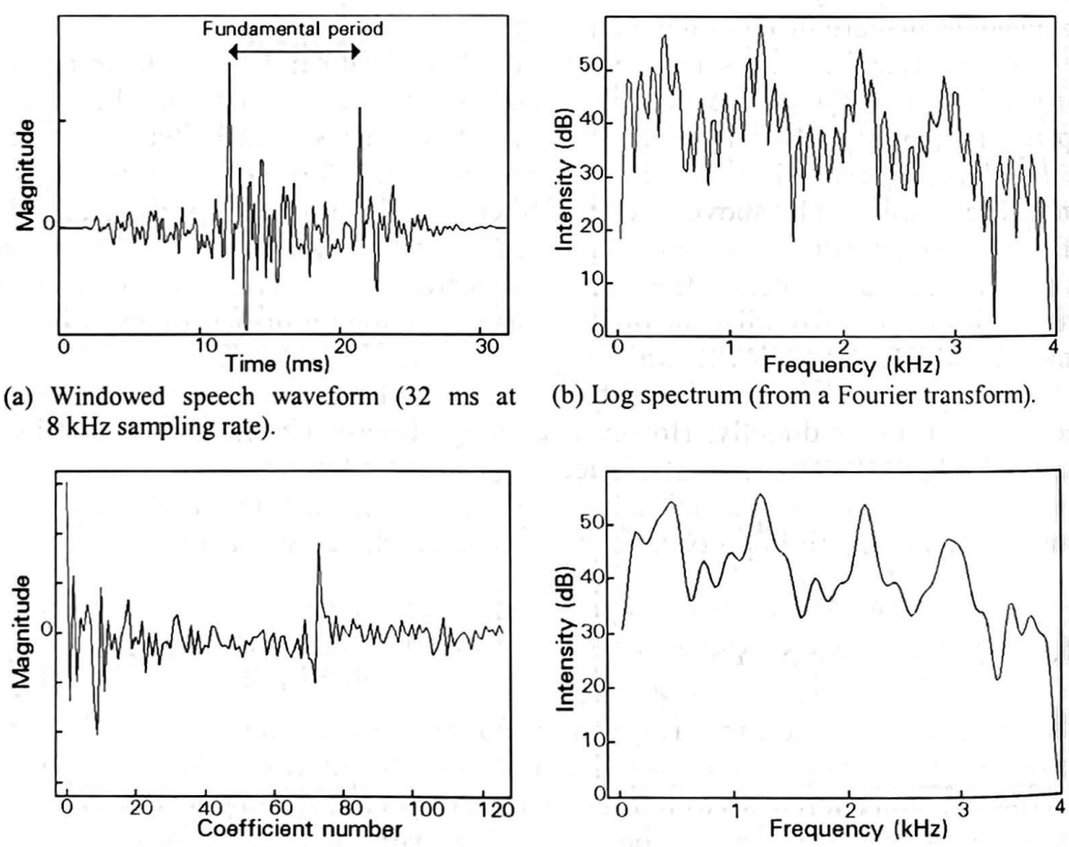
One option is to use the output of a filter-bank analysis to provide the recognition features directly. However, although filter-bank energies were widely used and achieved a fair amount of success as acoustic features in early recognition systems, there are substantial advantages to be gained by applying further transformations and this approach is the more usual choice nowadays.

10.5 CEPSTRAL ANALYSIS

The frequency resolution that is given by Fourier analysis applied to a 20–25 ms window of speech is generally sufficient to resolve the individual harmonics of the voiced excitation source, as well as showing the spectral shaping that is due to the vocal tract. Because the filtering operation of the vocal tract is the most influential factor in determining phonetic properties of speech sounds, it is desirable to separate out the excitation component from the filter component. The vocoders described in Chapter 4 are based on this principle. **Cepstral analysis** is another technique for estimating a separation of the source and filter components. Here the starting point is the observation that passing an excitation signal through a vocal-

tract filter to generate a speech signal can be represented as a process of **convolution** in the time domain, which is equivalent to multiplying the spectral magnitudes of the source and filter components. When the spectrum is represented logarithmically, these components are *additive*, because the logarithm of a product is equal to the sum of the logarithms ($\log(A \times B) = \log(A) + \log(B)$). Once the two components are additive, it is relatively straightforward to separate them using filtering techniques.

A typical logarithmic spectrum cross-section shows the rapidly oscillating component due to the excitation superimposed on a more gradual trend representing the influence of the vocal tract resonances (see Figure 10.3(b)). If we now imagine that this combined shape represents a time-domain signal, the rapid oscillations would correspond to high-frequency components, while the more gradual changes would be due to low-frequency components. If a Fourier transform were applied, the two components would therefore appear at opposite ends of the resulting spectrum. Thus by starting with the log magnitude spectrum and computing a Fourier transform, to obtain the so-called **cepstrum** (an anagram of "spectrum"), the excitation is effectively separated from the vocal-tract filtering, as shown in Figure 10.3(c). In fact, because the log magnitude spectrum is a symmetric function, the Fourier transform can be conveniently simplified to a



(a) Windowed speech waveform (32 ms at 8 kHz sampling rate).

(b) Log spectrum (from a Fourier transform).

(c) Cepstrum computed from the log spectrum shown in (b).

(d) Log spectrum reconstructed from the first 40 cepstral coefficients in (c).

Figure 10.3 Analysing a section of speech waveform to obtain the cepstrum and then to reconstruct a cepstrally smoothed spectrum.

discrete **cosine transform (DCT)**. For a spectral representation comprising N channels with log magnitudes A_1 to A_N, the DCT can be computed as follows:

$$c_j = \sqrt{\frac{2}{N}} \sum_{i=1}^{N} A_i \cos\left(\frac{\pi\, j(i-0.5)}{N}\right) \quad \text{for } 0 \le j < N, \tag{10.1}$$

where c_j is the jth **cepstral coefficient**. When $j = 0$, Equation (10.1) simplifies so that c_0 is proportional to the mean of the individual log channel signals A_1 to A_N. The c_1 term reflects the balance between energy at low frequencies and energy at high frequencies. As j increases, c_j captures increasingly fine spectral detail: first overall spectrum shape, then general formant structure, then more detailed spectral structure between the formants and, at high values of j, the excitation structure. There is no simple relationship between the c_j terms and the formants. However, for periodic speech the effect of the excitation source tends to be seen as a clear 'spike' at the pitch period duration (see Figure 10.3(c)). Cepstral analysis is therefore one method that can be used to estimate fundamental frequency. For example, in Figure 10.3(c) the spike occurs at c_{73} which, for the sampling frequency of 8 kHz (i.e. a sample duration of 0.125 ms), corresponds to a fundamental period of $73 \times 0.125 = 9.125$ ms. This value can be seen to be roughly equal to the interval between the pitch pulses in Figure 10.3(a).

Although the cosine transformation given by Equation (10.1) ensures that the Euclidean distance in transformed space is exactly equal to the distance between the sets of untransformed channel signals, the information that is of phonetic significance becomes concentrated in the lower-order terms. Filtering the cepstrum (a process usually referred to as **liftering**) can be applied to remove certain components or alter the relative influence of the different components. A simple lifter is one which simply truncates the cepstral sequence, by giving a weight of one to the low coefficients (up to some specified index) and a weight of zero to all the higher coefficients. By setting the cut-off point to just below the coefficient corresponding to the pitch period, most of the influence of the fundamental period is effectively removed from the spectrum. This process is shown in Figure 10.3(d), in which the spectrum has been re-constructed (by a Fourier transform) from just the low-order cepstral coefficients. The resulting spectrum can be seen to be much smoother than the original and show the formant peaks more clearly. The lower the cut-off point is set, the more detail will be removed and the smoother the spectrum will be. The process of smoothing the spectrum by truncating the sequence of cepstral coefficients is often referred to as **cepstral smoothing**.

The effectiveness of cepstral analysis for separating out the fundamental-frequency component of a speech signal depends on the frequency of the fundamental relative to the frequencies of the formants. The method generally works best for adult male speech (as shown in the example in Figure 10.3). For typical female and children's speech both the pitch and formant frequencies are higher, but the pitch increases more relative to the formant frequencies and so the cepstrum gives a less clear separation of the excitation component. It is therefore more difficult to set a cut-off point for cepstral smoothing that removes the pitch influence without also removing useful information about the formant structure.

In addition to the beneficial effect of concentrating on the information that is of greatest phonetic significance, discarding the high-order cepstral coefficients

reduces the number of features, so less computation is needed for the pattern-matching process.

The cepstrum has another desirable property for use in speech recognition. For typical speech signals it is found that, in contrast with the original channel signals, the variation of the separate coefficients tends to be uncorrelated. As a consequence, when using HMMs with continuous-density probability distributions, full covariance matrices can be replaced by much simpler diagonal covariance matrices (see Section 9.8.1) without any great loss in performance. Using diagonal covariance matrices substantially reduces both the computational requirements and the number of parameters needed to represent each distribution.

Cepstral coefficients have the property that (ignoring coefficients that are associated with pitch) both the variance and average numerical values decrease as the coefficient index increases (see Figure 10.3(c)). A consequence for a DTW recognizer using a simple Euclidean distance metric is that the distance calculation is affected most by the lowest-order coefficients and the coefficients that are more related to formant structure tend to be given insufficient weight. A solution that has often been adopted is to apply a lifter with a weighting for each coefficient that acts to roughly equalize the variances for the different coefficients. The problem does not arise when using probability distributions in HMM systems, because the variance is accommodated in the probability calculations. The liftering is often still applied, however, because the effect of making the variances of all the features cover a similar range makes it easier to study model parameters and to place restrictions on variances as part of re-estimation (see Section 11.4.1).

As explained above, the c_0 coefficient is proportional to the mean of the log channel signals and therefore provides an indication of overall level for the speech frame. Sometimes c_0 is included in the feature set, but often it is discarded and replaced by a different energy measure that is derived from the true signal energy. The energy in each frame will depend on overall speaking level, but for identifying sounds the most relevant factor is the *relative* level for different frames in an utterance. Therefore, for those applications for which the whole utterance becomes available before recognition needs to start, the measured energy is often normalized with respect to the maximum energy found over all frames in the utterance. (See Section 8.3.2 for further discussion about measures of speech level.)

In order to retain the advantages of a perceptually motivated filter-bank analysis, for ASR the cosine transform is usually applied to the output of non-linearly spaced filter-bank channels (see Section 10.4 above). A popular choice is to use **mel-frequency cepstral coefficients (MFCCs)**, which are obtained by applying a DCT to the output of mel filters such as the ones shown in Figure 10.2. An acoustic representation using MFCCs is often simply referred to as a **mel cepstrum**. As explained above, it is generally advantageous to discard the higher-order coefficients. For example, with 8 kHz bandwidth speech, there might be 24 mel channels but only the first 12 MFCCs are generally used in the final feature set. Although the use of the non-linear filter-bank means that the cosine transform no longer gives a simple separation of the excitation from the vocal-tract filtering (and much of the excitation effect will usually have already been smoothed out by the mel averaging), the truncation of the cepstral sequence has a general spectral smoothing effect that is normally desirable because it tends to remove phonetically irrelevant detail.

10.6 ANALYSIS BASED ON LINEAR PREDICTION

An alternative to filter-bank methods for representing the short-term spectrum is to derive linear prediction (LP) coefficients (usually called LPC analysis because of its origin in linear predictive coding, see Chapter 4). In the past, mainly during the 1970s, many recognizers were built using LPC-derived features and these systems generally gave performance comparable with that obtained from recognizers using filter-bank methods. During the 1980s it became more popular to use LPC-derived cepstral coefficients rather than the LP coefficients themselves because, as in the case of the filter-bank representation, the addition of the cepstral transformation was found to improve recognition performance. A convenient method exists for computing cepstral coefficients directly from the LP coefficients. LP analysis has the advantage that it produces an estimate of the smoothed spectrum, with much of the influence of the excitation removed. However, there is less freedom to apply non-linear processing to combat noise than there is with a filter-bank front-end. In addition, LPC inherently gives uniform weighting to low- and high-frequency regions of the spectrum. A non-linear frequency scale can be incorporated, but complicates the analysis to a greater extent than when using filter-bank methods.

Perceptual linear prediction (PLP) (Hermansky, 1990) is one LP-based analysis method that successfully incorporates a non-linear frequency scale and other known properties from the psychophysics of hearing. In PLP analysis, a Fourier transform is first applied to compute the short-term power spectrum, and the perceptual properties are applied while the signal is represented in this filter-bank form. The spectrum is transformed to a Bark scale, and this spectrum is pre-emphasized by a function that approximates the sensitivity of human hearing at different frequencies (see Figure 3.5). The output is compressed to approximate the non-linear relationship between the intensity of a sound and its perceived loudness. The all-pole model of LPC is then used to give a smooth, compact approximation to the simulated auditory spectrum, and finally the LP parameters are usually transformed to cepstral coefficients for use as recognition features. Apart from the use of LPC to achieve spectral smoothing, PLP analysis is very similar to MFCC analysis, but with perceptual properties incorporated in a way that is more directly related to psychophysical results (see Table 10.1 for a comparison of the two methods). In recent years a number of recognition systems have used PLP-based cepstral coefficients as acoustic features, and experimental evidence suggests that overall they give performance that is comparable with that obtained using MFCCs.

Table 10.1 Comparison between the properties of PLP cepstral coefficients and typical MFCCs.

MFCCs	PLP cepstral coefficients
Cepstrum-based spectral smoothing	LPC-based spectral smoothing
6 dB/octave pre-emphasis applied to speech waveform	equal-loudness pre-emphasis applied to spectrum
triangular mel filters	critical-band filters
logarithmic amplitude compression	cube root amplitude compression

10.7 DYNAMIC FEATURES

In the HMM probability calculations (see Section 9.3), the probability of a given acoustic vector corresponding to a given state depends only on the current vector and the current state, and is otherwise independent of the sequence of acoustic vectors preceding and following the current vector and state. It is thus assumed that there is no dependency between the observations, other than through the underlying state sequence. In reality, however, an acoustic feature vector representing part of a speech signal is highly correlated with its neighbours. In fact, it is often the dynamic characteristics of the features that provide most information about phonetic properties of speech sounds (related to, for example, formant transitions or the closures and releases of stop consonants). These correlations can be captured to some extent by augmenting the original set of (**'static'**) acoustic features (such as MFCCs) with dynamic features that are a measure of the change in the static features. These dynamic features are often referred to as **time derivatives** or **deltas**. One way of computing the delta features is by simple differencing between the feature values for two frames either side of the current frame:

$$\Delta y_t = y_{t+D} - y_{t-D}, \tag{10.2}$$

where D represents the number of frames to offset either side of the current frame and thus controls the width of the window over which the differencing operation is carried out. Typically D is set to a value of 1 or 2.

Although time-difference features have been used successfully in many systems, they are sensitive to random fluctuations in the original static features and therefore tend to be 'noisy'. A more robust measure of local change is obtained by applying linear regression over a sequence of frames:

$$\Delta y_t = \frac{\sum_{\tau=1}^{D} \tau \left(y_{t+\tau} - y_{t-\tau} \right)}{2 \sum_{\tau=1}^{D} \tau^2} \tag{10.3}$$

With linear regression, a value of $D = 2$ is the usual choice for an analysis frame rate of 100 frames/s. This regression window of five frames (50 ms) is long enough to smooth out random fluctuations, yet short enough to capture local dynamics.

The delta features described above are first-order time derivatives, which can in turn be used to calculate second-order time derivatives (sometimes referred to as **delta-deltas**). Including first-order time derivative features usually gives a large gain in recognition performance, and adding second-order derivatives (which capture changes in the first-order dynamics) tends to give an additional but smaller improvement. The majority of current HMM systems incorporate first-order derivative features, most often applied to a basic feature set of MFCCs and an energy feature, and many also include second-order derivatives. Most of the benefit from derivative features is due to their ability to capture dynamic information. However, these features also have the useful property that they are not affected by any constant or slowly changing disturbances to the signal (such as linear filtering in microphone pre-amplifiers and on telephone channels, for example), provided that these distortions are additive in the feature domain (see Section 11.2).

IPR2023-00037
Apple EX1016 Page 69

10.8 CAPTURING THE PERCEPTUALLY RELEVANT INFORMATION

Both in Chapter 3 and at the beginning of the current chapter we explained the desirability of capturing properties of human phonetic perception in the front-end analysis for ASR. Analysis methods such as PLP take into account several known facts about the lower levels of human auditory processing. However, there is no attempt to model higher-level auditory processing or more specific properties of speech perception in any of the analysis methods that have been described above.

It is now well established that the frequencies of the speech formants, particularly the first and second, are vitally important phonetically. Relative formant amplitudes are much less important, and the detailed structure of the lower-level spectral regions between formants is of almost no consequence. There would therefore seem to be potential for better performance in ASR if these factors could be taken into account when designing acoustic analysis methods and distance metrics. Although auditory models have shown considerable promise for incorporating into systems for ASR (see Section 3.7), these types of features have not yet replaced more general spectral features such as MFCCs or PLP-cepstra as the preferred choice in HMM-based systems. It is possible that substantial changes in the design of the recognizers themselves will be required before it will be possible to gain the full benefit from incorporating auditory models. We will return to this issue in Chapter 16, when we will also discuss the prospects and issues for extracting and using formant information more explicitly in ASR.

10.9 GENERAL FEATURE TRANSFORMATIONS

The DCT is one orthogonal transformation that reduces the dimensionality of a filter-bank output by concentrating the most useful information into a small number of features. Other orthogonal transformations for data reduction include **principal components analysis (PCA)** and **linear discriminant analysis (LDA)**. PCA performs a linear transformation on an input feature set, to produce a different feature set of lower dimensionality in a way that maximizes the proportion of the total variance that is accounted for. LDA also applies a linear transformation on the input feature set, but here the transformation is chosen to maximize a measure of class separability, and hence to improve discrimination. In order to determine the transformation, this procedure requires each input feature vector to have first been associated with a single class. PCA and LDA are both general data-reduction techniques that can usefully be applied to reduce the dimensionality of any diverse feature set, including for example static spectral or cepstral features with first- and second-order time derivatives, or even the output of auditory models. Both PCA and LDA generate new feature sets that are uncorrelated, thus allowing diagonal covariance matrices to be used for HMM state emission p.d.f.s.

10.10 VARIABLE-FRAME-RATE ANALYSIS

It has been assumed so far that all the frames in an utterance are of equal importance when making a comparison with stored templates or models. However,

a slight difference of vowel quality, for example, may not affect the identity of a word, whereas formant transitions at vowel–consonant boundaries may be crucial in identifying the consonant. Because, for many consonants, such transitions are very rapid, they do not occupy many frames. Although the addition of time-derivative features increases the importance of matching the transition characteristics, still rapid transitions may make only a small contribution to the cumulative distance or probability, even when they are matched very badly. The vowels and steady-state parts of long consonants can, in contrast, make a large contribution overall even when they match fairly well on each frame.

To overcome this difficulty it is necessary to give more weight to parts of the signal that are changing rapidly, and less weight to long steady regions. One way that is sometimes used to achieve this effect is to perform the original acoustic analysis at a fairly high frame rate (e.g. 100–200 frames/s), but then to discard a variable proportion of the frames depending on the distance between consecutive pairs of frames. Thus all frames are retained in rapid transitions, but perhaps only one in five is kept in very steady long vowels. This **variable-frame-rate** analysis method is similar to the scheme described in Section 4.3.5 for efficient speech coding. In the case of speech analysis for ASR, not only is there a computational saving, but also the overall match of an input utterance to stored templates or models shows much greater relative sensitivity to mismatch in transition regions.

CHAPTER 10 SUMMARY

- When deriving features for speech recognition, input speech is often first pre-emphasized by 6 dB/octave, so that the signal for subsequent analysis has a roughly flat spectral trend. Speech is analysed into a sequence of frames: most usually a 20–25 ms tapered window is applied at 10 ms intervals.
- One popular method of representing the speech spectrum is to use a filter bank with triangular filters whose width and spacing follow a mel scale. To obtain features for ASR, the output of such a filter bank is often subjected to a cosine transform (so deriving mel-frequency cepstral coefficients: MFCCs). An alternative is to derive cepstral coefficients from perceptual linear prediction.
- The cosine transform causes the features to become largely decorrelated so that diagonal covariance matrices can be used in the HMMs. In addition, information of phonetic significance is concentrated in the lower-order terms, so a more efficient representation can be obtained with fewer features.
- ASR performance is often greatly improved by adding 'delta' (first-order time-derivative) features, which are usually computed for each frame by applying linear regression over a window of five frames centred on the current frame.

CHAPTER 10 EXERCISES

E10.1 Why is cepstral analysis a useful tool in speech processing?

E10.2 Explain the stages that are typically used to analyse a speech signal into MFCCs and their first- and second-order time derivatives.

E10.3 How are properties of auditory perception simulated in front-ends for ASR?

CHAPTER 12

Automatic Speech Recognition for Large Vocabularies

12.1 INTRODUCTION

The previous four chapters have concentrated on introducing underlying theory and algorithms for ASR, together with some of the techniques for using the algorithms successfully in real situations. The discussion so far has deliberately concentrated either specifically on distinguishing between a small number of different words or on more general methods irrespective of the particular recognition task. In this chapter, we consider issues relevant to systems for recognizing continuously spoken utterances using large vocabularies, which may be anywhere from a few thousand up to around 100,000 different words.

12.2 HISTORICAL PERSPECTIVE

One of the earliest major efforts aimed at large-vocabulary ASR was initiated during 1971 in the United States by the Advanced Research Projects Agency (ARPA), with funding for a five-year programme of research and development. The overall objective was to make significant progress in the field of speech understanding by developing several alternative systems. The specific goal was to achieve a level of performance that was expressed in terms of semantic errors (less than 10%) on a continuous speech recognition task with a total vocabulary size of at least 1,000 words but using constrained-language input.

Although isolated-word recognition using pattern-matching techniques had achieved some initial success by the time of this ARPA programme, it was not generally obvious then how to extend the approach to accommodate the contextual effects that were known to occur in continuous speech. Therefore most systems adopted what at that time was the more traditional approach, using two separate stages. The first stage began by detecting **phonetic features** (e.g. formant frequencies, energy in different frequency bands, etc.) that were known to be important for distinguishing different speech sounds. Rules were used to convert from the measured features to a hypothesized phonetic transcription, which usually included some alternatives. The second stage then converted this transcription to a recognized word sequence. Inevitably there would be errors in the initial phonetic transcription, but the hope was that these errors would be corrected by the higher-level post-processing. However, in practice the first stage was so error-prone that information was lost which could not be recovered later. As a consequence, all the systems using this **knowledge-based** approach gave disappointing performance. In fact, the only system to achieve the required level of performance used a completely different method, based on a systematic search of a large network of

states with strong syntactic constraints, and it was one of the early large-vocabulary speech recognition systems using HMMs. The system was developed (somewhat separately from the main ARPA projects) at CMU by Lowerre (1976) as a Ph.D. project, extending the earlier pioneering work on HMMs by Baker (1975).

The results of the 1970s ARPA programme, while disappointing in terms of achievements for the money invested, provide a convincing demonstration of the benefits of **data-driven** statistical pattern matching over knowledge-based methods. In particular, the principle of delayed decision making is crucial, as it allows the overall best solution to be found incorporating all constraints, including those on construction of individual words and on allowed word sequences. This principle is fundamental to the design of all modern large-vocabulary speech recognizers.

Concurrent with the ARPA projects, research was in progress at IBM on the use of statistical methods for ASR. Early work was published by Jelinek (1976), independently of the work being carried out at CMU during the same period by Baker (1975). Work at IBM continued with an emphasis on applying HMMs to large-vocabulary speech recognition, and in the early 1980s the group focused on developing a system for dictation of office correspondence. The resulting system, "Tangora", as described by Jelinek (1985), was a speaker-dependent, isolated-word, near-real-time recognizer with a 5,000-word vocabulary. Although this system required users to leave pauses between words, it established the principles underlying the use of HMMs for a large-vocabulary task. Since the mid-1980s, further developments in many laboratories have led to significant further progress, and systems are now able to recognize fluent, naturally spoken continuous speech with very large vocabularies. There are a variety of systems for **large-vocabulary continuous speech recognition** (**LVCSR**) in existence, both as commercial products and as research systems in laboratories. At present, the successful systems are all based on HMMs, usually incorporating many of the refinements described in Chapter 11, but also with components that are specific to demands imposed by the need to cope with large vocabularies.

12.3 SPEECH TRANSCRIPTION AND SPEECH UNDERSTANDING

Large-vocabulary speech recognition tasks fall into two quite distinct categories:

1. *Speech transcription*: The user wishes to know exactly what the speaker said, in the form that it would be transcribed by an audio typist to produce orthographic text. Such a system may be used for dictation, and for tasks such as producing transcripts of broadcast news programmes.
2. *Speech understanding*: The semantic content of the message is required, and any recognition errors do not matter provided that the meaning is not changed. In fact often the real requirement is for the system to perform the correct action, irrespective of what words are recognized. Speech understanding systems may involve an interactive dialogue between a person and a machine to retrieve information from some computerized database. Other uses include automatic information extraction, for example to summarize spoken reports or broadcasts.

The interactive nature of many speech-understanding tasks, together with the fact that the subject area is often restricted, means that the relevant vocabulary at

any one point can be much smaller than the total vocabulary that is needed for more general transcription tasks. However, in order to interpret meaning of utterances, more detailed syntactic and semantic analyses are necessary than are required when just transcribing the words that were spoken. The principles of large-vocabulary recognition using HMMs apply both to transcription and to understanding, but the way in which the recognizer output is used is rather different. The first, main part of this chapter concentrates on transcription, while the latter part of the chapter briefly describes the use of large-vocabulary ASR in speech understanding systems.

12.4 SPEECH TRANSCRIPTION

The input speech waveform (typically sampled at 16 kHz) is first analysed into a sequence of acoustic feature vectors such as MFCCs (see Chapter 10). A popular choice is the first 12 cepstral coefficients and an overall energy feature together with first and second time derivatives of these features, giving a 39-element vector.

Once the input speech has been analysed into a sequence of feature vectors, the recognition task is to find the most probable word sequence $\hat{W}$ given the observed vector sequence $\boldsymbol{Y}$. Revisiting Bayes' theorem (see Section 9.2), but applying it to the task of finding a word *sequence*, the most probable sequence can be derived from the probability $P(W|\boldsymbol{Y})$ of any one sequence W as follows:

$$\hat{W} = \arg\max_{W} P(W \mid \boldsymbol{Y}) = \arg\max_{W} \frac{P(\boldsymbol{Y} \mid W)P(W)}{P(\boldsymbol{Y})} = \arg\max_{W} P(\boldsymbol{Y} \mid W)P(W) \,. \quad (12.1)$$

Equation (12.1) states that the most likely word sequence is the one which maximizes the product of $P(\boldsymbol{Y}|W)$ and $P(W)$. The first term denotes the probability of observing vector sequence $\boldsymbol{Y}$ given the word sequence W, and is determined by an **acoustic model**. The second term represents the probability of observing word sequence W independently from the acoustic signal, and is determined by a **language model**. Chapter 9 focused on the task of calculating acoustic-model probabilities, which is fundamental to any speech recognition system based on statistical models. However, for all but the most simple of applications, the language-model probability is also a major factor in obtaining good performance: restrictions imposed by the language model can greatly reduce the number of different alternatives to be distinguished by the acoustic model. As with the acoustic model, the language model for LVCSR is usually a statistical model that is automatically trained on data. In the case of the language model, these data usually take the form of *text* material chosen to be representative of the recognition task.

Assuming that models have been trained, Figure 12.1 illustrates a framework for classifying an unknown utterance by computing $P(\boldsymbol{Y}|W)P(W)$. The language model postulates a word sequence ("ten pots" in this example[1]) and determines its probability $P(W)$. In order to calculate the acoustic-model probability $P(\boldsymbol{Y}|W)$, a

[1] The phrase "ten pots" will be used in this chapter to illustrate a variety of different points. This phrase was chosen to provide a simple example for which the phonetic and orthographic transcriptions are very similar. For convenience of notation, we will represent the vowel in "pots" with its orthographic transcription /o/ in place of the correct phonetic notation for southern British English /ɒ/.

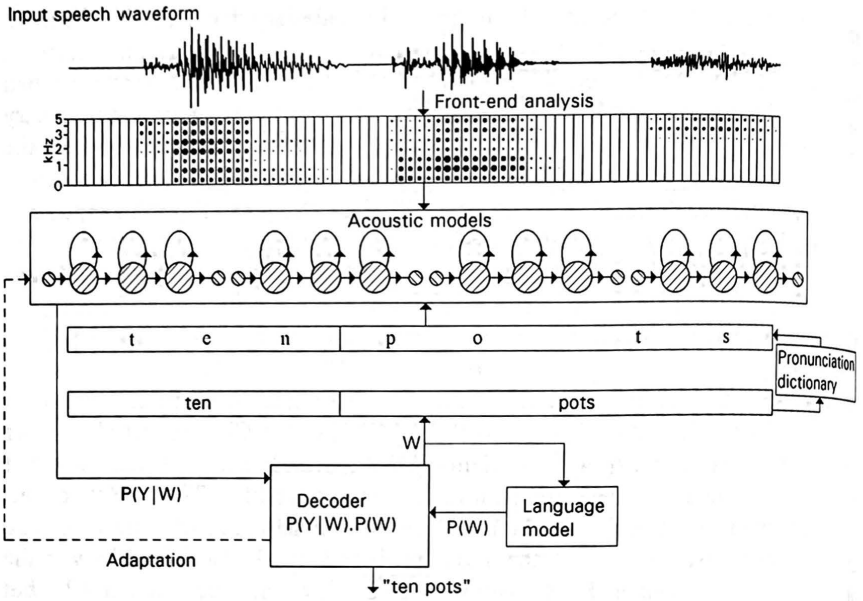


Figure 12.1 Framework for decoding a speech signal by computing the probability of a word sequence in terms of language-model and acoustic-model probabilities, shown for recognition of the phrase "ten pots". A simple filter-bank analysis is shown here for clarity of illustration, although in practice other features such as MFCCs would be used. Due to space limitations, only four of the seven models needed to represent the phone sequence in "ten pots" are shown.

composite model for the word sequence is generated. Rather than having a separate HMM for each word, the component models represent phone-size units and a pronunciation dictionary is used to specify the sequence of models for each word in the vocabulary. For any word sequence, the dictionary is used to look up the required sequence of phone models for each word, and these phone models are concatenated together to form the model for the word sequence. The probability of that model generating the observed acoustic sequence is calculated, and this probability is multiplied together with the language-model probability. In principle, this process can be repeated for all possible word sequences allowed by the language model, with the most likely sequence being selected as the recognizer output. In practice, decoding for LVCSR requires a very efficient search strategy for evaluating the vast number of different possibilities, as will be explained later.

12.5 CHALLENGES POSED BY LARGE VOCABULARIES

Issues for the design of large-vocabulary recognition systems include the following:

1. In continuous fluent speech, there are many instances when words cannot be distinguished based on acoustic information alone and it is necessary to rely on a language model for discrimination. Difficulties in making acoustic distinctions arise for two main reasons. Firstly, due to co-articulation between adjacent words, word boundaries are not usually apparent in the acoustic signal. In some cases, two utterances may be linguistically different but acoustically very similar

or even identical (as in the "grey day" versus "grade A" example given in Chapter 1). Secondly, the pronunciation of many words, particularly function words, can be so reduced that there is very little acoustic information at all.

2. The memory and computation requirements can become excessive. In recent years, advances in computer technology have greatly reduced the impact of this limitation. However, memory and computation are still influential, especially in determining the choice of search mechanism for use in decoding.
3. As the vocabulary size increases, it becomes increasingly difficult to provide enough representative examples of all the words, both as text to train the language model and as spoken examples to train the acoustic model.

Many of the design features of modern LVCSR systems are determined by the need to deal with these issues. The design of the acoustic model, the language model and the decoding operation are all crucial factors for the success of an LVCSR system. The following three sections describe each of these three components in turn.

12.6 ACOUSTIC MODELLING

Although some early systems used HMMs with discrete distributions for their emission p.d.f.s (e.g. Lee (1989)), current systems generally use fully continuous distributions or tied-mixture distributions, usually with diagonal covariance matrices. These latter types will be the focus of the explanation given here, which is based mainly on descriptions of the research system developed at Cambridge University (e.g. Young (1996)). This system is one of the most successful systems to date, but there are many other systems that have fairly similar structure and give broadly comparable performance, although they differ in various details.

The need to make the best use of any available acoustic training data has important consequences for the design of the acoustic-model component. With a large vocabulary, it is impractical to expect any one person to provide enough examples to train models for all the words from scratch, even if the system is intended for speaker-dependent operation. Therefore, a speaker-independent model set is used, at least to provide a starting point. Speaker-adaptation techniques are often used to improve performance for any one individual. Unsupervised adaptation may be performed using the recognizer output, as shown by the dotted data path in Figure 12.1. In addition, for a system to be used by one known person, that person can be required to speak some specific utterances, which can be used for supervised model adaptation before the person uses the system to perform any real task.

Even with several speakers to provide the data, it is not practical to train a separate model for each word in a large-vocabulary system. Even if it were practical, this approach would not make the best use of the data, as it does not take account of the fact that different words can share sub-components. Therefore large-vocabulary systems are based on **sub-word** models. The usual method, as shown in Figure 12.1, is to use models of phone-size units, with the sequence of phones for each word being specified in a pronunciation dictionary. Thus, the requirement for the training is to provide sufficient examples of all the phone-size units, and all the words in the vocabulary will not necessarily have occurred in the training data. In fact, provided suitable models are available, words can be added to the vocabulary at any time simply by extending the pronunciation dictionary.

12.6.1 Context-dependent phone modelling

As approximately 44 phonemes are needed to represent all English words, this number of models would be the minimum needed to build word models for English. However, the effects of co-articulation are such that the acoustic realization of any one phoneme can vary greatly with acoustic context. Therefore **context-dependent** HMMs are generally used, with different models for different phonetic contexts. Additional variation tends to arise because many speakers will, either consistently or occasionally, use word pronunciations that are different from those given in the dictionary. Although alternative pronunciations can be included in the dictionary, it is difficult to include every possible pronunciation and any that are not covered will need somehow to be accommodated in the chosen set of context-dependent HMMs.

The simplest and most popular approach is to use **triphones**, whereby every phone has a distinct HMM for every unique pair of left and right neighbours. For example, consider the word "ten". When spoken in isolation, this word could be represented by the sequence `sil t e n sil`, with the `sil` model being used for silence at the start and end. Using triphones, with the notation ${}_{x}y_{z}$ to denote phone y preceded by phone x and followed by phone z, the word would be modelled as

$$\mathtt{sil}\ {}_{\mathtt{sil}}\mathtt{t}_{\mathtt{e}}\ \ {}_{\mathtt{t}}\mathtt{e}_{\mathtt{n}}\ \ {}_{\mathtt{e}}\mathtt{n}_{\mathtt{sil}}\ \mathtt{sil}.$$

Now consider the phrase "ten pots", for which the triphone sequence would be

$$\mathtt{sil}\ {}_{\mathtt{sil}}\mathtt{t}_{\mathtt{e}}\ \ {}_{\mathtt{t}}\mathtt{e}_{\mathtt{n}}\ \ {}_{\mathtt{e}}\mathtt{n}_{\mathtt{p}}\ \ {}_{\mathtt{n}}\mathtt{p}_{\mathtt{o}}\ \ {}_{\mathtt{p}}\mathtt{o}_{\mathtt{t}}\ \ {}_{\mathtt{o}}\mathtt{t}_{\mathtt{s}}\ \ {}_{\mathtt{t}}\mathtt{s}_{\mathtt{sil}}\ \mathtt{sil}.$$

The two instances of the phone [t] are represented by different models because their contexts are different. Note that the triphone contexts span word boundaries, so that the first and last triphones used to represent a word depend on the preceding and following words respectively. For example, if the phrase were "ten dogs", the last triphone used to model "ten" would be ${}_{\mathtt{e}}\mathtt{n}_{\mathtt{d}}$ rather than ${}_{\mathtt{e}}\mathtt{n}_{\mathtt{p}}$. This use of **cross-word triphones** enables co-articulation effects across word boundaries to be accommodated, but creates complications for the decoding process as the sequence of HMMs used to represent any one word will depend on the following word.

The decoding task can be greatly simplified by using only **word-internal triphones**, whereby 'word boundary' acts as a context and so the sequence of HMMs is fixed for each word. Thus, in the above example the triphones ${}_{\mathtt{e}}\mathtt{n}_{\mathtt{p}}$ and ${}_{\mathtt{n}}\mathtt{p}_{\mathtt{o}}$ would be replaced by ${}_{\mathtt{e}}\mathtt{n}_{_}$ and ${}_{_}\mathtt{p}_{\mathtt{o}}$ respectively, with – being used to represent a word boundary. Early triphone systems were restricted to word-internal triphones, but the inability to model contextual effects across word boundaries is a serious disadvantage and current systems generally include cross-word context-dependent models. The consequences for decoding are explained in Section 12.8.

12.6.2 Training issues for context-dependent models

For a language with 44 different phones, the number of possible triphones is 44^3 = 85,184. In fact, phonotactic constraints are such that not all of these triphones can occur. However, an LVCSR system which includes cross-word triphones will still typically need around 60,000 triphones. This large number of possible triphones poses problems for training the models:

IPR2023-00037
Apple EX1016 Page 77

- The total number of parameters needed for the models is very large: it is usual to use three-state models with somewhere in the region of 10 mixture components to represent the output distribution for each state. This number of mixture components tends to be needed to represent the wide range of speakers (including a range of different accent types) who must be accommodated within a single model. Assuming that diagonal covariance matrices are used with 39-element acoustic feature vectors and 10 mixture components, each state would require around 790 parameters (39×10 means, 39×10 variances, and 10 mixture weights). Thus 60,000 three-state models would have a total of over 142 million parameters. Any feasible quantity of training data would not be large enough to train this number of parameters adequately.
- In any given set of training data, many triphones will inevitably not occur at all, especially if cross-word triphones are allowed (as it is very difficult to include all the phone combinations that might occur in all possible sequences of words). Thus some effective method is required for generating models for these **unseen triphones** that do not occur in the training data.

Similar issues have already been mentioned as difficulties with using whole-word models for large vocabularies. However, when using smaller model units that are meaningful in phonetic terms, it is easier to see how the problems can be reduced. The challenge is to balance the need for detail and specificity in the models against a requirement to have enough training data to obtain robust parameter estimates. To tackle the problem various different **smoothing** techniques have been used:

1. *Backing off*: When there are insufficient data to train a context-specific model, it is possible to **back off** and use a less-specific model for which more data are available. For example, one approach is to replace a triphone by the relevant **biphone**[2], representing the phone dependent on only one adjacent context, which may be either to the left or to the right. Given a choice between the left or the right biphone context, it is generally better to choose the right context as articulation tends to be anticipatory, such that following context has a greater influence than preceding context. If there are insufficient examples to train a biphone, it is possible to resort to the context-independent phone model, or **monophone**. The backing-off mechanism ensures that every model in the final system is adequately trained, but can result in only a relatively small number of full triphone models, so that several contexts are not modelled very accurately.
2. *Interpolation*: A greater degree of context dependency can be retained by **interpolating** the parameters of a context-specific (triphone) model with those of a less-specific model (such as a biphone or monophone), to give model parameters which represent some compromise between the two sets. Thus some of the context dependency of the original triphone models is preserved, while increasing their robustness by taking into account additional (less specific) data.
3. *Parameter sharing*: An alternative is to take all the triphones representing any one phone, apply some form of **clustering** procedure to group the individual

[2] Note that the term **biphone** is used to refer to a phone that is dependent upon a single context (either left or right), and that this unit is different from the **diphone** unit discussed in Chapter 5, which represents the region from the middle of one phone to the middle of the next.

models (or parts of models) into clusters with similar characteristics, and share the parameters between them. This sharing of parameters, often referred to as **parameter tying**, allows the data associated with similar states to be pooled to improve the robustness of the parameter estimates. This approach can retain a higher degree of context specificity than is possible with the first two methods.

Although both backing off and parameter interpolation have been used with some success, the greater power of more general parameter sharing to obtain a better compromise between accuracy and robustness is such that this method is now widely used in LVCSR. The method is described in more detail below.

12.6.3 Parameter tying

The technique of tying provides a general mechanism for sharing parameters between models or parts of models. One example is provided by the tied-mixture distributions introduced in Chapter 9, where the means and variances of each mixture component are tied across all model states. Smoothing the parameters of context-dependent models represents another application of tying. Here tying is usually applied to all model parameters for a subset of the triphones representing a phone. The aim is to tie together those models or states for which any differences in the acoustic realizations are not significant for phonetic discrimination.

Initial developments in parameter sharing between context-dependent models concentrated on clustering together triphone *models*, to give **generalized triphones**. However, this approach assumes that the degree of similarity between any two models is the same for all the states in the models. In fact, the different effects of left and right context are such that this assumption is rarely justified. For example, consider three triphones of /e/: $_{t}e_{n}$, $_{t}e_{p}$ and $_{k}e_{n}$. The first state of the $_{t}e_{n}$ and $_{t}e_{p}$ triphones can be expected to be very similar, while the last state of the $_{t}e_{n}$ and $_{k}e_{n}$ triphones will be similar. Thus tying at the state level offers much more flexibility to make the most effective use of the available data for training a set of models. We will now consider two important issues associated with the use of state tying: firstly the general procedure used to train the tied-state multiple-component mixture models (assuming that it is known which states to tie together), and secondly the choice of clustering method used to decide on the state groupings. The discussion focuses on state tying, but the principles apply in the same way when the tying is applied to complete models.

12.6.4 Training procedure

Careful design of the training procedure is essential to maximize the robustness and accuracy of the final set of tied-state context-dependent HMMs. When training sub-word models, it is not usual for the individual speech segments to have been identified and labelled in the available training data. In fact it is most likely that the data will have been transcribed as a sequence of words but not segmented at all. Rather than attempting to segment these data, they can be used directly for parameter estimation by adopting the **embedded training** approach described in Section 9.11. When using sub-word models, it is necessary first to construct a

model for each word from the sub-word units, before then constructing a model for the whole utterance from the individual words. Similarly to the procedure outlined in Section 12.4 for recognition, the pronunciation dictionary is used to look up the phone sequence required to represent each training utterance. A composite HMM is constructed by concatenating the appropriate sub-word models, and the relevant statistics for re-estimation are accumulated over all occurrences of each model.

Phone sequence constraints across different utterances are such that the embedded training method should generally be effective in associating appropriate speech frames with each model state, provided that in the early stages of training each model is used in a sufficient range of different contexts. For this situation it is even adequate to use the very simple 'flat' initialization of all the model parameters to identical values (see Section 9.9). It is usual to start with single-Gaussian distributions and train simple monophone models. Because there are very many examples of each one, these monophones can be trained very robustly, and provide a good basis for initializing the more specific context-dependent models. A typical procedure for training context-dependent models is illustrated in Figure 12.2, and summarized below:

1. A set of monophone HMMs, using single-Gaussian output distributions with diagonal covariance matrices, is created and trained.
2. All the training utterances are transcribed in terms of the complete set of possible triphones. For each triphone, an initial model is created by cloning the appropriate monophone. The transition probability matrix is typically not cloned, but is tied across all triphones of a phone. The triphone model parameters are re-estimated and the state occupancies, which represent the expected number of observations used to estimate the parameters of each triphone (see Section 9.5.2), are retained for later use.
3. For the set of triphones representing each phone, similar states are clustered together to create tied states (see Sections 12.6.5 and 12.6.6 for more explanation). As part of the state tying process, it is important to check that there are sufficient data associated with each tied state. This situation can be achieved by only allowing clusters for which the total state occupancy (i.e. the estimated 'count' of number of frames for which the state is occupied) exceeds a suitable threshold value (typically around 100). The parameters of the tied-state single-Gaussian models can then be re-estimated. The use of tying does not alter the form of the re-estimation formulae and can be made transparent to the re-estimation process if the data structures used to store the information are set up appropriately. Storage can be set up for accumulating the numerator and denominator for re-estimating each parameter of each tied state, with individual states simply pointing to the relevant storage.
4. Finally, multiple-component mixture models are trained using the iterative mixture splitting procedure explained in Section 9.10.4.

Delaying the introduction of the multiple-component Gaussians until the final stage has a number of advantages:

- The difficulties associated with training untied triphone mixture distributions are avoided, as multiple mixture components are only introduced once the model inventory has been set up to ensure adequate training data for each state.
- The state tying procedure is simplified because, by using single-Gaussian

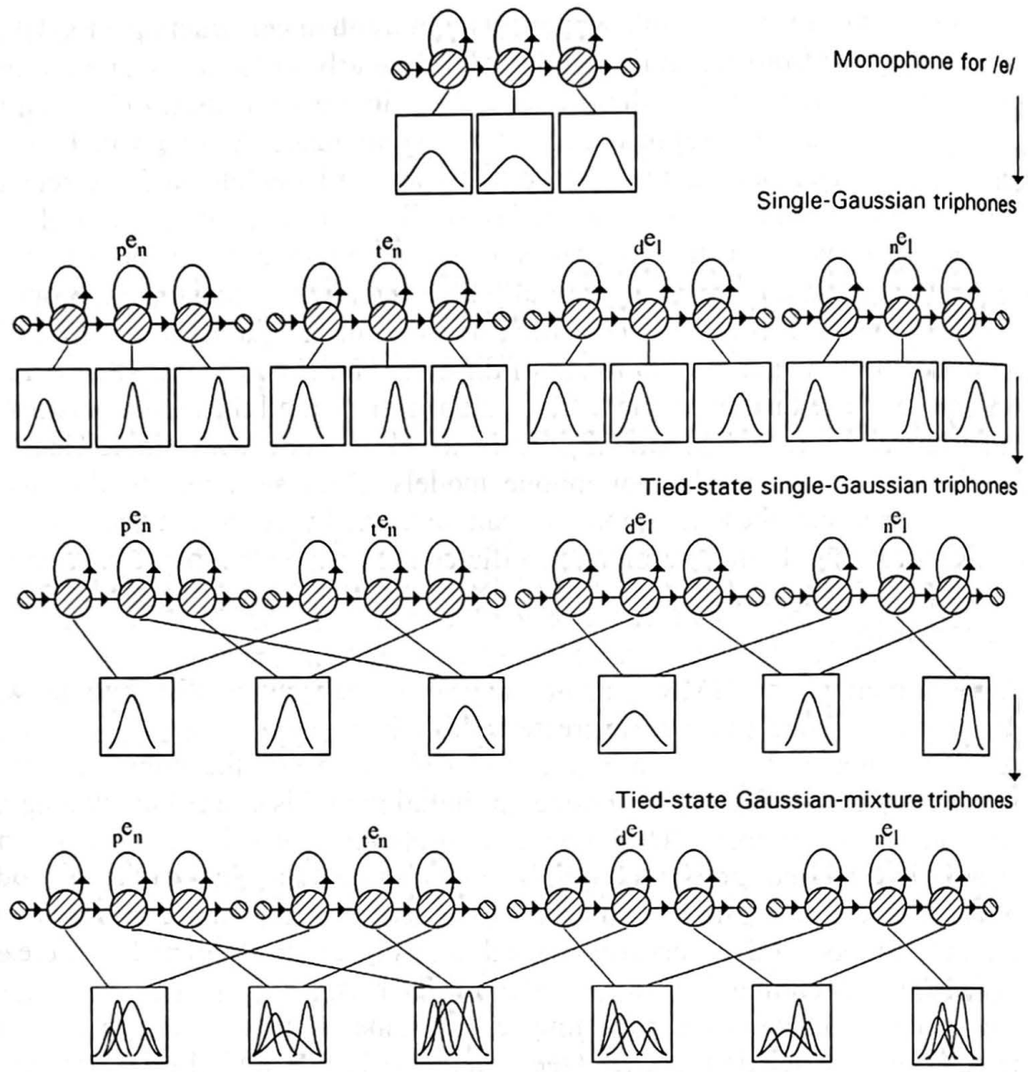


Figure 12.2 Sequence of stages for training tied-state Gaussian-mixture triphones, illustrated for a group of triphones representing the /e/ phoneme.

distributions, it is much easier to compute a similarity measure and to calculate the parameters of the combined states (see Section 12.6.6).

- By not introducing the mixture distributions at the monophone stage, the process avoids potential complications that could arise if the mixture components were used to accommodate contextual variation which would at a later stage be covered by the context-dependent models. It is generally better to accommodate contextual influences explicitly so that the predictable nature of these effects can be exploited as far as possible. The multiple mixture components are then needed mainly to allow for the fact that any one model represents data from a wide variety of different speakers.

In addition to the benefits in terms of robustness, computation and storage, state tying has the potential to lead to models with better discrimination. The potential advantages of sub-word over whole-word models in terms of discrimination power have already been mentioned. These arguments extend to the use of state tying. If the differences between the acoustic realizations associated with two different model states are simply a consequence of random variation, it is

detrimental for these differences to be included in the models. By combining them into a single model state, discrimination will be more focused on those regions of words containing the most useful acoustic cues. This benefit is dependent upon finding an appropriate method for determining which states to tie together.

12.6.5 Methods for clustering model parameters

Clustering methods for grouping together similar states can be divided into two general types. These methods can be used to cluster triphone states as follows:

1. *Bottom-up clustering*: Starting with a separate model for each individual triphone, similar states are merged together to form a single new model state. The merging process is continued until some threshold is reached which ensures that there are adequate data to train each new clustered state. This data-driven approach is often referred to as **agglomerative clustering**. The method should ensure that there are sufficient data to train every state in the final set, while keeping the models as context-specific as possible given the available training data. However, for any triphones that do not occur at all in the training data, it is still necessary to back off to more general models such as biphones or monophones.
2. *Top-down clustering*: Initially all triphones for a phoneme are grouped together and a hierarchical splitting procedure is used to progressively divide up the group according to the answers to binary yes/no questions about either the left or the right immediately adjacent phonetic context. The questions are arranged as a **phonetic decision tree**, and the division process starts at the root node of the tree and continues until all the leaf nodes have been reached. All the states clustered at each leaf node are then tied together. A tree is generated for each state of each phone. An example showing the use of a decision tree to cluster the centre state of some /e/ triphones is shown in Figure 12.3. The context questions in the tree may relate to specific phones (e.g. "Is the phone to the right /l/?"), or to broad phonetic classes (e.g. "Is the phone to the left a nasal?"). Using the tree, the correct tied state to use for any given context can be found by tracing the path down the tree until a leaf node is reached (see Figure 12.3).

The main advantage of the top-down approach to clustering is that a context-dependent model will be specified for *any* triphone context, even if that context did not occur in the training data. It is thus possible to build more accurate models for unseen triphones than can be achieved with the simple backing-off strategy, assuming that the questions in the tree are such that contexts are grouped appropriately. Although the tree could be constructed by hand based on phonetic knowledge, this approach does not work very well in practice, as it does not take into account the acoustic similarity of the triphones in the data. It is, however, possible to construct trees automatically by combining the use of phonetic questions with tests of acoustic similarity and a test for sufficient data to represent any new division. This automatic construction provides generalization to unseen contexts while maintaining accuracy and robustness in the acoustic models. A popular version of this effective technique for constructing phonetic decision trees is explained in more detail in the next section.

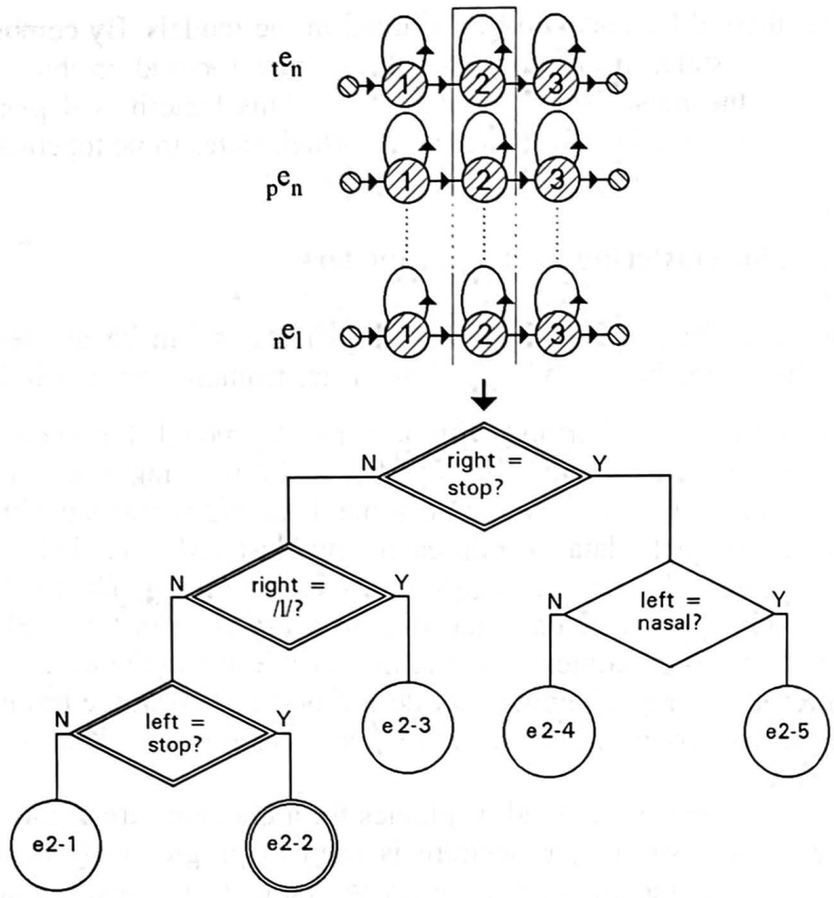


Figure 12.3 Example illustrating the use of a phonetic decision tree to cluster the centre state for a group of triphones of the /e/ phoneme. Each triphone is moved down the tree until a terminal 'leaf' node is reached (shown as circles), and a new model state is formed from the members of the cluster. The tree shown here is a very simple one intended simply as an illustration of the principles of clustering, and in any real application there will be many more questions before a terminal node is reached. The tree can then be used to find the appropriate clustered state for any given context. As an example, the route down this simple tree is shown (using double lines round the chosen decision boxes and the final leaf node) for the context of a preceding /t/ and a following /n/. In this context the clustered state e2-2 will be used.

12.6.6 Constructing phonetic decision trees

First, linguistic knowledge is used to choose a set of possible context questions that might be used to divide the data. This question set will usually include tests for each specific phone, tests for phonetic classes (e.g. stop, vowel), tests for more restricted classes (e.g. voiced stop, front vowel) and tests for more general classes (e.g. voiced consonant, continuant). Typically, there will be over 100 questions for the left context and a similar number for the right context. For each state of each phone, the aim is to build a tree where the question asked at each node is chosen to maximize the likelihood of the training data given the final set of tied states. A condition is imposed to ensure that there are sufficient data associated with each tied state (i.e. that the total occupancy of the tied state exceeds some threshold).

It would in principle be possible to build all possible tree architectures (for all states), train a set of models for each architecture, and choose the set for which the likelihood of the training data is the highest while satisfying the occupancy

condition for each of the final tied states. This strategy would, however, be computationally intractable. Fortunately, by making the assumption that the assignment of acoustic observations to states is unchanged from the assignment for the original triphone models, it is possible to build a tree in a computationally efficient manner using just the state occupancies and the triphone model parameters. When using single-Gaussian distributions, this information is sufficient to calculate new model parameters for any putative combination of the individual triphone states.

The tree-building process starts by placing all the states to be clustered together at the root node of the tree. The mean and variance vectors are calculated assuming that all the states $\boldsymbol{S}$ are tied. Using these values of the model parameters it is then possible to estimate $L(\boldsymbol{S})$, the likelihood of the data associated with the pool of states $\boldsymbol{S}$. The next step is to find the best question for splitting $\boldsymbol{S}$ into two groups. For each question q, the states are divided up according to the answer to the question and new model parameters are computed. The likelihoods $L(\boldsymbol{S}_y(q))$ and $L(\boldsymbol{S}_n(q))$ can then be calculated for the sets of data corresponding to the answers "yes" and "no" respectively. For question q the total likelihood of the data associated with the pool of states will increase by:

$$\Delta L_q = L(\boldsymbol{S}_y(q)) + L(\boldsymbol{S}_n(q)) - L(\boldsymbol{S}) . \tag{12.2}$$

Thus by computing ΔL_q for all possible questions, the question for which this quantity is the maximum can be selected. Two new nodes are created and the splitting process is then repeated for each of the new nodes, and so on. The splitting procedure is terminated when, for all of the current leaf nodes, the total occupancy of the new tied state which could be created at that node falls below the designated occupancy threshold. An additional termination condition is also used, whereby the splitting is halted when the increase in likelihood which would result from a split falls below a specified likelihood threshold. This second termination condition avoids the unnecessary use of different models for contexts which are acoustically similar (even if sufficient data are available for separate models to be used).

Once the tree has been constructed, this tree can be used to accomplish the state tying required for step 3 of the training process described in Section 12.6.4.

12.6.7 Extensions beyond triphone modelling

A useful feature of the phonetic decision tree approach is that it can be extended beyond simple triphone contexts. For example, decision trees can be built using questions relating to the next-but-one left and right contexts as well as the immediately adjacent contexts. The resulting models are often referred to as **quinphones**, as they can incorporate information over a sequence involving up to five phones. Questions relating to the presence of word boundaries can also be included. When building these complex decision trees with such a large number of possible contexts, it is not usually practical to start by training a fully context-dependent system because such a system would typically require a vast number of models and state occupancies to be stored. It is preferable to begin with some more manageable model set and use Viterbi alignment to provide a state-level segmentation of the data. If a tied-state triphone system has been built first, this