Spoken Letter Recognition with Neural Networks

by

James Henry Reynolds

Department of Engineering Science
University of Oxford

Michelmas Term, 1991

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfillment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

J. H. Reynolds, Wadham College

James Henry Reynolds
Wadham College

Doctor of Philosophy
Michelmas Term, 1991

Spoken Letter Recognition with Neural Networks

Abstract

Neural networks have recently been applied to real-world speech recognition problems with a great deal of success. This thesis developes a strategy for optimising a neural network known as the Radial Basis Function classifier (RBF), on a large spoken letter recognition problem designed by British Telecom Research Laboratories. The strategy developed can be viewed as a compromise between a fully adaptive approach involving prohibitively large amounts of computation, and a heuristic approach resulting in poor generalisation. A value for the optimal number of kernel functions is suggested, and methods for determining the positions of the centres and the values of the width parameters are provided. During the evolution of the optimisation strategy it was demonstrated that spatial organisation of the centres does not adversely affect the ability of the classifier to generalise.

An RBF employing the optimisation strategy achieved a lower error rate than a multilayer perceptron and two traditional static pattern classifiers on the same problem. The error rate of the RBF was very close to the theoretical minimum error rate obtainable with an optimal Bayes classifier. In addition to error rate, the performance of the classifiers was assessed in terms of the computational requirements of training and classification, illustrating the significant trade-off between computational investment in training and level of generalisation achieved.

The error rate of the RBF was compared with that of a well established method of dynamic classification to examine whether non-linear time normalisation of word patterns was advantageous to generalisation. It was demonstrated that the dynamic classifier was better suited to small-scale speech recognition problems, and the RBF to speaker-independent speech recognition problems. The dynamic classifier was then combined with a neural network algorithm, greatly reducing its computational requirement without significantly increasing its error rate. This system was then extended into a novel system for visual feedback therapy in which speech is visualised as a moving trajectory on a computer screen.

Acknowledgements

I would like to thank my supervisor, Dr. Lionel Tarassenko, for illuminating discussions, support, and his thoroughness in reading this thesis. I would also like to thank him for the international conferences I have been able to attend; Denver and Dublin spring to mind. I am also indebted to the members of our neural network group who have given me ideas, knowledge, and occasional hysterics. Of these my special thanks to Jon Tombs. The following people did their best to hinder my writing this thesis, for which I shall be forever grateful: Emma Williams, Roberto Cipolla, Bobby Rao, Anna Piussi. I wish to thank Saul Frampton for ridding this thesis of the word 'conciousness', and Dan Kucerov for his special contribution to the Visual Ear. Rue, my cat, was always near to me, especially around meal times for some reason. The bees never let my adrenalin level subside whilst working in the summer-house at Plum Tree Cottage. I am very grateful to my family for brightening up the darker moments of this thesis.

I wish to thank the Science and Engineering Research Council (SERC) for supporting this work, and am grateful to British Telecom Research Laboratories (BTRL) for providing me with the CONNEX alphabet database.

Contents

List of Figures

Chapter 1

Pattern Recognition

1.1 Introduction

There has recently been a great deal of optimism in the field of pattern recognition and machine perception. This is partly due to the relentless increase in computing power and availability which now allows significant real-world problems to be tackled, and partly due to the emergence of a new breed of algorithms known collectively as neural networks. Throughout their history (which can be traced back to McCulloch and Pitts in 1943 [37]), neural networks have caused controversy because of inflated claims as to their power, and biological plausibility. This has had the effect of marginalising the subject to the extent that many of the so-called 'latest' methods now employed by neural network researchers turn out not to be original at all. A detailed inspection of Duda and Hart's famous book, *Pattern Classification and Scene Analysis* [15], reveals how sophisticated pattern recognition techniques had become by 1973. It seems prudent to ask what exactly neural networks have contributed to the field before judging whether the renewed optimism is well founded.

Before proceeding with the analysis there is one important claim of neural networks worthy of correction. That is the claim that they are similar to cognitive mechanisms in the nervous systems of animals, including humans. There seems to be no evidence that either the training techniques, or the operation of the algorithms, has any more biological

plausibility than other abstract mathematical techniques including logical programming[1]. The similarities which do exist are certainly too general to make useful comparisons, and are in no sense the exclusive domain of neural networks. These are 'learning from experience', 'distributed representation', 'massive interconnection', 'parallelism', and 'fault tolerance'. One of the reasons why such claims have been made of neural networks is the *hope* that the present difficulties in machine perception[2] will be overcome. Optimism has waxed and waned regularly over the last thirty years of research in this area as supposedly radical ideas have been shown to suffer from the very same problems that they were initially intended to solve.

A pattern recogniser comprises three separate parts: A transducer, a pre-processor, and a classification algorithm (see Figure 1.1). Its operation can be interpreted as a series of

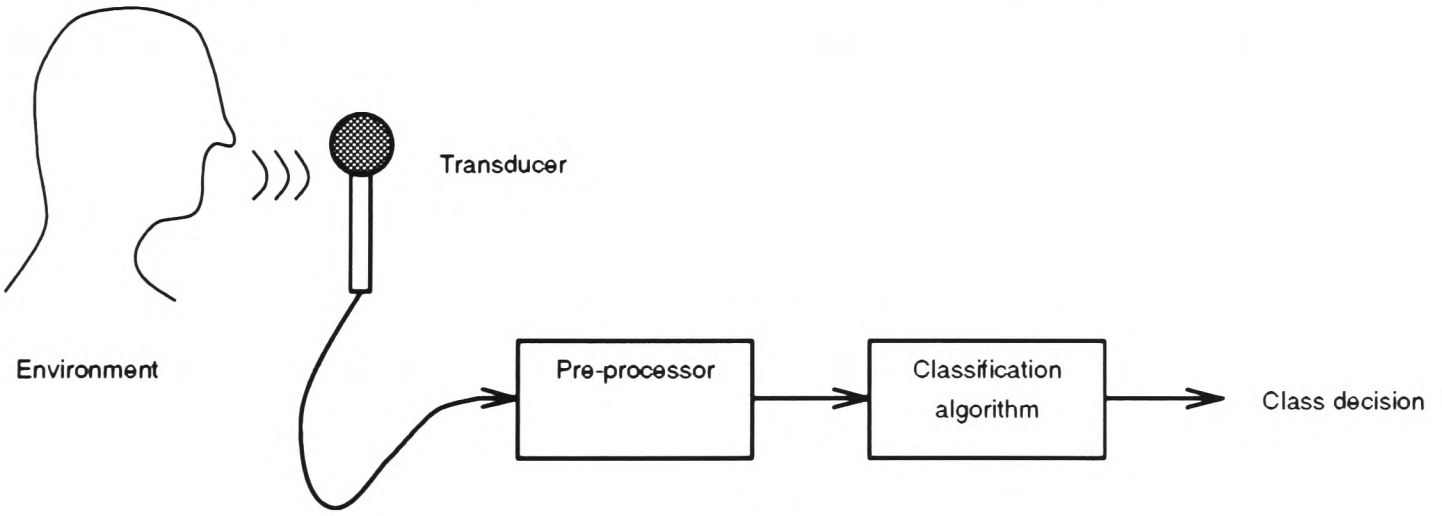


Figure 1.1: *Generalised pattern recogniser. In this case the transducer is a microphone converting human speech into an electrical signal.*

transformations into increasingly abstract levels of representation. The transducer converts

[1]An exception must be made for researchers who deliberately seek to model some aspect of animal behaviour (see for example [3, 25, 52]).

[2]It is intriguing to note that a modern digital computer capable of sustaining 25 million floating point operations per second (25 MFLOPS) cannot be programmed reliably to identify a working vocabulary of words from a human operator. Nor can it be made to name, or handle objects in their natural settings, navigate in realistic unknown environments, or associate environments with events – all of which a human is capable of within a short period after birth.

environmental input stimuli into a computer processable form, allowing the pre-processor to measure their constituent features[3]. The resulting sets of measurements are called *feature vectors*, or simply *input patterns*. The task of the classification algorithm, or *classifier*, is to associate input patterns with one of a number of selected sub-categories, or *classes*:

$$\mathcal{R}^{N_I}_{patterns} \rightarrow \mathcal{R}^{N_C}_{classes} \tag{1.1}$$

where N_I is the dimensionality of the input patterns (number of features), and N_C is the number of classes. Features that vary widely between classes, but little within classes, should be selected to aid classification. They should be stable with time and not be dependent on the method by which they are extracted. For computational efficiency the number of features should be kept to a minimum. Features known to be of little value in discriminating between classes should therefore be avoided. Equation 1.1 requires that the space spanned by the input patterns be partitioned into different regions according to class. In general, the quality of the partition depends on the linear separability of the classes, and the sophistication of the classifier. All classifiers, whether traditional or neural, require empirical knowledge of the input patterns to form the partition. The incorporation of empirical knowledge into the classifier is termed *training*, and is accomplished using a set of labelled training patterns known as the *training set*. The object of training is to build a model of the data source so that unseen patterns, or *test patterns*, will be classified correctly. This ability is known as *generalisation*, and is usually measured in terms of *error rate*, the percentage of the test patterns incorrectly classified.

Much of the present theory of pattern recognition was developed in the 1960's and

[3]If the classifier is required to solve a logical problem (such as the *XOR*, or *parity* problems), then the transducer and feature extraction stages are unnecessary. In general however, the type of classifier discussed in this thesis is better suited to piecewise continuous data sources such as those occurring naturally in the environment.

70's. The absence of large-scale computing resources prevented the systematic assessment of classifiers on real world problems [34]. This is no longer true, and many of the techniques developed during this period have received renewed attention recently. Some examples of these 'traditional' classifiers are listed below.

1. The nearest neighbour and K-nearest neighbour classifiers.
2. The optimal linear transformation.
3. The perceptron.
4. The method of potential functions.

The most significant development of recent research in pattern recognition has been the incorporation of such methods into multilayer architectures, containing one or more layers of non-linear *processing units* (see Figure 1.2). Modifiable connections, or *weights*, connect the

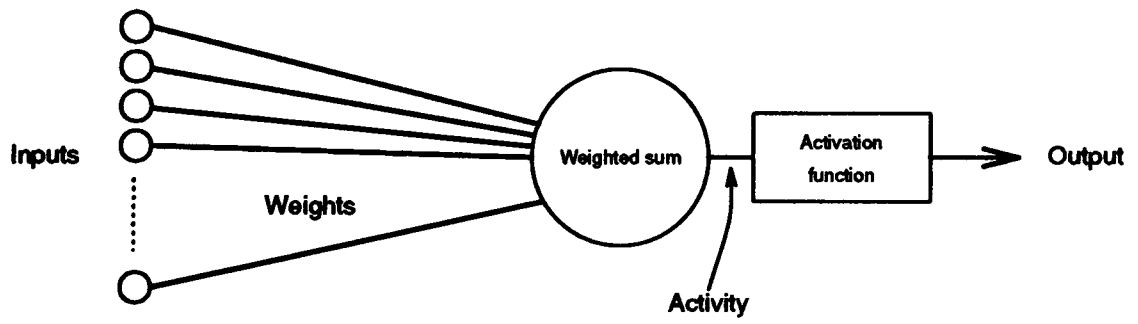


Figure 1.2: *Generalised processing unit. The activity of the processing unit is generated by the weighted summation of its inputs. The activity is passed through the activation function to obtain the processing unit output.*

classifier inputs (or the outputs of processing units in previous layers) to the processing units. The 'activity' of a processing unit is generated by the weighted summation of its inputs. The activity is passed through a non-linearity, known as the 'activation function', in order to obtain the output of the processing unit. Most architectures are feed-forward, meaning that information travels through layers in a forward direction. Architectures containing

feed-back and lateral connections have also been proposed but these are not considered in this thesis. The resulting systems are described as neural networks partly because they are well suited to parallel implementation (the fact that traditional 'non-neural' classifiers are also suited to parallel implementation is often forgotten), and partly because the processing units bear some resemblance to biological neurons. However, much of the complexity of real nervous systems is lost in this analogy. In particular, there is no provision for the temporal processing capabilities or wide variation in neuron functionality that is known to exist in real nervous systems.

1.2 Traditional Classifiers

A proper evaluation of neural network classifiers requires that their performance be compared with equivalent non-neural, or traditional, classifiers. In preparation for such an evaluation two traditional classifiers are first reviewed.

1.2.1 Optimum Linear Transformation (OLT)

The most direct method of pattern classification is the optimal linear transformation, which may be implemented in the architecture of Figure 1.3. There is one processing unit, or *discriminant function*, for each class of pattern. As each employs a *linear* activation function, the output of the ith discriminant function is simply the weighted summation of its inputs:

$$o_i^m = \sum_{j=1}^{N_I} x_{ij} a_j^m + x_{i0} \tag{1.2}$$

where x_{ij} is the jth weight connected to it, a_j^m is the jth component of the mth training pattern, and x_{i0} is a fixed bias. The task of a discriminant function is to separate one class of training patterns from all the others – a process known as *dichotomisation.* Test patterns are therefore classified according to which discriminant function yields the maximum output.

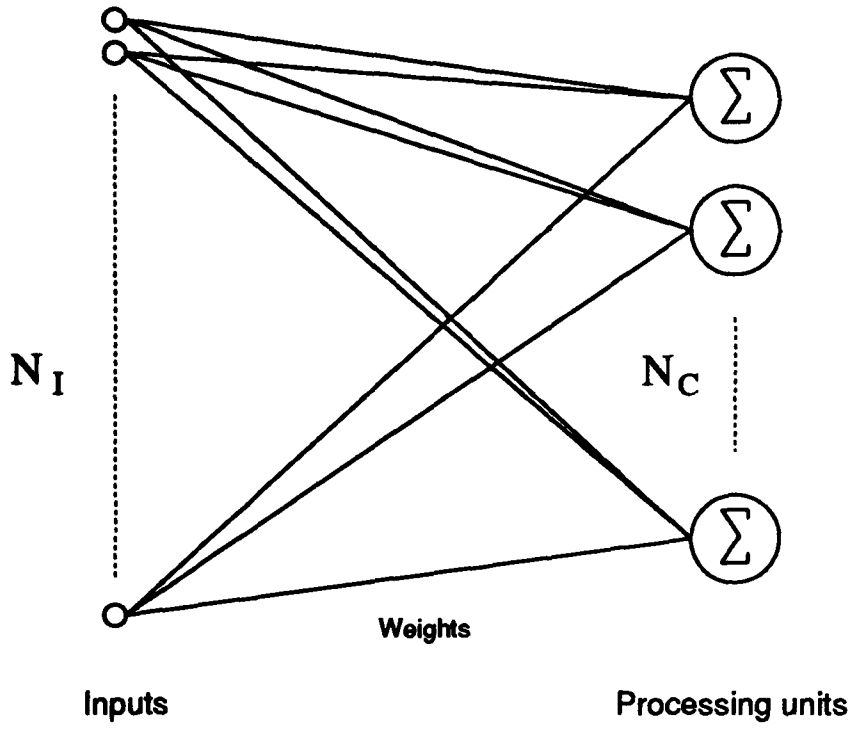


Figure 1.3: *Architecture of the optimal linear transformation. The processing units perform a weighted summation of the inputs, and the activation function is linear.*

A test pattern is assigned to class i^* if:

$$o_{i^*}^{test} > o_i^{test} \qquad \forall i \neq i^* \tag{1.3}$$

Training is accomplished using a *one-out-of-N_C* output coding scheme, such that the target output of the mth training pattern is given by:

$$\begin{aligned} b_i^m &= 1 \qquad \textit{if pattern } m \textit{ belongs to class } i \\ &= 0 \qquad \textit{otherwise} \end{aligned} \tag{1.4}$$

Training can be viewed as the attempted solution of M simultaneous equations, where M is the total number of training patterns in the training set. Again, for the ith discriminant function:

$$\begin{array}{rcl} \sum_{j=1}^{N_I} x_{ij} a_j^1 + x_{i0} &=& b_i^1 \\ \sum_{j=1}^{N_I} x_{ij} a_j^2 + x_{i0} &=& b_i^2 \\ \vdots & & \\ \sum_{j=1}^{N_I} x_{ij} a_j^M + x_{i0} &=& b_i^M \end{array}$$

This problem is more succinctly expressed in matrix form:

$$A\mathbf{x}_i = \mathbf{b}_i \tag{1.5}$$

The purpose of the fixed bias is now made clear: it compensates for the difference in means between the target and actual outputs (the two sides of Equation 1.5) [8]. When $M \gg N_I$, there is no exact solution to Equation 1.5 and an optimisation criterion must be introduced. A *squared error* criterion may be defined as follows:

$$\begin{aligned} e_i &= \sum_{m=1}^{M} (b_i^m - o_i^m)^2 \\ &= ||\mathbf{b}_i - A\mathbf{x}_i||^2 \end{aligned} \tag{1.6}$$

The decision boundary[4] formed by minimising the squared error term of Equation 1.6 approximates the optimal Bayesian decision boundary [58]. Since the complexity of the OLT is fixed (unlike classifiers employing varying numbers of processing units), the *exactness* of the approximation depends only on the number of patterns in the training set and the extent to which they represent the underlying probability distribution of the data source – increased numbers of training patterns leading to an improved approximation. The outputs of the discriminant functions are likewise an approximation to the Bayes *a posteriori* probabilities of the classes [15].

The OLT will be able to discriminate the classes only if a hyperplane exists in input space that will separate each class of pattern from the others. Pattern classes for which this is true are referred to as *linearly separable*. However, the pattern classes of most problems will not be linearly separable, as depicted in Figure 1.4b. In such cases generalisation will only be effective if there are sufficient training patterns to form a good approximation to the

[4] In general, the decision boundary formed by a linear discriminant function is a hyperplane in $N_I - 1$ dimensions, where N_I is the dimension of the input patterns.

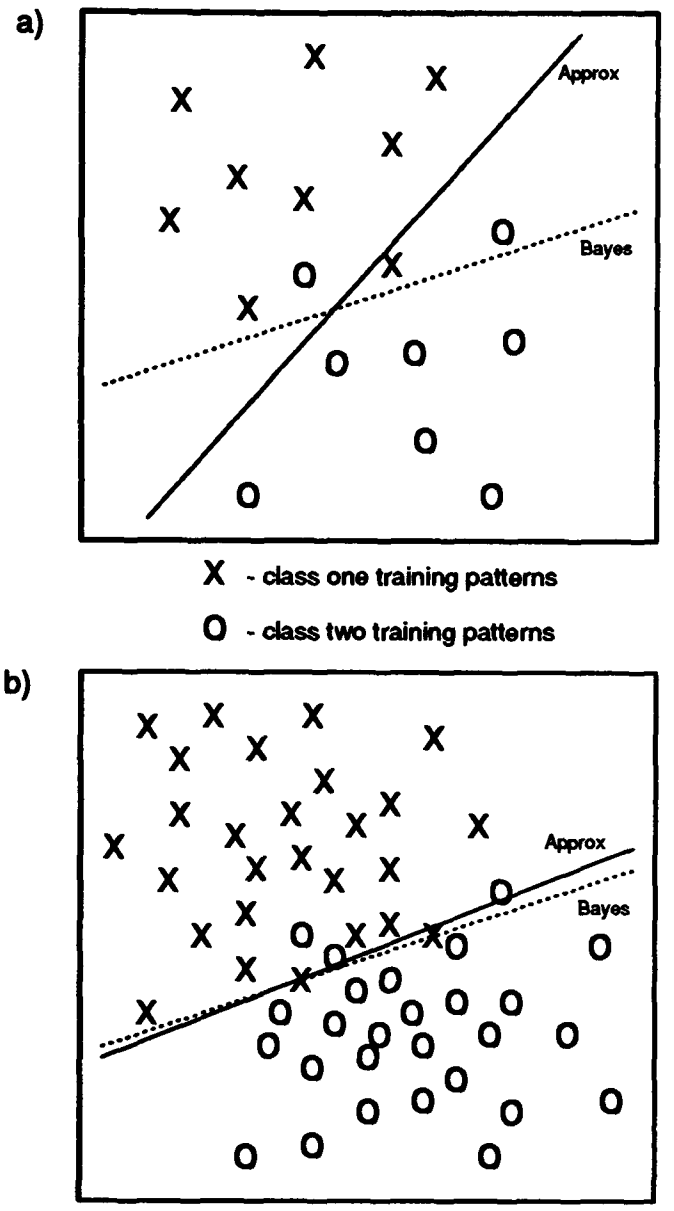


Figure 1.4: *Approximation to the Bayes decision boundary with different numbers of training patterns. a) Coarse approximation with few training patterns. b) Fine approximation with many training patterns.*

Bayes discriminant function, and will therefore improve with greater numbers of training patterns. The following two algorithms are commonly used to perform the OLT.

Pseudoinverse

Equation 1.5 can be written as:

$$\mathbf{x}_i = A^+ \mathbf{b}_i \tag{1.7}$$

where A^+ is the *pseudoinverse*[5] of the rectangular matrix A, such that $A^+A = I$. Equation 1.7 will always yield the minimum squared error solution [15], although it will not be unique if A is rank-deficient (in this case the solution of minimum norm ($||x||^2$) is usually favoured). If A is known to be of full rank, then the simplest technique for computing a solution is *LU decomposition.* However, it is safest to use *singular value decomposition (SVD)*, since this method makes allowance for rank-deficiency automatically before inversion. The $M \times N_I$ matrix A is factorised as follows:

$$A = QDP^T \tag{1.8}$$

where Q is an $M \times N_I$ column-orthonormal matrix, and P is an $N_I \times N_I$ orthogonal matrix:

$$Q^TQ = I \tag{1.9}$$

$$P^TP = PP^T = I \tag{1.10}$$

The $N_I \times N_I$ matrix D is a diagonal matrix, with diagonal elements, $d_1, d_2 \ldots, d_{N_I}$, arranged in decreasing order of magnitude. These are the singular values of A. The inverse of D is also a diagonal matrix with diagonal elements $d_1^{-1}, d_2^{-1} \ldots, d_{N_I}^{-1}$. However, some singular values may be zero or negligible in comparison with the others, indicating that A is rank-deficient. Instead of inverting these singular values, they are zeroed, which has the effect of discarding unnecessary rows in A. If the rank of A is ρ, then the diagonal values of D^{-1} become $d_1^{-1}, d_2^{-1} \ldots, d_\rho^{-1}, 0, \ldots, 0$. Equation 1.7 may therefore be reformed as:

$$\mathbf{x}_i = PD^{-1}Q^T\ \mathbf{b}_i \tag{1.11}$$

Because it is difficult to determine what constitutes a 'negligible' singular value, a constant known as the *tolerance* is supplied to the algorithm by the user. Singular values that

[5]The pseudoinverse of a matrix A can be written as $A^+ = (A^TA)^{-1}A^T$. Thus $A^+A = I$. Pseudoinverses may be generated for any rectangular matrix, and coincide with the true inverse when A is square.

are smaller than the tolerance are zeroed. If, for example, A was populated with values generated from an 8-bit A–D converter, a suitable tolerance would be $\frac{1}{2^8}$, since this is the smallest meaningful value other than zero.

Widrow-Hoff Rule

The Widrow-Hoff rule (or LMS rule) is a first-order *gradient descent* technique, offering an iterative solution to the OLT. Gradient descent requires that the amount by which the weights are adapted upon each iteration is proportional to the gradient (derivative) of the output error with respect to the weights – with negative constant of proportionality [49]. Thus upon presentation of the mth training pattern:

$$\Delta \mathbf{x}_i \propto -\frac{\partial e_i^m}{\partial \mathbf{x}_i} \tag{1.12}$$

This derivative can be split into two separate derivatives using the chain rule:

$$\frac{\partial e_i^m}{\partial \mathbf{x}_i} = \frac{\partial e_i^m}{\partial o_i^m} \frac{\partial o_i^m}{\partial \mathbf{x}_i} \tag{1.13}$$

The derivatives may be obtained by differentiating Equations 1.6 and 1.2 respectively:

$$\frac{\partial e_i^m}{\partial o_i^m} = -2(b_i^m - o_i^m) \tag{1.14}$$

$$\frac{\partial o_i^m}{\partial \mathbf{x}_i} = \mathbf{a}^m \tag{1.15}$$

Substituting Equations 1.14 and 1.15 into Equation 1.13, we obtain the Widrow-Hoff rule for weight adaptation:

$$\Delta \mathbf{x}_i = \eta \, (b_i^m - o_i^m) \, \mathbf{a}^m \tag{1.16}$$

where η is a parameter known as the *learning rate*, and each training pattern, $\mathbf{a}^m$, is selected randomly from the training set. The weights of each discriminant function are initially

defined as small random numbers. Many iterations of Equation 1.16 are then required before the output error converges to a local minimum value. It is impossible to forecast how many adaptations will be required for convergence since the magnitude of the learning rate, the values of the initial weights, and most importantly the complexity of the problem, all affect its speed. Optimisation is therefore required to obtain the most efficient conditions for convergence. The Widrow-Hoff rule can be applied independently of the rank of A, but the extra complexity associated with rank-deficient problems means that longer adaptation periods will be required for convergence.

Pseudoinverse techniques are definitely the most elegant method of performing the OLT, since they provide an *analytical* solution which is not subject to the correct optimisation of learning rate and adaptation time associated with an iterative solution. However, the high memory requirement of pseudoinverse techniques restricts their application to relatively large computing systems (see Chapter 5).

1.2.2 K-Nearest Neighbour Classifier (KNN)

Although the KNN is one of the simplest and most effective classifiers yet developed, its usefulness has been limited until recently by the massive amount of computation required during the recognition, or classification phase. A distance *metric* is first selected based upon the particular characteristics of the data source. For example, a Euclidean metric might be suitable for piecewise continuous data sources occurring naturally in the environment. The distance between the test pattern and the mth training pattern would then be given by:

$$d_m = ||\mathbf{a}^m - \mathbf{a}^{test}|| \qquad 1 \leq m \leq M \tag{1.17}$$

where $||\ \cdot\ ||$ represents the Euclidean norm. The distances between the test pattern and each of the training patterns are computed, and the training patterns are arranged in order

of increasing distance from the test pattern. All but the first K training patterns are then discarded. A histogram of the classes to which those K patterns belong is constructed, and the test pattern is assigned to the most prevalent class. Should there be a tie of classes, one class is selected using a heuristic method – the simplest of which is a random choice from the frontrunners. Generalisation is found to depend strongly on the value of the parameter K. Its optimal value may be discovered with the following procedure:

1. Set $K = 1$ (the nearest neighbour classifier).

2. Record the error rate on test data.

3. Increment K and repeat 2 until the error rate increases or remains constant.

The method of classification employed by the KNN is qualitatively different to that of the OLT. In the latter, hyperplane decision boundaries segment the input space into decision regions according to class. In the KNN, more complex decision regions can be constructed, enclosed by boundaries which are equidistant between training patterns belonging to different classes (see Figure 1.5). The different methods of classification can be described as being based on between-class *discrimination* in the OLT, and within-class *distances* in the KNN. Generalisation in the KNN can be interpreted as *interpolation* between training patterns in input space. It therefore improves with increasing amounts of training data, tending to the performance of the optimal Bayes classifier (which has the lowest error rate of *any* classifier) [15]. This relationship may be quantified: if γ_K is the error rate of the KNN, the error rate of the optimal Bayes classifier, γ_B, can be predicted to lie in the range is $\frac{\gamma_K}{2} \leq \gamma_B \leq \gamma_K$ [24].

Unfortunately, the time taken to classify test patterns increases linearly with the number of training patterns, forcing a trade-off between error rate and classification time. A

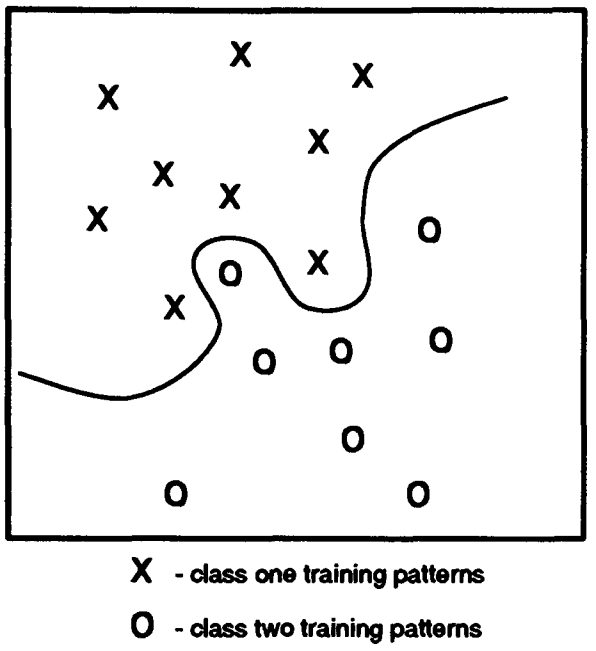


Figure 1.5: *Complex decision boundary formed by the KNN. The decision boundary is equidistant between training patterns of different classes.*

technique which reduces classification time maintains a running list of the closest K training patterns during classification. If, during its calculation, the distance to a training pattern exceeds the Kth nearest distance, then the calculation moves on immediately to the next training pattern. Exactly the same performance is obtained using this technique, but a large amount of computation is eliminated. An interesting modification to the KNN chooses the K nearest training patterns in the same fashion, but deduces the best matching class using linear interpolation rather than by construction of a histogram [60]. The resulting classifier, known as HERBIE[6], has yet to be thoroughly compared with existing classifiers.

1.3 Neural Network Classifiers

Many real world classification problems contain classes of patterns which are not linearly separable, and therefore cannot be separated by hyperplane decision boundaries through input space. More complex decision boundaries are provided by the two neural network classifiers reviewed below.

[6]HEuRistic BInary Engine.

1.3.1 Multilayer Perceptron (MLP)

During the 1950's Rosenblatt developed a discriminant function known as a *perceptron* [47], which was implemented in the architecture of Figure 1.2. The only difference between the perceptron and a linear discriminant function of the type used in the OLT, is the replacement of the linear activation function by a threshold function. A suitable threshold function for the output coding scheme of Equation 1.4 would be:

$$\mathcal{T}(o_i) = 1 \qquad o_i \geq 0.5 \tag{1.18}$$

$$= 0 \qquad otherwise \tag{1.19}$$

The perceptron is trained using an iterative adaptation rule similar to the Widrow-Hoff rule:

$$\Delta \mathbf{x}_i = \eta \, (b_i^m - \mathcal{T}(o_i^m)) \, \mathbf{a}^m \tag{1.20}$$

In his book, *Principles of Neurodynamics* [48], Rosenblatt proposed the 'perceptron learning theorem' which asserted that any problem for which a solution exists would be solvable in a finite number of iterations of the adaptation rule given above. During the 1960's a rigorous analysis of the perceptron was conducted by Minsky and Papert, leading to the book, *Perceptrons* [38]. They demonstrated that problems 'for which a solution exists' meant problems containing linearly separable classes of patterns, and that such problems were also solvable by a linear discriminant function[7]. They also noted that some problems which were not linearly separable (such as the XOR problem) were solvable by a perceptron with an *intermediate unit* with a non-linear activation function. In the absence of a general training

[7]Multi-class problems can be tackled with N_C independent perceptrons, in which case the class to which the input pattern belongs is indicated by the perceptron whose output is 1 (for a one-out-ofN_C coding scheme). The same problem can be tackled by the OLT, in which case the input pattern is assigned to the unit with maximum output. The ability of the OLT to classify input patterns corrupted by noise is superior to that of the perceptron [54].

method for such systems, they concluded that perceptrons were of limited use. Many years later, the development of the *back-propagation* algorithm [49] provided an effective method for training multilayer perceptrons and contributed widely to the recent interest in neural networks.

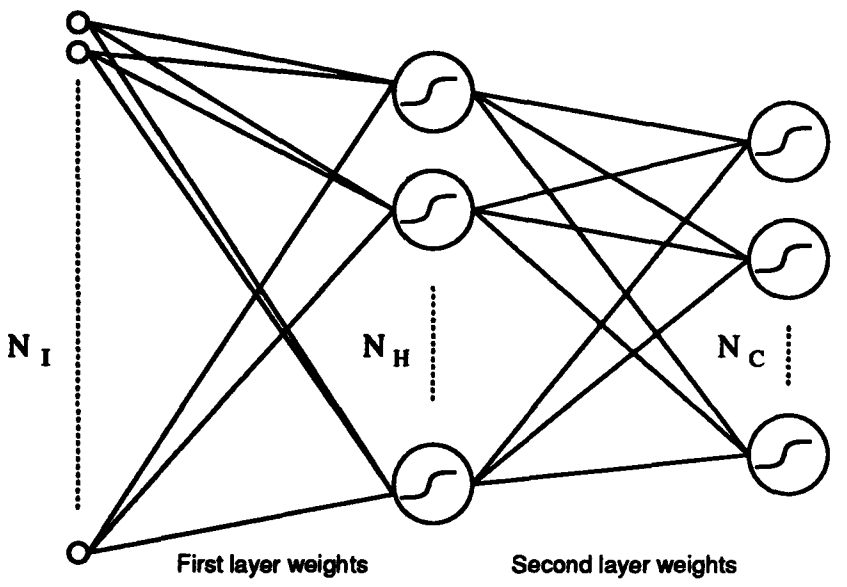


Figure 1.6: *Architecture of the multilayer perceptron.*

The multilayer perceptron is a feedforward neural network containing arbitrary numbers of layers of processing units (see Figure 1.6). All processing units have the same functionality irrespective of their position in the network, although a distinction is made between *output units*, which occur in the final layer of the network, and *hidden units*, which occur between the network inputs and the output units. Each processing unit performs a weighted sum of its inputs, which may be the network inputs or the outputs of processing units in preceding layers depending on the position of the processing unit in the network. The result is passed through a non-linear, differentiable activation function known as a sigmoid function. For instance, upon presentation of the mth training pattern to the network, the output of the

ith hidden layer unit connected to the network inputs is given by:

$$o_i^m = \frac{1}{1 + \exp(-\sum_{j=0}^{N_I} a_j^m x_{ij})} \tag{1.21}$$

where x_{ij} is the weight connecting the jth input to this unit. The task of training is to obtain a set of weights which enable the network to approximate the function specified by the training data. Training is conducted using a first-order gradient descent procedure known as the back-propagation algorithm, in which weights are adapted recursively throughout the network to minimise an error criterion at the output units. Training is complete when the output error has converged to a locally minimum value, but is usually terminated earlier than this, when the output error falls beneath a pre-determined threshold. For instance, a typical termination condition might be:

$$|b_i^m - o_i^m| \leq 0.1 \qquad \forall m \ \forall i$$

where o_i^m is the output of the ith output unit to the mth training pattern, and t_i^m is its target output.

As with the OLT, if the squared error term of Equation 1.6 is employed as the error criterion, the network outputs may be interpreted as estimates of the *a posteriori* probabilities of the classes. However, the additional complexity of the MLP allows a much more exact approximation to the optimal Bayesian classifier than is possible with the OLT. It has been proved that one hidden layer of sigmoidal processing units is sufficient to approximate any desired function [13]. However, there is no analytical method for determining the optimum number of processing units to employ in this hidden layer, despite the existance of upper bounds [2]. The optimum number is therefore determined empirically by evaluating error rate with different numbers of units. A disadvantage of the back-propagation algorithm is that the search for a solution is *unconstrained*; the mappings required by the different layers

of the network *change* during the course of training due to their interdependence. Convergence can therefore be very slow, especially when complex mappings are required. Some researchers have interpreted the operation of the hidden layer units as high-level feature extractors [51, 17], responding to input patterns in such a way as to disambiguate the classes. In general however, their interpretation is ambiguous; the solution was not *designed*, it was *discovered.*

1.3.2 Radial Basis Function Classifier (RBF)

The method of radial basis functions was first proposed by Powell [43] as a generalisation of a family of methods based upon a non-linear discriminant function known as a Φ machine, the most notable of which was the method of potential functions [15]. Broomhead and Lowe [7] later demonstrated that the method could be implemented as a neural network architecture (see Figure 1.7), and hence the method became known as the radial basis function classifier (RBF).

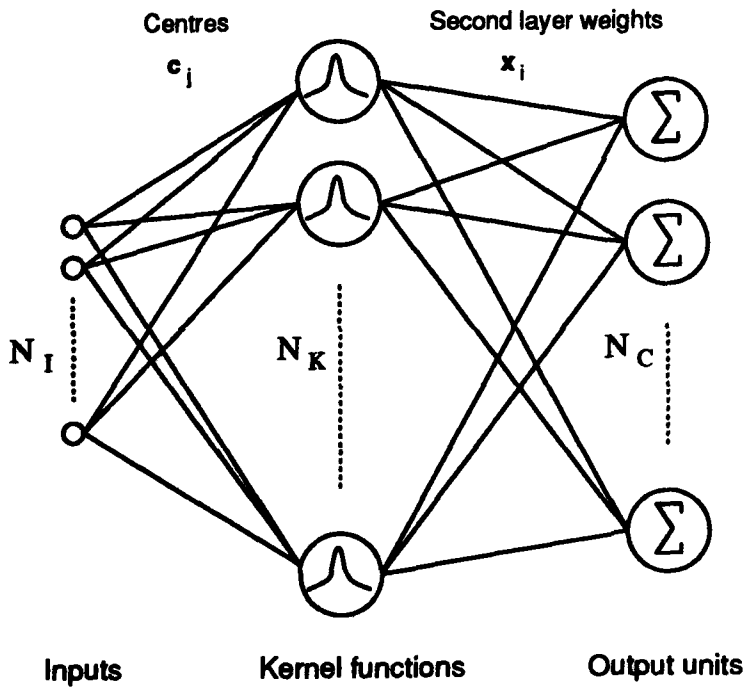


Figure 1.7: *Architecture of the radial basis function classifier.*

In the first layer of the RBF input patterns are projected into a space defined by the

outputs of a set of non-linear *kernel functions*, placed at selected points, or *centres*, in input space. If $\mathbf{c}_j$ is the centre of the jth kernel function, its response to the mth training pattern is given by:

$$\Phi_j^m \equiv \Phi(||\mathbf{a}^m - \mathbf{c}_j||) \tag{1.22}$$

In this way an arbitrarily sized 'basis' of representation of input patterns is constructed, since the number of kernel functions, N_K, is a variable:

$$\mathcal{R}^{N_I} \rightarrow \mathcal{R}^{N_K} \tag{1.23}$$

The second layer of the RBF performs a linear transformation of the projected input patterns. The discriminant function of Equation 1.2 must therefore be adjusted to include the non-linear projection of the input patterns:

$$o_i^m = \sum_{j=1}^{N_K} x_{ij}\Phi_j^m + x_{i0} \tag{1.24}$$

Similarly, the squared error term in Equation 1.6 becomes:

$$e_i = ||\mathbf{b}_i - \mathbf{\Phi}\mathbf{x}_i||^2 \tag{1.25}$$

where $\mathbf{\Phi}$ is the matrix of kernel function outputs to the M patterns in the training set. A pseudoinverse technique is used to compute the second layer weights:

$$\mathbf{x}_i = \mathbf{\Phi}^+\mathbf{b}_i \tag{1.26}$$

Non-linear projection of the input patterns into the space of the kernel function outputs increases the linear separability of the classes, enabling more complex decision boundaries than single hyperplanes to be formed in input space. Further, changes in N_K vary the complexity of the classifier (and therefore the degree of fit to the training patterns), allowing its generalisation performance to be optimised.

Kernel functions with outputs that decrease monotonically to zero as the distance $||\mathbf{a}^m - \mathbf{c}_j||$ tends to infinity, are known as *receptive fields*, since they respond significantly only to input patterns occurring in regions of input space close to their centres. The kernel function employed in the method of potential functions is an example of a receptive field. It models the decay in electrostatic potential with distance from a unit charge:

$$\Phi_j^m = \frac{1}{||\mathbf{a}^m - \mathbf{c}_j||} \tag{1.27}$$

A more flexible alternative is the *radially symmetric*[8] (isotropic) Gaussian function:

$$\Phi_j^m = \exp\left(-\frac{||\mathbf{a}^m - \mathbf{c}_j||^2}{2\sigma_j^2}\right) \tag{1.28}$$

where σ_j is the *width* parameter of the Gaussian function. Kernel functions that do not act as receptive fields have also been studied [6]. An example is the simple linear function:

$$\Phi_j^m = ||\mathbf{a}^m - \mathbf{c}_j|| \tag{1.29}$$

or the thin plate spline function:

$$\Phi_j^m = ||\mathbf{a}^m - \mathbf{c}_j||^2 \log(||\mathbf{a}^m - \mathbf{c}_j||) \tag{1.30}$$

The radially symmetric Gaussian function is used throughout this thesis, as it allows the adjustment of kernel function width (note that width parameters do not occur in the other kernel functions preventing their optimisation in this respect). Centres can be selected randomly from the training set, or they can be derived using a *self-organising* algorithm that maximises their statistical correspondence with the training data. Alternatively, the positions of the centres and any additional kernel function parameters may be derived using a fully adaptive procedure [36].

[8] This simplification is satisfactory so long as there is no *a priori* evidence to suggest that the components of the input patterns should be weighted unequally, as might be the case, for instance, when input patterns are derived from multiple sources.

The operation of the RBF can be split into two clearly defined stages, associated with the two layers: input patterns undergo a projection into a space in which the linear separability of the pattern classes increases, after which a linear transformation is more likely generate the correct classification. It is therefore possible to say *how* classification proceeds in a more explicit manner than in, say, the MLP. The problem of unconstrained gradient descent encountered in the MLP is also avoided, since the output error is minimised over a single independent layer.

1.4 Summary of Thesis by Chapter

The first objective of this thesis is to develop an effective method for optimising the RBF on a difficult speech recognition problem (speaker-independent spoken letter recognition). The background to this problem is provided in Chapter 2 with an examination of human speech perception and the inherent ambiguity of the speech waveform. In Chapter 3 existing methods of optimising the parameters of the RBF are reviewed and significant new improvements proposed. Empirical results are presented in Chapter 4 which demonstrate the effectiveness of the optimisation procedures and provide insight into their operation.

The second objective of this thesis is to compare the performance of the optimised radial basis function classifier with the multilayer perceptron, and the two traditional classifiers examined in this chapter. Performance is assessed not only in terms of the error rates achieved in the speech recognition problem, but also in terms of the computational requirements of both the training and classification phases. The results of the assessment are presented in Chapter 5.

All the classifiers examined up until that point are referred to as 'static', or 'fixed length' classifiers, since they process input patterns of fixed dimension (in this case time-normalised

word templates). The third objective of this thesis is to investigate 'dynamic' classification algorithms capable of modelling temporal variation in speech explicitly. Chapter 6 investigates a classifier which combines a well established dynamic classification algorithm known as dynamic time warping (DTW), with a neural network algorithm in order to reduce computation without a significant increase in error rate. A small modification to the design enables speech sounds to be visualised as moving trajectories on a computer screen during classification. Chapter 7 investigates how the design can be extended to form a unique method of visual feedback from speech, which may find application in speech therapy.

Chapter 2

Speech Recognition

2.1 Introduction

Spoken language enables human beings to communicate effortlessly with each other. Where possible this is enriched through the use of hand gestures and facial expressions, but these are not necessary as demonstrated by the success of the telephone. Humans are not the only animals to use a sophisticated method of inter-communication. It has recently become clear that whales, dolphins and higher primates also possess some linguistic ability, although it would be homocentric to expect this to be manifested in a similar structure to our own languages. But even humans, who have long claimed to be the sole users of language, have so far not been able to construct a machine capable of recognising, let alone *understanding*, the words spoken in the course of a natural human conversation. Such a machine would be applicable to a wide variety of problems. The ultimate piece of office equipment would surely be the translating telephone, a device allowing natural communication between two people speaking in different languages. This would require the simultaneous recognition, translation and synthesis of speech in two languages. Closer to the horizon is the automatic typewriter, capable of committing dictated text into printed copy. An area of perhaps more immediate concern, and one which is becoming a practical proposition, is the construction of communication aids for the disabled. The ability to control domestic appliances by voice would make a big impact on the lives of immobile people. The speech-impaired could

also benefit from such technology if the speech recogniser was trained to respond to their individual voices.

There are many reasons why it has proved harder than expected to construct machines that recognise speech as well as humans. Foremost is the reliance on bottom-up processing of the acoustic waveform as the sole means of recognition. This stems from the absence of a definitive theory of human speech perception, rather than a lack of computing power. We have the computing resources, but cannot fathom how best to use them. A brief review of what we do know about human speech perception is presented below, preparing the way for an examination of the difficulties faced by automatic speech recognisers. The speech recognition problems studied in this work were derived from the CONNEX alphabet database [1]. This database is described in full, including an examination of the pre-processing techniques used on the data, and a summary of previous research making use of the database.

2.2 Speech Perception

Before it is possible to discuss how the human brain perceives speech, we need to study the nature of the speech waveform and the performance of the human hearing system.

2.2.1 Nature of Speech

Speech is an acoustic waveform conveying verbal information between human beings. The bulk of its energy lies in the frequency range 100 Hz – 5 kHz. A speech utterance can be broken down into a set of words, each of which can be reduced into a set of basic sound units known as *phonemes.* Depending on the degree of acoustic variation associated with each phoneme, between 40 and 60 phonemes have been suggested to represent spoken English,

[1] This database was specifically collected by British Telecom Research Laboratories (BTRL) for research into neural network models.

the requirement being that all spoken words (not written) are uniquely defined. Phonemes can be arranged into categories according to their method of production in the vocal tract. A useful model of speech production is the *excitation-modulation* model which views the vocal tract as a filter excited by either a periodic signal for voiced speech, or turbulence (wide-band excitation) for unvoiced speech (see Figure 2.1, adapted from Parsons [42]). The

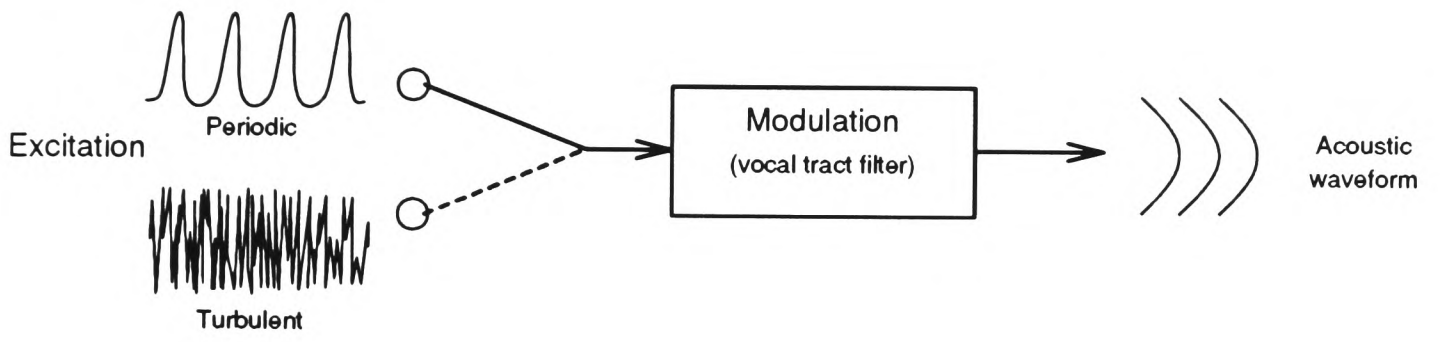


Figure 2.1: *Excitation-modulation model of vocal tract.*

acoustic waveform is produced by the convolution of the excitation source and vocal tract impulse response in the time domain, or the product of the excitation spectrum and vocal tract transfer function (spectrum envelope) in the frequency domain:

$$x(t) = g(t) * h(t) \qquad \textit{time domain}$$

$$X(f) = G(f)\,H(f) \qquad \textit{frequency domain}$$

The implicit assumption of the model is that $h(t)$ is time-invariant, which is not the case in practice due to the constantly changing shape of the vocal tract. However, over short periods of time the vocal tract may be considered stationary justifying the use of the model (see Section 2.4.1). Two basic categories of phonemes can be determined using this model: *vocoids* and *contoids*. Vocoids are voiced sounds, originating at the glottis when the vocal cords are forced to vibrate in the flow of air from the lungs (a phenomenon known as phonation). They are characterised by periodic waveforms, with a fundamental frequency,

or pitch, depending on the degree of tension in the vocal cords (in general, the pitch of male voices lies in the range 80 Hz – 200 Hz, with the pitch of female voices an octave higher). As the waveform travels through the vocal tract to the surrounding air it is modulated by the vocal organs. Tone quality is determined by the position of the tongue and the shape of the opening formed by the lips, and nasality by the position of the velum.

Contoids are produced when air flowing out of the lungs meets a constriction in the vocal tract, leading to turbulent air flow. Different types can be produced by varying the point and degree of the constriction, and by the selective use of phonation (contoids may be voiced or unvoiced). The main subcategories of vocoid and contoid are given in Table 2.1.

Table 2.1: *The main categories of vocoid and contoid.*

Category	Subcategory	Example
Vocoid	Vowels	h<u>a</u>d
	Nasals	<u>m</u>oon
	Liquids/Glides	<u>y</u>ork
Contoid	Fricatives	<u>s</u>now
	Plosives	sto<u>p</u>
	Africates	wa<u>tch</u>

2.2.2 Human Hearing System

The average human hearing system detects sounds in the frequency range 16 Hz – 16 kHz[2], which is in excess of the bandwidth of human speech. However, the perceived frequency of a sound, and therefore the detail with which it is resolved, is a non-linear function of its real frequency. By asking human subjects to estimate the frequency ratio of a series of

[2]The bandwidth of the human hearing system is strongly dependent on age; it may extend beyond 20 kHz in the case of children, and drop as low as 5 kHz in the case of elderly people (particularly males).

test notes relative to a reference note[3], their perceived frequencies can be obtained. These are measured in *mels*, and the graph plotting perceived frequency against real frequency is known as the *mel scale* (see Figure 2.2, adapted from Parsons [42]). It is clear that the lower frequencies are resolved with greater precision than the higher frequencies, indicating that the former carry more information per unit bandwidth. For computational convenience, the mel scale is often approximated as a linear scale up to 1 kHz, and a logarithmic scale thereafter.

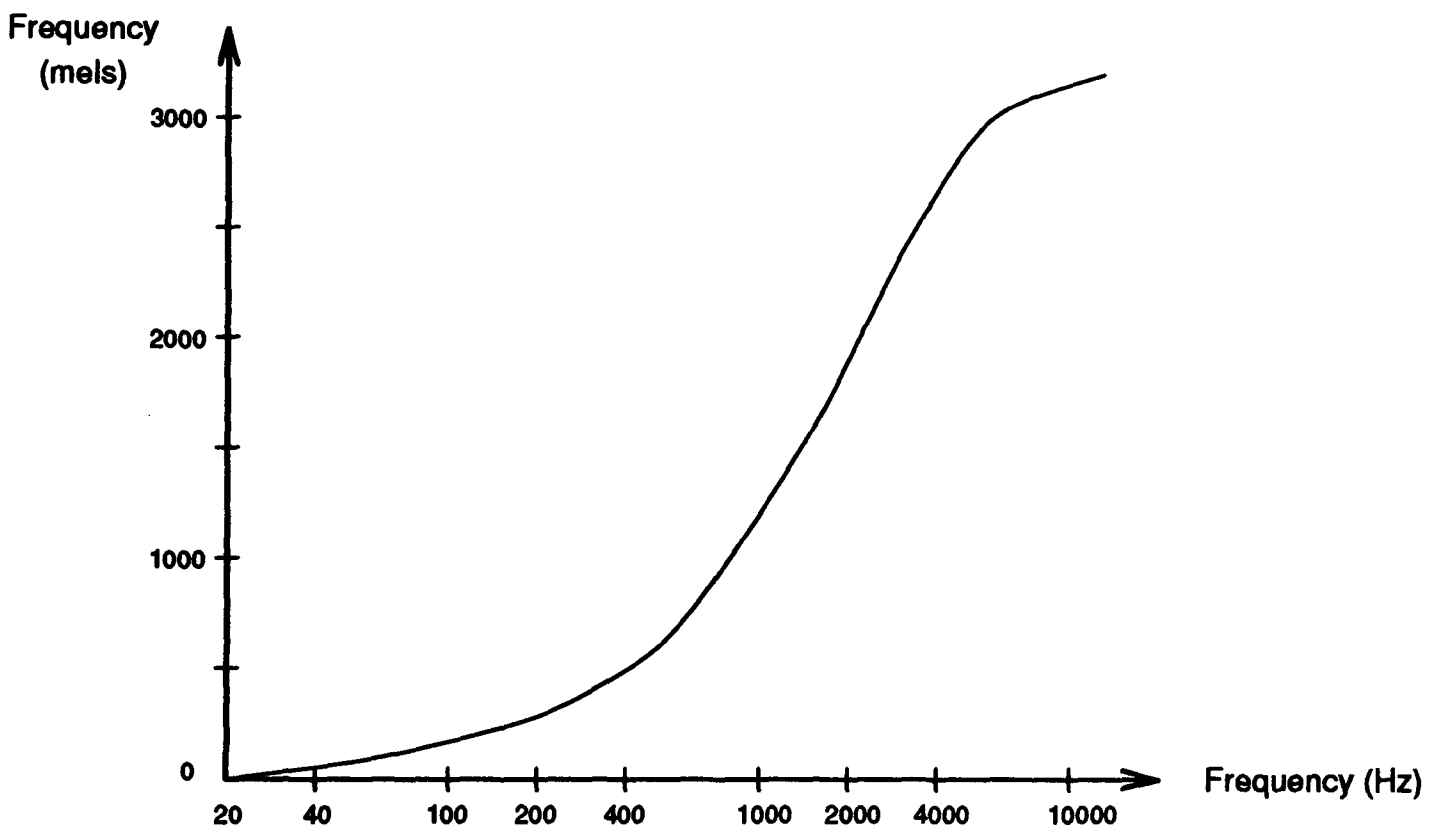


Figure 2.2: *Perceived frequency as a function of real frequency: the mel scale.*

The hearing system processes sounds ranging in intensity from 0 dB – 130 dB, where 0 dB is associated with a sound pressure of 2×10^{-5} N/m^2 (the usual boundary of silence for humans at 1 kHz). As with perceived frequency, the perceived loudness of the sound (mea-

[3]Subjects are provided with a frequency generator driving a loudspeaker, and are asked to set the frequency of the generator to $\frac{1}{2}, 2, \frac{1}{10}, 10$ times the frequency of a reference sound. Many such experiments are conducted over a wide number of individuals to obtain a statistically valid scale.

sured in *phons*) varies non-linearly with frequency (see Figure 2.3, adapted from Holmes [21]). It appears that the hearing system has a very flexible automatic gain control mecha-

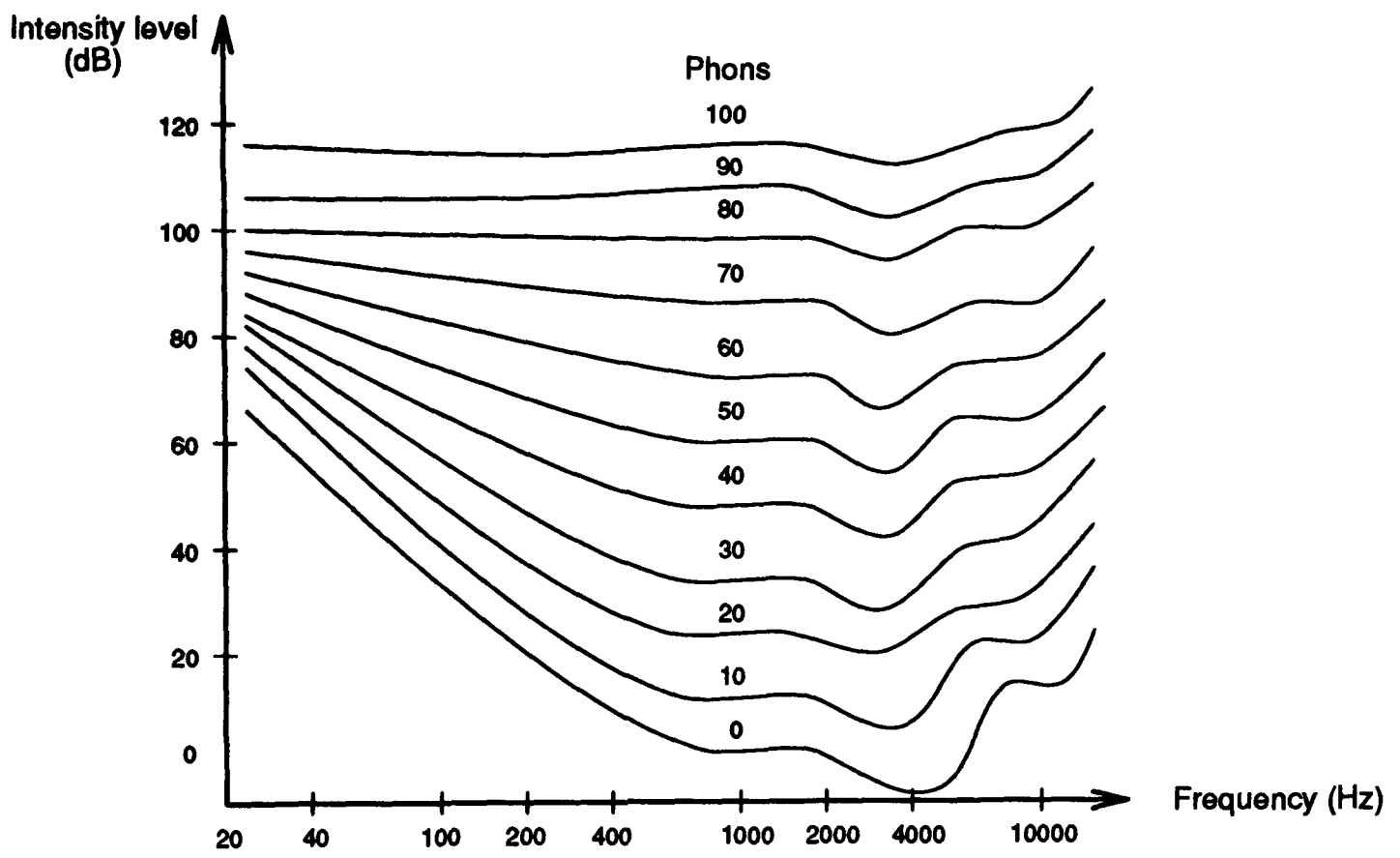


Figure 2.3: *Contours of equal perceived loudness as a function of frequency.*

nism because the average intensity level of speech sounds has little effect on intelligibility, provided it falls within the accepted decibel range. It does convey information regarding the emotional state of the speaker however, and in some languages (such as Russian) plays a semantic role.

2.2.3 Speech Perception

Modern theories of speech perception contain a mixture of ideas that attempt to explain the following psychophysical observations [42]:

- The brain processes speech sounds differently to ordinary sounds, possibly in a separate area. When ordinary sounds are made progressively more like speech sounds there is a sharp transition in the way they are perceived.

- Speech sounds are perceived in categories, not as a continuum. When the acoustic properties of the word 'bad' are slowly altered to produce the words 'dad', and 'gad', sharp transitions are again noted in the way they are perceived.

- Humans cannot ignore speech sounds[4].

- Speech sounds obscured by noise are more readily interpreted if they form part of a coherent context. For instance, sentences of random words are less likely to be recognised than those conveying meaning. Nonsense sounds are even less likely to be recognised.

The theory of Cole and Jakimik [10] (adapted from Parsons [42]) outlined below is one of the better established modern theories:

- Speech cannot be perceived on the basis of the acoustic waveform alone, since this provides insufficient information to disambiguate meaning. For instance, how do humans differentiate between the utterances:

  'recognise speech'

  and:

  'wreck a nice beach'

[4]The fact that the interpretation of speech sounds is automatic does not necessarily prove that it is innate. Many aspects of human behaviour become automatic after they have been learnt, and speech interpretation is probably one of them.

when there is almost no *acoustical* difference between them ? Knowledge of the grammatical constraints of the language, the intentions of the speaker, and the topic under discussion, are obviously important sources of extra information for the listener. Speech perception is therefore driven by the acoustic waveform, but guided by mental processes (the latter are often so powerful that a listener can predict speech before it is uttered, thereby avoiding the necessity to converse).

- Speech is processed word by word in the order in which they arrive. Each word contributes to the overall context of the sentence, reducing the search space for the following words.

- The same process is applied to the phonemes making up an individual word. Each phoneme, once categorised, is used to limit the search space of the subsequent phonemes (the first phoneme certainly carries most information). Recognition is often completed before the word is finished.

Most automatic speech recognisers also use a left-to-right, bottom-up processing strategy. However, human speech recognition is distinguished by its additional reliance on top-down guidance, the significance of which cannot be overstated. Top-down guidance enables humans to understand speech from a huge variety of speakers, such as the crudely synthesised speech of a computer, or the incoherent speech of a heavily accented foreigner. Its predictive ability also allows words to be recognised when corrupted by environmental noise, or by the conversations of other people.

2.3 Automatic Speech Recognition

Matching human performance at speech recognition is a very challenging and as yet unsolved problem. The purpose of this section is to provide a background description of the difficulties faced, and indicate what steps, if any, can be taken to overcome them.

2.3.1 Variation in Speech

Such is the variation encountered in everyday speech that there is no automatic correspondence between acoustic waveforms and phonemes. Quite different waveforms may be intended to represent the same phoneme, or conversely, similar waveforms may be associated with different phonemes. The main problem facing the bottom-up approach to automatic speech recognition is therefore the inherent *variation*, or ambiguity of the acoustic waveform. A speech recogniser can only be expected to perform successfully if it contains an accurate model of this variation. This requires access to a large body of training data providing examples of the following sources of variation.

Inter-speaker Variation

No two voices have equivalent quality or prosody (the collective term for suprasegmental aspects of speech such as intonation, rhythm, and stress). Speaker sex, age, and origin also contribute variation.

Intra-speaker Variation

Over sufficient time the voice of an individual will change (this is particularly true during adolescence). The physical condition of the speaker also has an effect on voice quality, especially if they are weak, suffering from a throat disorder, or in an emotionally disturbed state.

Words are seldom uttered with exactly the same duration. Furthermore their constituent sounds may be uttered at relatively different rates.

Environmental Noise

Other than the speech itself, all sounds heard by the listener can be considered to be noise. The extent to which the speech waveform is disrupted depends entirely upon the environment in which it is heard (e.g. airport runway, empty church).

Speech Interference

A special type of environmental noise occurs when the listener is attending to one speaker in the presence of others. This leads to potentially more ambiguity than ordinary noise sources since speech signals are less easily ignored.

2.3.2 Solution through Simplification

At present, good recognition results can only be obtained by artificially simplifying natural speech recognition problems in the following ways.

Number of Speakers

Problems may be ranked in order of difficulty as single-speaker, multiple-speaker, or speaker-independent. Single-speaker problems are by far the easiest to solve since intra-speaker variation is generally less pronounced than inter-speaker variation. Problems involving speech from a restricted number of speakers (sometimes many) are termed multiple-speaker problems. They generally increase in difficulty with the number of speakers involved. Both the training data and test data making up the problem are normally collected from the entire set of speakers. In speaker-independent problems the training data is collected from

a large number of speakers with the intention of building an extensive model of inter-speaker variation. They are regarded as the hardest problems to solve since the adequacy of the model is assessed with test data collected from a different set of speakers to those used during training.

Size of Vocabulary

The difficulty of a problem will obviously increase with the size of the vocabulary intended to be recognised. Small vocabularies such as the digits, '0–9', or the alphabet, 'a–z', are often used to limit difficulty whilst maintaining practical value. Larger vocabularies can easily be implemented if a grammatical structure is imposed on incoming speech, producing a small but dynamically changing vocabulary.

Type of Speech

The task of speech recognition is made considerably easier if speech is entered in the form of *isolated words*, rather than continuously as normal, since this automatically defines their endpoints. The interruptions imposed between each word also encourage them to be spoken clearly, and reduce the effect of coarticulation[5] across word boundaries. The latter is particularly destructive of short words in continuous speech [42], which are often abbreviated or ignored altogether.

Recording conditions

To avoid contamination by environmental noise, speech can be collected in an anechoic chamber or other quiet environment.

[5]The dependence of articulation on phonemic context.

2.4 The CONNEX Alphabet Database

It is a fact that the classifiers introduced in the previous chapter are more suited to processing isolated words than continuous speech utterances, because the accessibility of the endpoints in the former allow them to be represented as static, or fixed dimension patterns. To avoid introducing non-standard modifications to the classifiers in order to process continuous speech, it was decided to assess their performance on a database of isolated words. All the speech recognition problems tackled in this thesis were constructed out of the CONNEX alphabet database, which consists of three isolated utterances of the letters of the alphabet, 'a–z', by each of 104 speakers taken from all age groups (between 18 and 65), and distributed evenly between the sexes. The version of the CONNEX alphabet database used in this thesis came in a pre-processed form. Each word in the database is represented as a series of feature vectors rather than the original acoustic waveform in digital form, reducing the storage requirement from 600 to 50 Mbytes. The CONNEX alphabet database is an attractive choice for evaluating speech recognisers for the following reasons:

- The database contains approximately 8000 isolated words, which is of sufficient size to allow a good model of speech variation to be constructed from the training data, and for a statistically valid measure of generalisation to be obtained from the test data.

- The database contains speech from 104 speakers which is sufficient to design speaker-independent problems (52 speakers for training, 52 for testing). This increases the practical value of the research.

- Discrimination of the alphabet is known to be a 'hard' problem, since it contains three very confusable sub-vocabularies of letters: the 'e' set ('b, c, d, e, g, p, t, v'), the 'a'

set ('a, j, k'), and the 'mn' pair. The 'e' set is particularly hard to discriminate since the sounds are almost identical apart from their brief initial consonant. Successful discrimination of the alphabet is a good indication that alternative vocabularies of the same size will also be discriminable.

- The database has already been used to assess the performance of a variety of traditional and neural network classifiers, establishing a benchmark for subsequent work (see Section 2.5).

2.4.1 Pre-processing

Speech pre-processing is a very advanced field in its own right. Some of its key features are reviewed below using the methods adopted in the CONNEX alphabet database as an example.

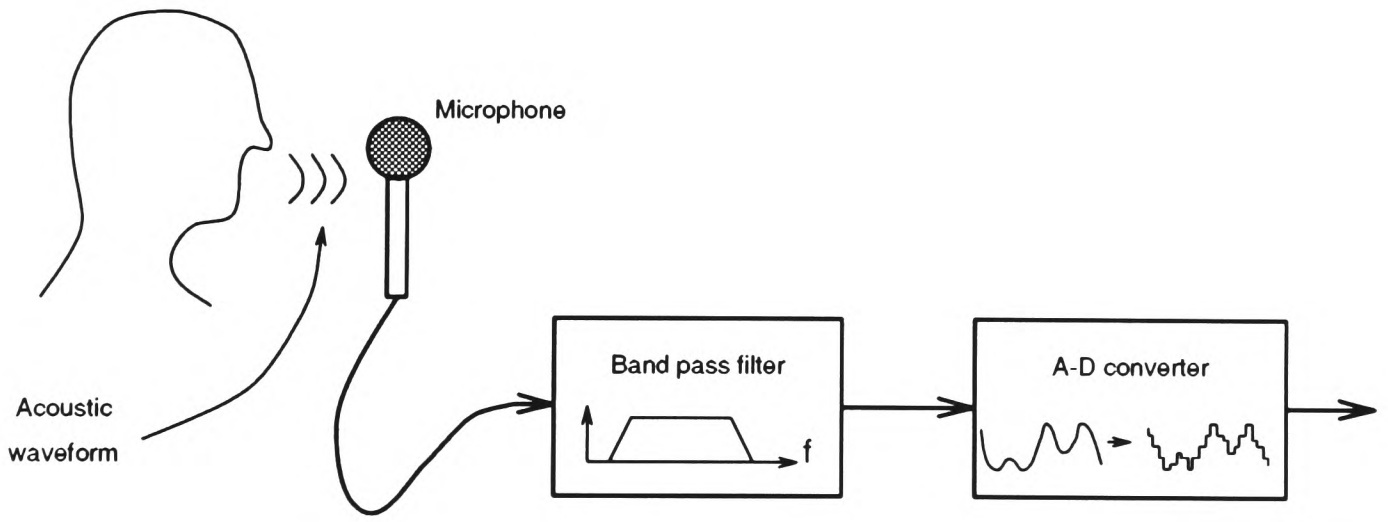


Figure 2.4: *Three stages in digitising the acoustic waveform.*

Digitising the Acoustic Waveform

Since automatic speech recognition is performed by digital computers the acoustic waveform must be converted into the digital domain. This takes place in three stages (see Figure 2.4).

The acoustic waveform is firstly converted into an analogue electronic signal by a microphone (transducer). This signal is then band-limited (see below) and sampled by a 16-bit A–D converter at 20 kHz. Band-limiting consists of the following filters:

1. Low-pass, cut off 8 kHz. This is known as an anti-aliasing filter, since its function is to prevent frequencies in excess of 10 kHz from appearing as lower frequencies in the sampled signal (Nyquist's theorem requires that the sampling rate be at least twice as high as the maximum signal frequency to avoid aliasing).

2. High pass, cut off 170 Hz. This filter removes unwanted low-frequency noise and partially removes the fundamental frequency component of the speech signal[6].

We noted in Section 2.2.1 that the bulk of the energy associated with the speech signal lies below 5 kHz. The 170 Hz – 8 kHz bandwidth made possible by the 20 kHz sampling rate is therefore more than adequate for speech recognition. Since each sample consists of 16 bits, one second of speech requires $20K \times 16 = 320K$ bits for storage. In contrast, civil telephony transmits 8 bit samples at 8 kHz[7], so that one second of speech requires just $8K \times 8 = 64K$ bits for storage. Speech recognition experiments can of course be conducted on speech digitised to the specification of civil telephony, but experimental results are less likely to be prejudiced if higher quality data is used.

Endpoint Detection

During the classification of isolated words it is assumed that the digital signals associated with the words refer solely to their speech content, not to the background noise preceding

[6]The value of the fundamental frequency, or pitch, does not play a significant role in discriminating between phonemes in the English language.

[7]The resulting bandwidth of 300 Hz – 3.4 kHz is transmitted without serious loss of intelligibility.

and following their utterance. This requires detection of the boundaries, or *endpoints* of the words in the incoming digital signal. The simplest method of accomplishing this is based on the analysis of the short-term energy in the signal, given by:

$$\varepsilon = \frac{1}{S} \sum_{i=1}^{S} s_i^2 \tag{2.1}$$

where s_i is the ith sample in the S samples produced by the A–D converter over a short interval of time (normally 5 ms–20 ms). At regular intervals ε is compared with a threshold, θ, placed just above the energy level associated with background noise. If ε exceeds the threshold for a certain number of intervals then it is assumed that the word has begun. Similarly, if the energy falls beneath the threshold for a sufficient number of intervals then the word is assumed to have ended (see Figure 2.5). Unfortunately, there are significant

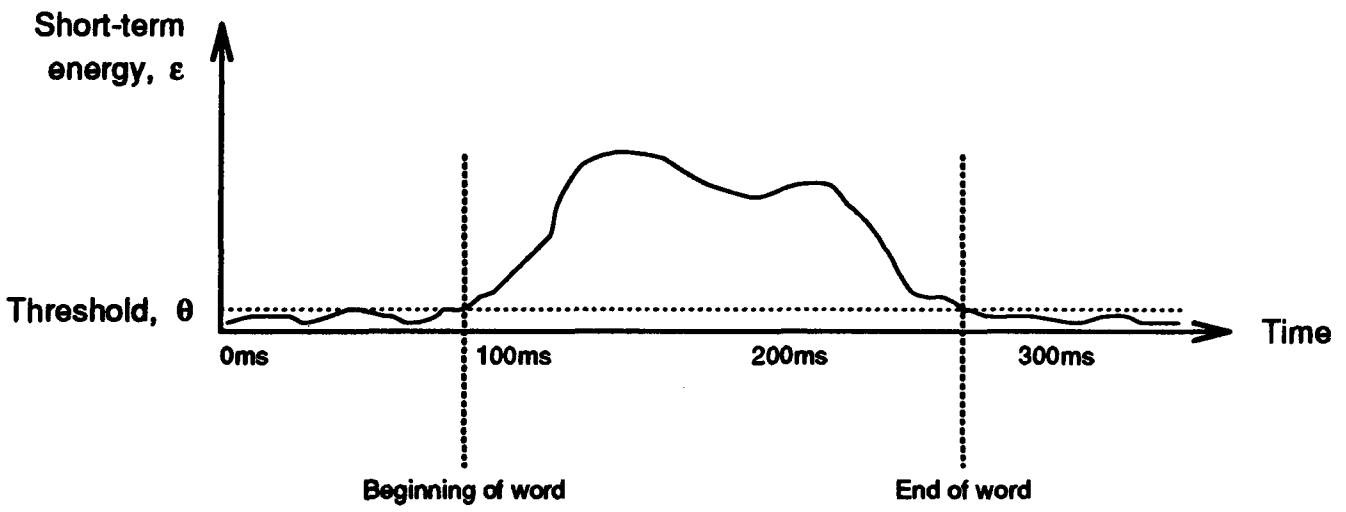


Figure 2.5: *Simple energy-based endpoint detection.*

problems with such a simple algorithm:

- Fricatives are characterised by very low energy, often not appreciably greater than background noise. For this reason a simple energy-based algorithm may not detect their presence. One solution is to include zero-crossing rate as an additional means

of detection, since this parameter is very sensitive to the high frequencies associated with fricative excitation.

- The brief periods of silence in words containing plosives can lead to the assumption that the word has finished. A minimum word length time is often imposed, or if possible, the end of a word should be located by backtracking from the end of the recording period (i.e. from a point at which the word is known to have finished).

- Breath noise and other non-intended verbal artifacts are not likely to be rejected by the simple algorithm.

The endpoint detection algorithm used in this database was developed by Wilpon et al [59] (an explanation is beyond the scope of this thesis). The endpointed words were then played back and their endpoints adjusted manually in cases where the algorithm had failed. At this point 136 words which were either spoken incorrectly or of excessive length were rejected. Thus, out of a total of 8112 words ($104 \times 3 \times 26$), 7976 remained.

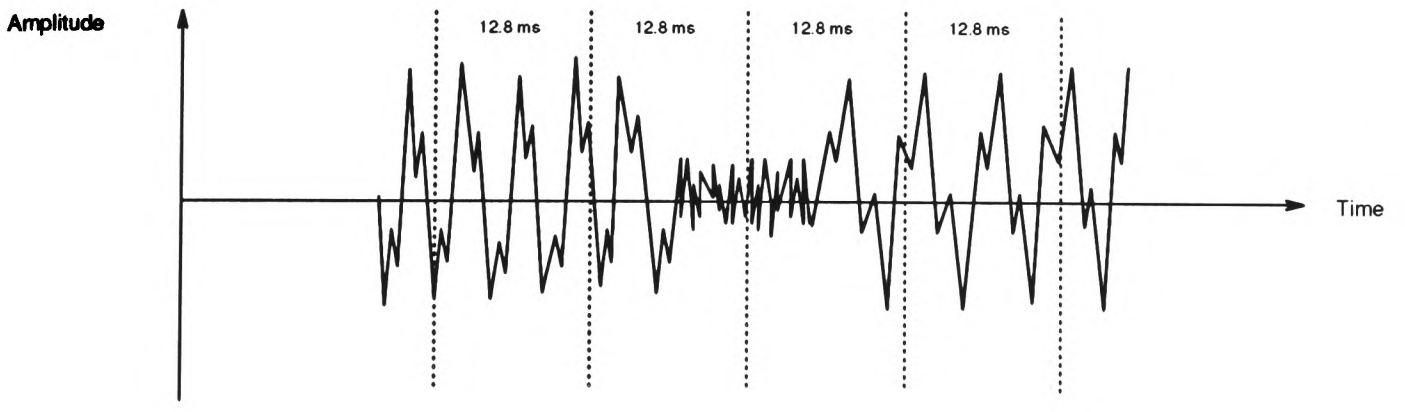


Figure 2.6: *Quasi-static representation of speech signal.*

Feature Extraction

The process of digitisation and endpoint detection results in variable length digital signals corresponding to the acoustic waveforms of the words spoken. Much of the speech signal is redundant due to the relative slowness with which speech can be articulated; speech is a *quasi-static* signal, meaning that it can be considered stationary over short periods of time, usually fixed at some value in the range 5 ms – 30 ms (see Figure 2.6). The short periods of time are known as frames, and the segment of the speech signal associated with each frame can be described by a set of features, known as a feature vector. Depending on its length, each word is represented by a variable number of feature vectors. When these are combined together, the resulting pattern is known as a word *template.* We noted in the previous chapter that well chosen features help discriminate between different classes of patterns. Features such as the short-term energy and zero-crossing rate are referred to as *time-domain features*, since they are obtained directly from the digitised acoustic waveform. Such features provide a coarse basis for comparison of speech sounds. For instance, it is difficult to distinguish between vowel sounds purely on the basis of their short-term energy profile – they are just too similar. Finer grain features of the speech signal can be obtained by transforming the speech signal into the frequency domain where its component frequencies are revealed. All but the most primitive techniques of feature extraction include a form of spectral analysis. The technique employed in this database was *mel frequency cepstral analysis* (cepstral analysis incorporating the mel frequency scale), which has been used extensively since its effectiveness was documented in a highly regarded survey [14].

Mel Frequency Cepstral Analysis

The Discrete Fourier Transform (DFT) of the S sample digital signal, $x_0, x_1, \ldots, x_{S-1}$, associated with each frame of speech, is given by:

$$X_k = \sum_{i=0}^{S-1} x_i \; w_i \; e^{-j\frac{2\pi ik}{S}} \qquad 0 \leq k \leq S-1 \tag{2.2}$$

where $X_1, X_2, \ldots, X_{S-1}$, are the complex DFT coefficients, and $w_1, w_2, \ldots, w_{S-1}$ are the coefficients of a window function applied to the samples to offset the spectral leakage caused by the finite length of the DFT. The spectrum of the signal is represented discretely by the magnitudes of the DFT coefficients, each of which corresponds to the signal energy in the frequency interval:

$$\frac{k-\frac{1}{2}}{S}\omega_s \;\leq\; \omega \;\leq\; \frac{k+\frac{1}{2}}{S}\omega_s$$

where ω_s is the sampling frequency[8]. Because the signal consists of a sequence of real numbers, the sequence of DFT coefficients is Hermitian, implying that:

$$|X_k| \;=\; |X_{S-k}|$$

The entire spectrum may therefore be represented by the first $\frac{S}{2}+1$ DFT coefficients, covering the frequency range, $0 \geq \omega \geq \frac{\omega_s}{2}$. Cepstral analysis proceeds by performing a second DFT on the log magnitudes of the original DFT coefficients. Because the original spectrum was Hermitian its magnitude spectrum is even. This enables the second DFT to be performed with the computationally simpler Discrete Cosine Transform (DCT):

$$C_i = \sum_{k=0}^{\frac{S}{2}} \log|X_k| \; \cos\frac{2\pi ik}{S} \qquad i = 0, 1, 2, \ldots \tag{2.3}$$

[8]Due to spectral leakage the X_k actually represent the signal energy contained in a small bandwidth surrounding the central frequency, the bandwidth depending on the nature of the window function employed.

The zeroth coefficient, C_0, is proportional to the average component of the log magnitude spectrum. Although this coefficient gives an indication of degree of voicing (voiced sounds have higher intensities than unvoiced sounds), it was not employed here as a recognition feature. Almost all the variance associated with the remaining coefficients is concentrated in the first eight or so, which form a compact representation of the vocal tract transfer function (see Appendix B). The method presented above produces what are known as linear frequency cepstral coefficients, because the original DFT coefficients are placed at linear intervals in frequency. Mel frequency cepstral coefficients can be obtained by imposing a set of band-pass filters on the DFT spectrum, with centre frequencies conforming to the mel frequency scale (see Figure 2.7). If there are P filters, and X'_k is the output of the kth filter

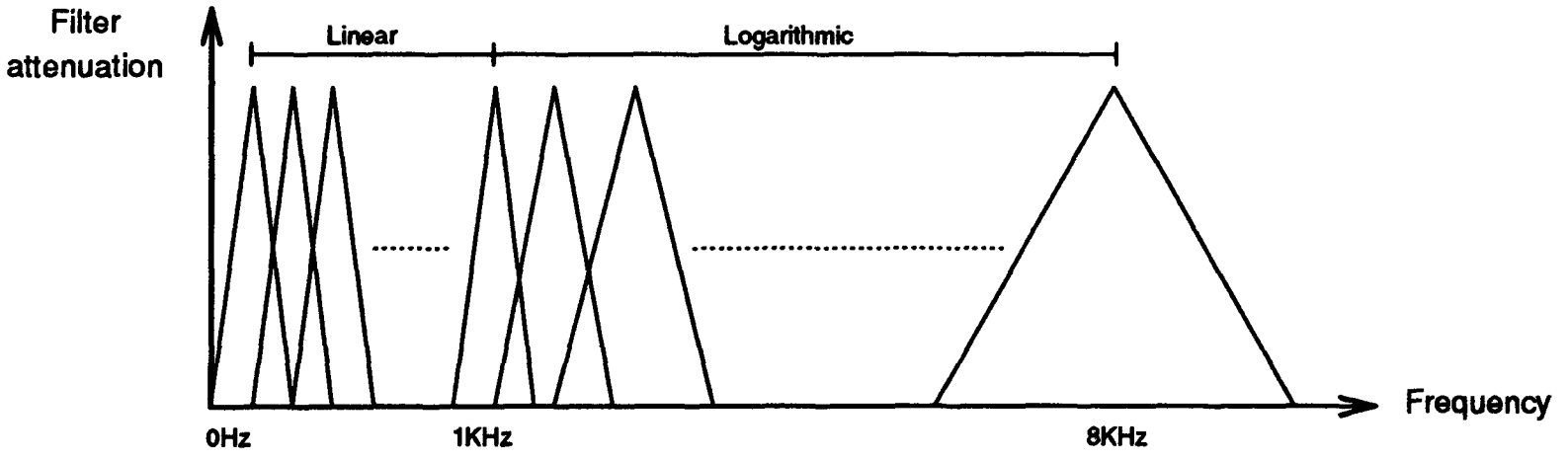


Figure 2.7: *Band-pass filters simulating the mel frequency scale. For simplicity the centre frequencies are spaced linearly up to 1kHz, and logarithmically beyond.*

(computed as the weighted average X_k over the kth filter bandwidth), the mel frequency cepstral coefficients can be derived as follows:

$$MC_i = \sum_{k=1}^{P} \log |X'_k| \ \cos i(k - \frac{1}{2})\frac{\pi}{P} \qquad i = 0, 1, 2, \ldots \tag{2.4}$$

Again the zeroth coefficient is unused.

It should be noted that the DFT was implemented with the Fast Fourier Transform

algorithm (FFT) for computational efficiency[9]. Frames consisted of 512 samples ($S = 512$), corresponding to a time interval of 25.6 ms (20 kHz sampling rate). The samples in each frame were multiplied by the Hamming window function:

$$w_i = 0.54 + 0.46\cos(\pi\frac{2i - S + 1}{S}) \qquad 0 \leq i \leq S - 1$$

Frames were overlapped by 256 samples to recover the spectra of the samples lost during windowing. Spectra were therefore computed every 12.8ms. A mel frequency scale was imposed on the spectrum by means of 26 triangular band-pass filters (see Figure 2.7), the outputs of which were used to compute eight mel frequency cepstral coefficients ($MC_1 \ldots MC_8$) as in Equation 2.4. These coefficients were taken as the final feature vector from which word templates are produced.

Linear Time Normalisation

We noted at the beginning of this section that the fixed dimension classifiers introduced thus far are well suited to isolated word recognition. However, the pre-processing strategy outlined above results in variable length word templates due to the variable duration with which words are uttered. Classification must therefore be preceded by a time normalisation process which transforms the variable length templates into a fixed input dimension (techniques which classify variable length templates directly will be examined in Chapter 6). The simplest method involves truncating all the variable length templates to the same length (usually the length of the shortest template). This method is only practical if all words are approximately the same duration, an unlikely situation in any useful vocabulary spoken by large numbers of speakers. A better method is linear time normalisation, which represents each variable length template by the same number of frames (see Figure 2.8).

[9]The number of complex multiplications required by the DFT is $(S-1)^2$, and for the FFT just $\frac{S}{2}\log_2 S$.

Fixed length templates consist of 15 frames, less than the 17 frames making up the shortest

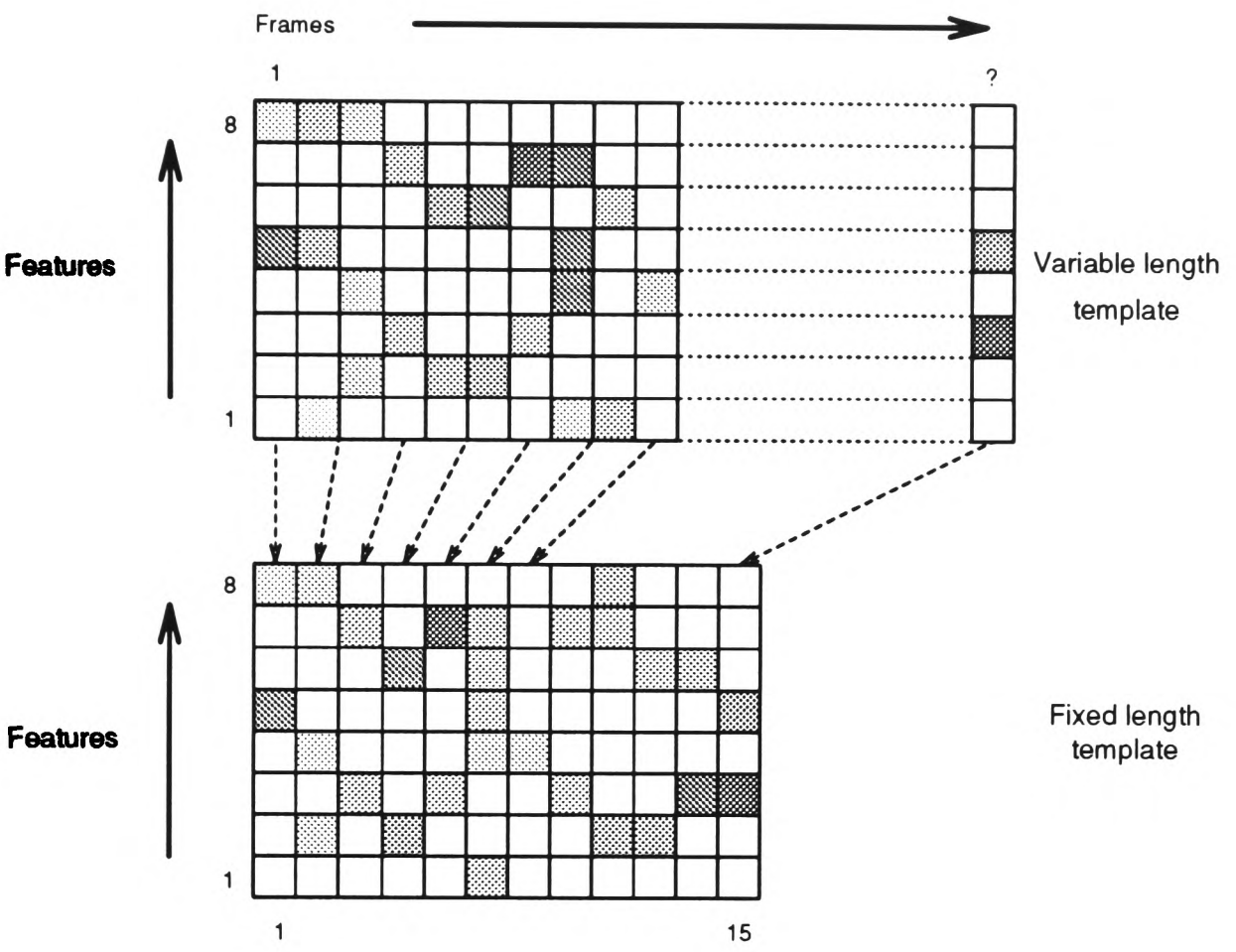


Figure 2.8: *Conversion of variable length template into fixed length template.*

variable length template in the database. All fixed length templates are therefore obtained by *compression*, minimising redundancy along the time axis. The first and last frames of the variable length template are preserved. The remaining 13 frames of the fixed length template are extracted by linear interpolation at regular intervals along the time axis of the variable length template. As each frame is represented by an eight-dimension feature vector, the total dimension of the fixed length template (and therefore the dimension of the input patterns) is given by:

$$N_I = 15 \times 8 = 120$$

2.4.2 Specification of Problems

The database comes already divided into a training set and a test set suitable for the *speaker-independent* spoken letter recognition experiments. Each set contains three utterances of the letters of the alphabet from 52 speakers, roughly balanced in sex and age. After pre-processing the training set amounts to 3999 input patterns and the test set to 3977 input patterns. Together, these sets of patterns are referred to as Problem A1 and are used extensively in the assessment of the static classifiers introduced in Chapter 1. Experiments were also conducted on subsets of this problem, referred to as Problems A2, A3, B, and C respectively. The specification of these problems is summarised in Table 2.2. The training

Table 2.2: *Specification of speaker-independent problems.*

Problem	Classes	# training speakers	# test speakers	# training patterns, M	# test patterns	F ratio
A1	'a–z'	52	52	3999	3977	1.2179
A2		14		1078		1.2024
A3		7		541		1.2251
B	'b,c,d,e,g,p,t,v'	52		1239		0.4138
C	'f,h,i,l,q,r,s,w'			1227		1.2874

sets of Problems A2 and A3 are approximately $\frac{1}{4}$ and $\frac{1}{8}$ of the size of the training set of Problem A1 However, the test sets for the three Problems are identical to ensure that all Problems are assessed with an equivalent degree of statistical validity. Problems B and C contain reduced sets of classes, notable for their different ease of discrimination. The former contains the very similar letters of the 'e' set, while the latter contains a set of letters chosen specifically for their mutual dissimilarity. This is illustrated by comparing their F ratios, a standard measure of the class separability in a problem based on its training data. It can

be stated qualitatively as:

$$F = \frac{variance\ of\ class\ means}{mean\ of\ class\ variances}$$

(an analytical result is given in Appendix C). A high value of F ratio indicates that the spread between the classes is *on average* greater than the spread inside the classes, although individual classes may still overlap. The F ratios of Problems A1–A3 are very similar since the training sets of Problems A2 and A3 are subsets of Problem A1. As expected, the F ratios of Problems B and C are lower and higher respectively.

A small *multiple-speaker* problem, referred to as Problem D, was also constructed from the database for the assessment of the dynamic classifiers introduced in Chapter 6. The problem comprised ten speakers chosen randomly from the database. Two out of the three utterances of the alphabet from each speaker were incorporated into its training set, and the remaining utterance into its test set. The specification of Problem D is summarised in Table 2.3. Note that the F ratio of Problem D is higher than any of the speaker-independent

Table 2.3: *Specification of multiple-speaker problem.*

Problem	Classes	# speakers	# training patterns, M	# test patterns	F ratio
D	'a–z'	10	520	260	1.5144

problems, demonstrating its greater simplicity.

2.5 Summary of Previous Research

Traditional static classifiers have frequently been applied to speech recognition problems in order to obtain performance benchmarks. Recently there has been a considerable amount of research comparing the performance of equivalent neural network classifiers with these

traditional classifiers [5, 36, 31]. An objective of this thesis is to compare the performance of the KNN and OLT traditional classifiers, with the RBF and MLP neural network classifiers on the CONNEX alphabet database. Woodland's results [61] constitute the most important existing study of speech recognition making use of the CONNEX alphabet database. By adopting Woodland's pre-processing technique as the basis of the technique outlined in Section 2.4.1, the results obtained for the static classifiers examined in this thesis are directly comparable with his results (it is often difficult to make meaningful comparisons between the results published by different authors owing to the non-conformity of speech pre-processing techniques used and the variability of problem specifications). Woodland assessed the performance of an MLP on a speaker-independent problem identical to Problem A1. After extensive optimisation, the error rate achieved by the MLP was 11.7%. Woodland also presented results for a state-of-the-art implementation of the dynamic time warping algorithm[10] on the same problem. The lowest error rate achieved by the latter was 14%. The consideration of this algorithm, and another dynamic classification algorithm known as a hidden Markov model, will be left until Chapter 6.

[10] Designed by British Telecom Research Laboratories.

Chapter 3

RBF Optimisation: Theory

3.1 Introduction

This chapter examines how the RBF introduced at the end of Chapter 1 may be optimised for maximum generalisation. The first objective must therefore be to highlight the parameters of the RBF which are most important in this respect. The output of the ith discriminant function in the RBF, upon presentation of the mth training pattern, $\mathbf{a}^m$, is given by:

$$o_i = \sum_{j=1}^{N_K} x_{ij}\Phi_j^m + x_{i0} \tag{3.1}$$

where Φ_j^m is the output of the jth non-linear kernel function, which in this thesis is a radially symmetric Gaussian function:

$$\Phi_j^m = \exp\left(-\frac{||\mathbf{a}^m - \mathbf{c}_j||^2}{2\sigma_j^2}\right) \tag{3.2}$$

Note that $\mathbf{c}_j$ is the kernel function centre, and σ_j its width parameter. The task of learning is to generate from the training data a set of discriminant functions which are a close approximation to the functions specified by the target outputs of the training patterns. This requires a hyperplane decision boundary to be constructed in the space defined by the kernel function outputs, separating one class of training patterns from the others. Generalisation can be interpreted as *interpolation* between training patterns after their projection into this space (of dimension N_K). It depends primarily on the *degree of fit* to the training

patterns in this space, which may be optimised by varying the number of kernel functions. Generalisation also depends on how well the centres of the kernel functions model the underlying distribution of the training data. Finally, generalisation depends on how localised or distributed the representation of the input patterns is after their projection into the space of the kernel functions. The 'locality' of representation will be the term used to describe this property. We shall see that the locality of representation may be optimised by varying the width parameters of the Gaussian kernel functions. In summary, generalisation depends on the following network parameters:

1. The number of kernel functions, N_K.

2. The locations of the centres, $\mathbf{c}_j$.

3. The kernel function width parameters, σ_j.

Each of the above is examined from a theoretical point of view, and a *strategy* for its optimisation is proposed.

3.2 Number of Kernel Functions

There is no analytical method for obtaining the value of N_K giving maximum generalisation in a particular problem. However, limits on N_K can be suggested by examination of the fitting process.

3.2.1 Underfitting and Overfitting

Consider the two-dimensional training patterns belonging to a simple two class problem. Figure 3.1a illustrates the decision boundary formed when the training patterns are *underfitted*, or too few kernel functions sample the input space. Since the responses of all kernel

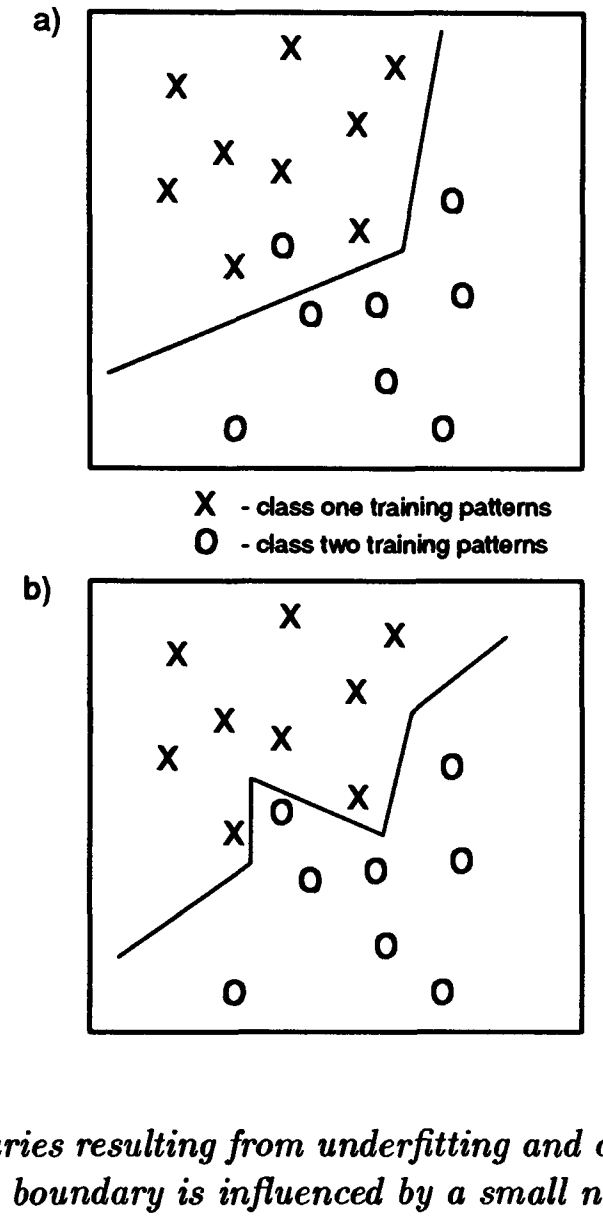


Figure 3.1: *Decision boundaries resulting from underfitting and overfitting. a) Underfitting. The position of the decision boundary is influenced by a small number of training patterns. Test patterns occurring in undersampled regions of input space are likely to be classified incorrectly. b) Overfitting. The position of the decision boundary is influenced to the locations of individual training patterns and their associated noise. Test patterns occurring in regions of input space unrepresented by training patterns are likely to be classified incorrectly.*

functions are negligible to training patterns occurring in undersampled regions of input space, such patterns are effectively excluded from the fitting process. The decision boundary is therefore placed independently of their locations, resulting in poor generalisation of test patterns occurring in undersampled regions. The input space becomes better sampled with increasing numbers of kernel functions, resulting in more complex decision boundaries that are influenced by the locations of greater numbers of training patterns. Generalisation will continue to improve under these conditions until the training patterns become

overfitted. Figure 3.1b illustrates the complex decision boundary resulting from overfitting. Note that its position is influenced by the locations of individual training patterns, and is therefore subject to the inconsistency, or *noise* associated with them. Such complex decision boundaries are not advantageous to generalisation as they are too dependent on the characteristics of the training set. Test patterns occurring in regions of input space which are not represented in the training set are likely to be classified incorrectly.

The fitting process can be examined in greater detail by considering the linear separability of the pattern classes, which, in a specific sense, depends on N_K. The maximum number of dichotomies of M patterns[1] is simply 2^M. However, the actual number of dichotomies capable of being formed by a linear discriminant function (i.e. the mapping from $\mathbf{\Phi}$ to $\mathbf{o}_i$ in this case) is also dependent on the dimensionality of the patterns, N_K. Cover [11] showed that it is given by:

$$\begin{aligned} C(M, N_K) &= 2^M && if\ M \leq N_K + 1 \\ &= 2 \sum_{k=0}^{N_K - 1} \binom{M-1}{k} && if\ M > N_K + 1 \end{aligned} \tag{3.3}$$

It is apparent that for fixed M, $C(M, N_K)$ rises as the pattern dimensionality is increased, *irrespective* of the locations of the patterns[2]. Although there is no explicit relationship between $C(M, N_K)$ and the linear separability of the classes, indirect correspondence can be assumed. It is therefore clear that for fixed M, the linear separability of the classes increases with N_K.

[1] All combinations of k patterns out of M, where $1 \leq k \leq M$.

[2] This observation is true provided that the patterns are in *general position*, or every subset of N_K or fewer patterns is linearly independent [11].

3.2.2 Proposed Optimisation Strategy

The optimum value of N_K is determined by two opposing constraints: N_K should be high to avoid underfitting, yet should not be so high as to cause overfitting. The optimum value of N_K can be determined empirically as follows:

1. Allocate a number of kernel functions small enough to guarantee that the training data will be underfitted.

2. Record the error rate on test data.

3. Add a kernel function and repeat step 2 until the error rate increases or remains constant.

In practice, more than one kernel function can be added at step 3 to speed up the optimisation process. As outlined, this optimisation strategy appears to break *rule number one* of pattern recognition: the test set should not be used to optimise classifier parameters. However, the method has become well established for the following reasons:

- The technique of cross-validation[3] is costly in both training data and training time.

- If the test set is sufficiently large it will represent a wide selection of possible patterns. Results obtained are therefore likely to be similar for another set of test patterns taken from the same source.

In a complex problem, the optimum value of N_K is likely to be the highest value that does not cause overfitting (the linear separability of the classes will then be at its highest useful

[3]Cross-validation is the technique whereby three sets of patterns are involved in evaluating classifier performance: a training set, a cross-validation set, and a test set. As usual the network parameters are trained using the training set, but are subsequently optimised subject to the minimisation of error rate on the cross-validation set. The test set is then used to evaluate network performance in the normal way.

level). An estimate of this value is the value of N_K at which the discriminant function reaches 'capacity', or $C(M, N_K)$ is half its maximum value. Cover [11] showed that this occurs when $\frac{M}{N_K} = 2$. The highest value of N_K which will not cause overfitting will therefore be approximately $\frac{M}{2}$.

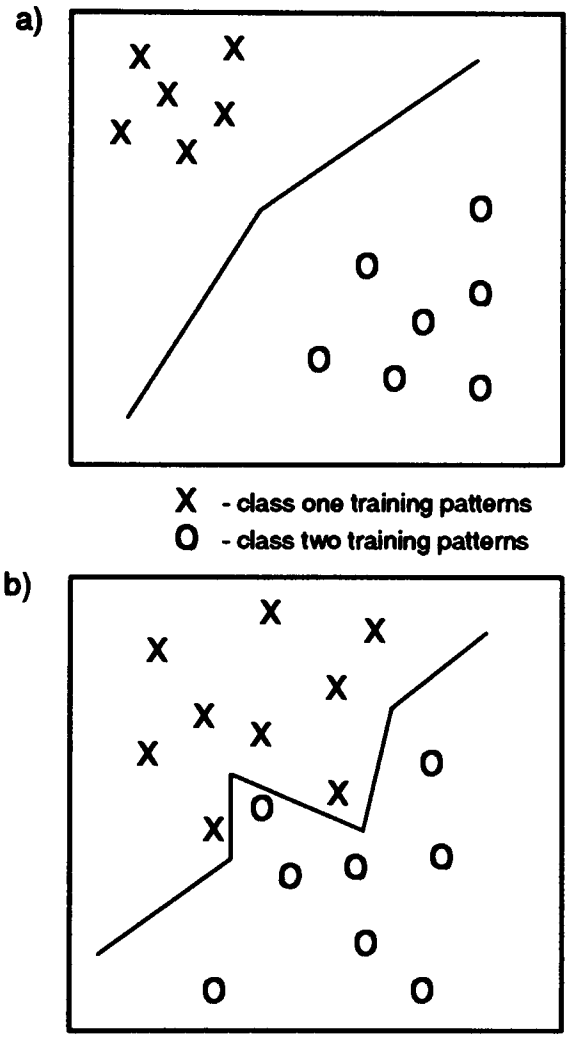


Figure 3.2: *Decision boundaries for simple and complex problems. a) Simple problem. Underfitting is overcome with fewer kernel functions, resulting in a simple decision boundary. Overfitting is unlikely to affect generalisation because the patterns are well separated from the decision boundary. b) Complex problem. Large numbers of kernel functions are required to overcome underfitting, resulting in a complex decision boundary. Generalisation is sensitive to the exact location of the decision boundary and is degraded by overfitting.*

3.2.3 Dependence of N_K on Complexity

Because the pattern classes of simple problems are *inherently* more separable than those of complex problems, less complex problems require fewer kernel functions (i.e. simpler decision boundaries will separate the classes). However, the number of kernel functions leading to overfitting is not dependent on problem complexity (being independent of the positions of the patterns), but its negative effect on generalisation is less noticeable in simpler problems because the patterns are well separated from the decision boundaries (see Figure 3.2).

3.3 Locations of the Centres

For good generalisation the centres of the kernel functions should evenly sample all regions of input space occupied by training data. The allocation of centres is a therefore a problem in subset model selection [9], in that the statistical characteristics of the training data should be reflected discretely in the centres. There are two widely used methods for achieving this, with a corresponding trade-off between training time and model accuracy.

3.3.1 Random Allocation

One solution is to choose randomly selected training patterns as the centres (to avoid duplication, no training pattern should be selected more than once). The number of centres is therefore limited by the size of the training set, M. This method has negligible training time, but there are two drawbacks:

1. Unless centres are allocated from a large pool of training data, random allocation is unlikely to form an accurate model of the data.

2. Random allocation results in inconsistent performance, because the accuracy of the model is dependent on a random process.

Obviously, the accuracy of the model of the training data formed by random allocation improves with increasing values of N_K. In the limit, $N_K = M$, one centre will be allocated for each training pattern, and clearly a more accurate model could not be obtained. However, it should be obvious from the preceding section that this number of kernel functions would lead to degradation of generalisation due to overfitting. Thus N_K would not be set equal to M in practice.

3.3.2 Adaptive Allocation

Random allocation makes no attempt at finding centres that accurately model the training data as a whole; a subset of patterns is selected and the rest ignored. Adaptive allocation algorithms adjust the positions of the centres subject to the minimisation of the average *distortion* between the centres and training patterns:

$$D = \frac{1}{M} \sum_{m=1}^{M} ||\mathbf{a}^m - \mathbf{c}_{j*}||^2 \tag{3.4}$$

where $\mathbf{c}_{j*}$ is the centre with minimum Euclidean distance to the mth training pattern:

$$||\mathbf{a}^m - \mathbf{c}_{j*}|| \leq ||\mathbf{a}^m - \mathbf{c}_j|| \qquad 1 \leq j \leq N_K \tag{3.5}$$

Minimising the distortion has the effect of generating a point density function for the centres which tends to approximate the probability density function of the training data [33], fulfilling the requirement of subset model selection. Since there is no analytic method for deriving the set of centres providing the minimum distortion, a *self-organising* algorithm can be used to adapt the centres iteratively until distortion converges to a local minimum.

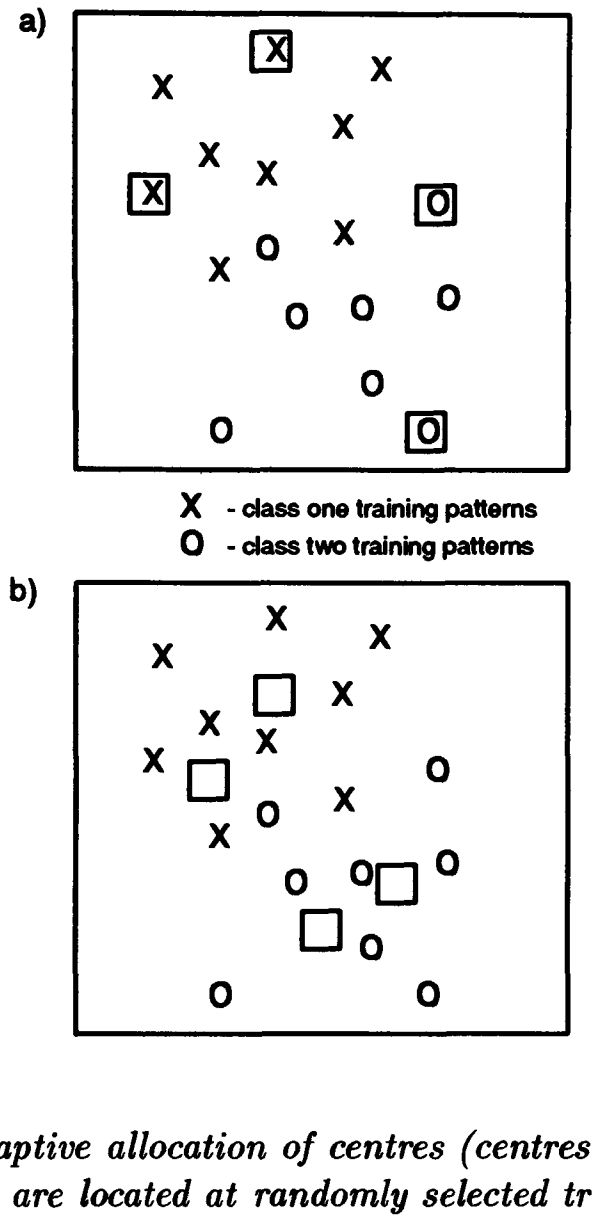


Figure 3.3: *Random and adaptive allocation of centres (centres are denoted by boxes). a) Random allocation. Centres are located at randomly selected training points. b) Adaptive allocation. Centres are located at positions which model the training data as a whole. Note that the centres define two clusters corresponding roughly to the classes of training patterns.*

The advantage of adaptive allocation over random allocation is clearly illustrated in Figure 3.3. The centres obtained using the adaptive algorithm do not coincide with particular training patterns, as occurs with random allocation. Instead, the centres 'move' into positions in which the distribution of the training data is better modelled. Random allocation is an obvious method of allocating the centres prior to adaptation, but in general, *any* initial set of centres can be used. The two most widely used adaptive allocation algorithms are based on the principle of *clustering* (see Chapter 6).

Adaptive K-means Algorithm

The original K-means clustering algorithm is an iterative procedure. K centres are initially allocated randomly, and the Euclidean distance between each training pattern and each centre computed. Each training pattern is labelled with the index of the centre it was closest to, and the position of each centre is then *recomputed* as the average of those training patterns bearing its label. The process is applied repeatedly to each new set of centres, until the absence of adaptation signifies that the distortion is locally minimal.

The same algorithm can be formulated as an adaptive process [39], in which adaptation takes place *per pattern*, not *per epoch* as above. Training patterns are selected randomly from the training set, and the Euclidean distance to each centre computed. In a 'winner-take-all' strategy the closest centre, $\mathbf{c}_{j^*}$, is adjusted *towards* the training pattern and the other centres remain unmodified:

$$\begin{aligned} \mathbf{c}_j(t+1) &= \mathbf{c}_j(t) + \eta\,(\mathbf{a}^m - \mathbf{c}_j) \qquad if \;\; j = j^* \\ \mathbf{c}_j(t+1) &= \mathbf{c}_j(t) \qquad\qquad\qquad\qquad otherwise \end{aligned} \tag{3.6}$$

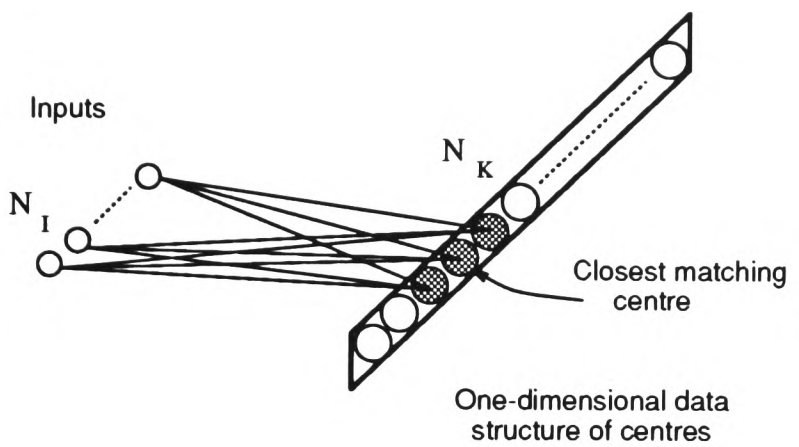
The total number of adaptive steps required, T, depends only on the number of centres, N_K, and the extent to which the distortion is to be minimised [29]. It does not depend on the dimensionality of the input patterns, N_I, nor the size of the training set, M [29]. In contrast to the original K-means algorithm, the distortion will not reach a definite minimum, but its rate of descent will slow down sufficiently for convergence to be detected. In the absence of a formal method for predicting the necessary number of adaptive steps, heuristic methods have been employed. One such method suggests using $T = 500 \times N_K$, to construct an accurate model of the training data [29]. However, less accuracy (and therefore less adaptive steps) might be tolerable for highly variable data sources such as human speech

utterances – there would be little point in accurately modelling a set of M training patterns when an alternative set could be derived with different statistical properties. The number of adaptive steps required is also affected by the learning parameter, η, which may be fixed at some small value throughout adaptation, or decay during the course of adaptation. If η is too high (say > 0.75) instability can occur and convergence may be prevented altogether. If η is too low (say < 0.01) then convergence will be slowed down, causing long training times or poorly tuned centres if training is terminated early. Within these bounds the value of η only affects rate of convergence, not the solution eventually obtained.

Feature Map Algorithm

Kohonen's feature map [28] was originally developed in an attempt to model the two-dimensional *topology preserving* maps in the cortices of higher animal brains. These maps have the interesting property that spatial proximity in the neurons indicates physical similarity between the stimuli to which they respond. Information is thought to be ordered spatially across the map surface to simplify connectivity to higher levels of cortical processing. The feature map does indeed capture this property of real cortical maps, offering a novel representation of data sources. However, its basic operation is very similar to other self-organising algorithms such as the adaptive K-means algorithm.

Figure 3.4 shows how the set of N_K centres can be arranged into an arbitrarily shaped data structure, or *map*, in one or two dimensions (the two-dimensional maps used in this thesis were square, limiting the choice of N_K to numbers with integer square roots). The locations of the centres are adapted using a modified version of Equation 3.6, which not only adapts the location of the closest centre, $\mathbf{c}_{j^*}$, but also those centres inside a *neighbourhood* defined around this centre in the map. This is referred to as 'soft' competition [40]:



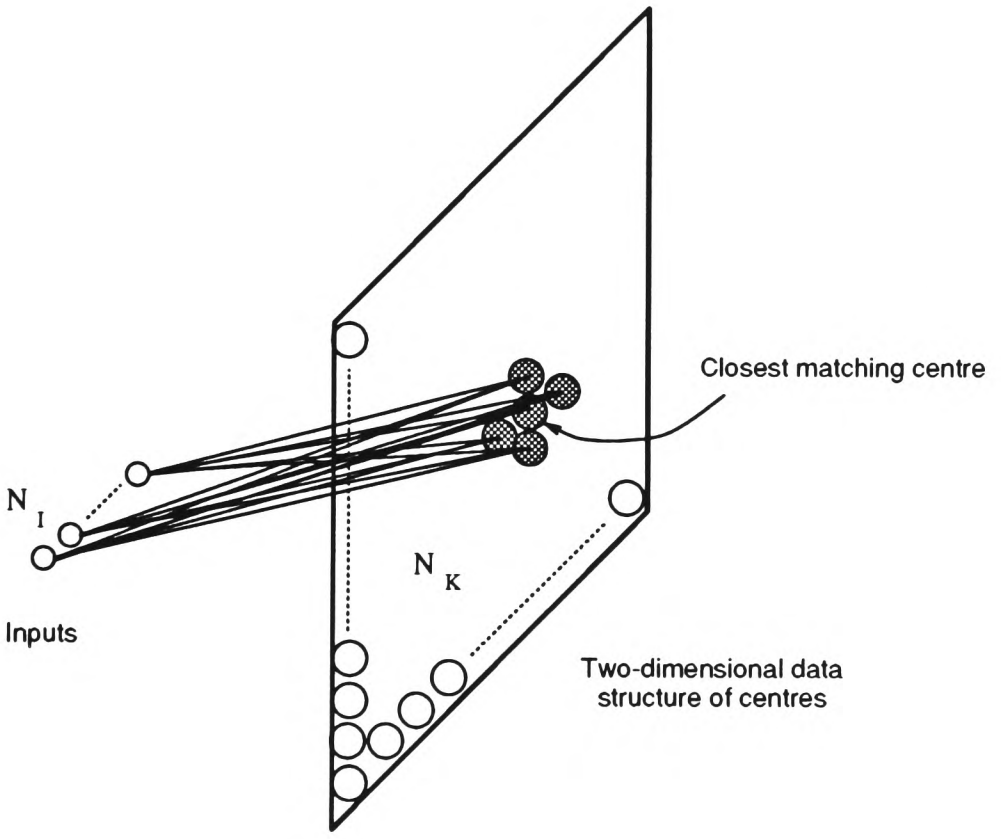


Figure 3.4: *Structure of one and two-dimensional feature maps. The highlighted centres depict a neighbourhood surrounding the closest matching centre to the input pattern.*

$$\begin{aligned} \mathbf{c}_j(t+1) &= \mathbf{c}_j(t) + \eta\,(\mathbf{a}^m - \mathbf{c}_j) \qquad if \;\; j \in N_{j^*}(t) \\ \mathbf{c}_j(t+1) &= \mathbf{c}_j(t) \qquad\qquad\qquad\qquad otherwise \end{aligned} \tag{3.7}$$

The direction in which the centres are adapted is therefore dependent on their closeness in Euclidean distance to training patterns, *and* their position in the data structure. The latter constraint forces the map to become ordered spatially (centres occurring in similar regions of input space are arranged near to each other in the map). The neighbourhood function, $N_{j^*}(t)$, initially includes most centres in the map, so that ordering occurs on a global scale. However, during the course of adaptation the neighbourhood function decays until it includes only the single centre, $\mathbf{c}_{j^*}$, at which point it remains fixed for the remainder of adaptation. It is apparent that the feature map trains in two clearly defined stages, the first requiring approximately an order of magnitude less adaptive steps than the second:

1. The 'soft' stage during neighbourhood decay when centres become spatially ordered.
2. The 'winner-take-all stage' in which centres are fine-tuned to minimise distortion (identical in operation to the adaptive K-means algorithm).

In the case of the one-dimensional map, the neighbourhood can simply be defined as those centres occurring within a certain radius, $R(t)$, of the closest matching centre in the map:

$$\begin{aligned} j &\in N_{j^*}(t) \qquad if |j - j^*| \leq R(t) \\ j &\notin N_{j^*}(t) \qquad otherwise \end{aligned} \tag{3.8}$$

Global ordering can be obtained if the initial neighbourhood radius, $R(0)$, encompasses the majority of the centres in the map. For example, a value of $\frac{3}{4}N_K$ would be suitable.

Various neighbourhood are commonly defined in the two-dimensional feature map, three of which are shown in Figure 3.5. The hexagonal and circular neighbourhoods decay in

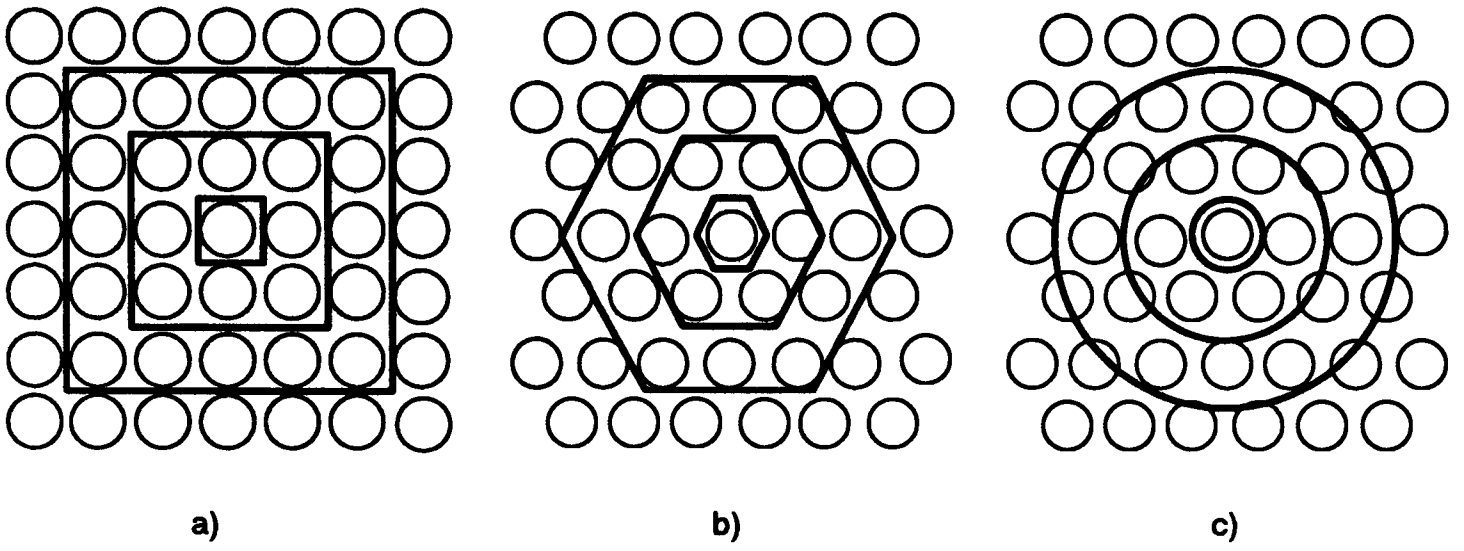


Figure 3.5: *Neighbourhood shapes for two-dimensional feature map. a) Square neighbourhood. b) Hexagonal neighbourhood. c) Circular neighbourhood. Note the interleaved rows of centres in b) and c).*

smaller steps than the square neighbourhood, providing a slightly reduced final distortion. In fact, the hexagonal and circular neighbourhoods are functionally equivalent because the hexagon makes a perfect approximation to a circle when the rows of centres are interleaved as shown in Figure 3.5b and c. However, the circular neighbourhood is favoured since it is simpler to implement in software: if $r(j)$ is the geometric distance (in the map) between the jth centre and the closest matching centre, then:

$$j \in N_{j^*}(t) \qquad if\ r(j) \leq R(t)$$

$$j \notin N_{j^*}(t) \qquad otherwise$$

A suitable initial neighbourhood radius would be the length of the longest side of the map. For a square data structure this is simply $\sqrt{N_K}$. A neighbourhood radius decay regime suitable for both one and two-dimensional feature maps, would decrease the initial radius

to zero (single centre adaptation) linearly over a small fraction of the adaptation time (see Figure 3.6). The two components of Equation 3.7 can be replaced by a single equation with

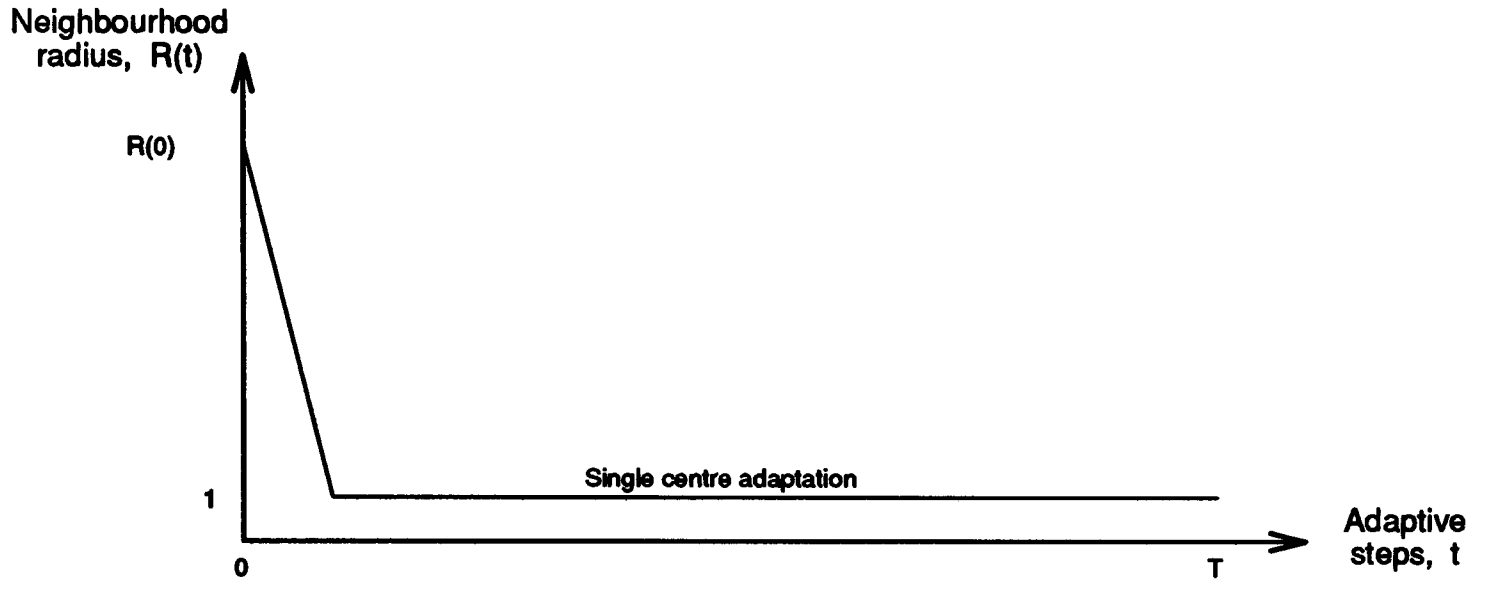


Figure 3.6: *Neighbourhood radius decay regime for both one and two-dimensional feature maps.*

the inclusion of a *spatial* (not temporal) neighbourhood decay function, $\beta(r)$:

$$\mathbf{c}_j(t+1) = \mathbf{c}_j(t) + \eta\, \beta(r)\, (\mathbf{a}^m - \mathbf{c}_j) \tag{3.9}$$

In the original feature map $\beta(r)$ is a threshold function at the neighbourhood radius:

$$\begin{aligned} \beta(r) &= 1 \qquad if \;\; r(j) \leq R(t) \\ &= 0 \qquad otherwise \end{aligned} \tag{3.10}$$

However, a small reduction in distortion can be obtained by redesigning $\beta(r)$ as a *ramp* function, or better as a *centre-surround* function [4] (see Figure 3.7). By eliminating the discontinuity of the threshold function, these functions reduce the likelihood that centres in close proximity to one another in the map will be adjusted in opposing directions. It is interesting to note that neurons in real cortical maps implement a quite similar function to the centre-surround via excitatory and inhibitory lateral connections [27].

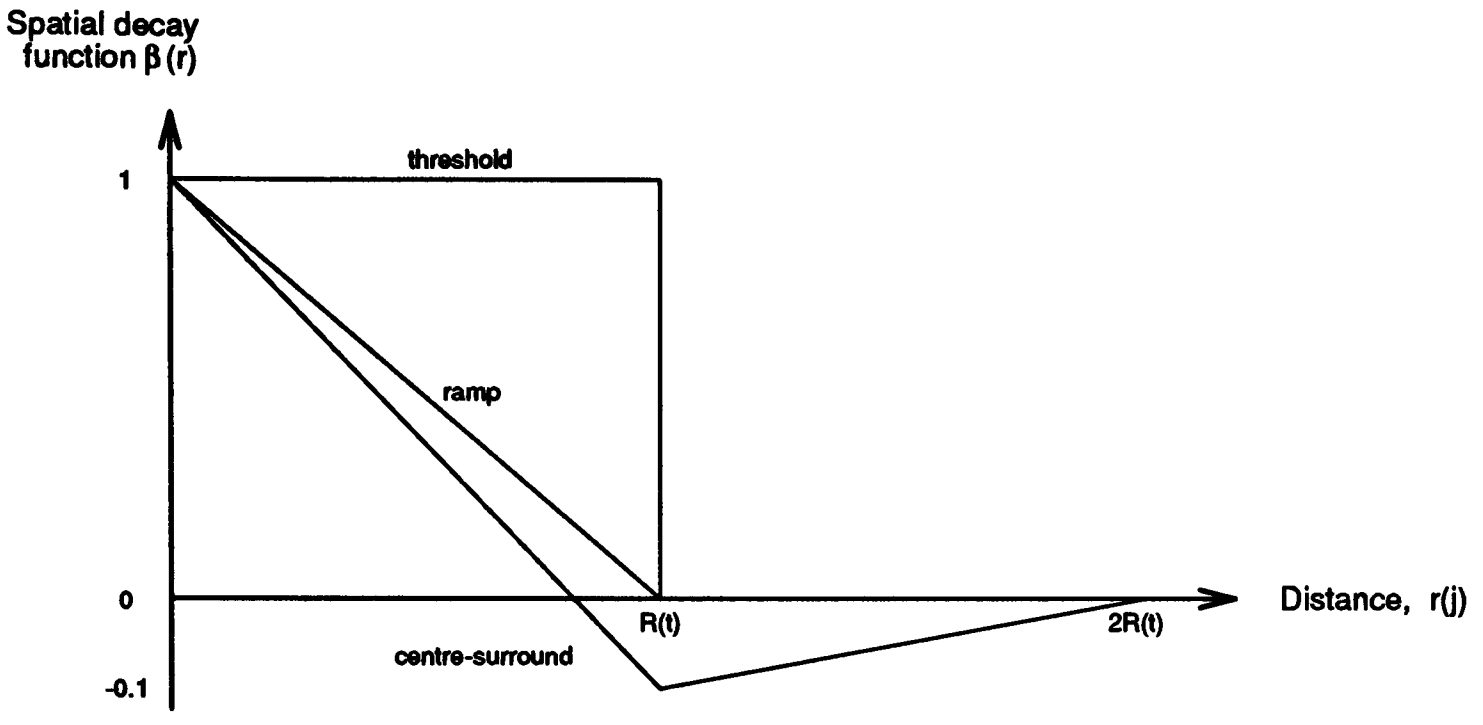


Figure 3.7: *Spatial neighbourhood decay functions.*

Moody and Darken [39] suggest that the feature map makes an 'intrinsic assumption about the underlying dimensionality of the problem', and that since no such assumption is made by the adaptive K-means algorithm, the latter will form a more ideal model of the training data. This claim stems from the mistaken assumption that the dimensionality of the map constrains the mobility of the centres in input space. In the context of the adaptive K-means algorithm, the arrangement of the centres in the map does not depend on their locations in input space. But in the feature map adaptation attempts to order the centres in the map according to their location in input space. We have seen that the ordering process occurs only in the preliminary stages of adaptation, after which the operation of the feature map becomes identical to that of the adaptive K-means algorithm. Ordering will therefore only affect the topological arrangement of the centres within the map, not their final locations in input space.

It is impossible to maintain topological correspondence between data sources when they

are expressed in different dimensions[4]. A 1:1 correspondence between distance in the map and distance in input space could only be expected if the underlying dimensionality of the data source were the same as the dimension of the map. For instance, the ordering of a two-dimensional feature map trained on a database of vowel sounds represented by their first two formant frequencies, has a close resemblance to the vowel quadrilateral developed by phoneticians [55]. The fact that the input dimensionality was two is not the key consideration here; if the vowels had been represented by the outputs of a 19 channel filter bank the vowel quadrilateral would still have been visible. The underlying dimensionality of speech is a complex issue. Certainly vowels can be represented well in a two-dimensional map, but the inclusion of nasals and consonants produces serious distortion. Higher dimension maps could be constructed to represent these sounds in a topologically correct manner, but only one, two, and three-dimensional maps are interpretable by humans visually. Of these the two-dimensional map is preferred for its simple exposition of data sources on a computer screen.

3.3.3 Proposed Optimisation Strategy

The choice of method to allocate centres depends on the answers to the following questions, which must be discovered empirically by evaluating the random, adaptive K-means, and feature map algorithms on the same classification problem:

- Can the error rate resulting from random allocation be reduced with an adaptive allocation algorithm ?
- Is the reduction justified given the associated increase in training time ?

[4] For example, when a video camera converts a two-dimensional image into a one-dimensional signal, the signal represents discontinuous segments of the image.

- Is the error rate resulting from adaptive allocation more consistent than for random allocation ?
- Does the adaptive K-means algorithm provide a lower error rate than the feature map algorithms by developing without spatial ordering ?
- Does the two-dimensional feature map offer a lower error rate than the one-dimensional feature map by relaxing the constraint of spatial ordering ?

The answers to these questions will be presented in the following chapter.

3.4 Kernel Function Widths

The Gaussian kernel function is an example of a receptive field, responding most strongly to input patterns occurring in the same region of input space as its centre (see Figure 3.8). The

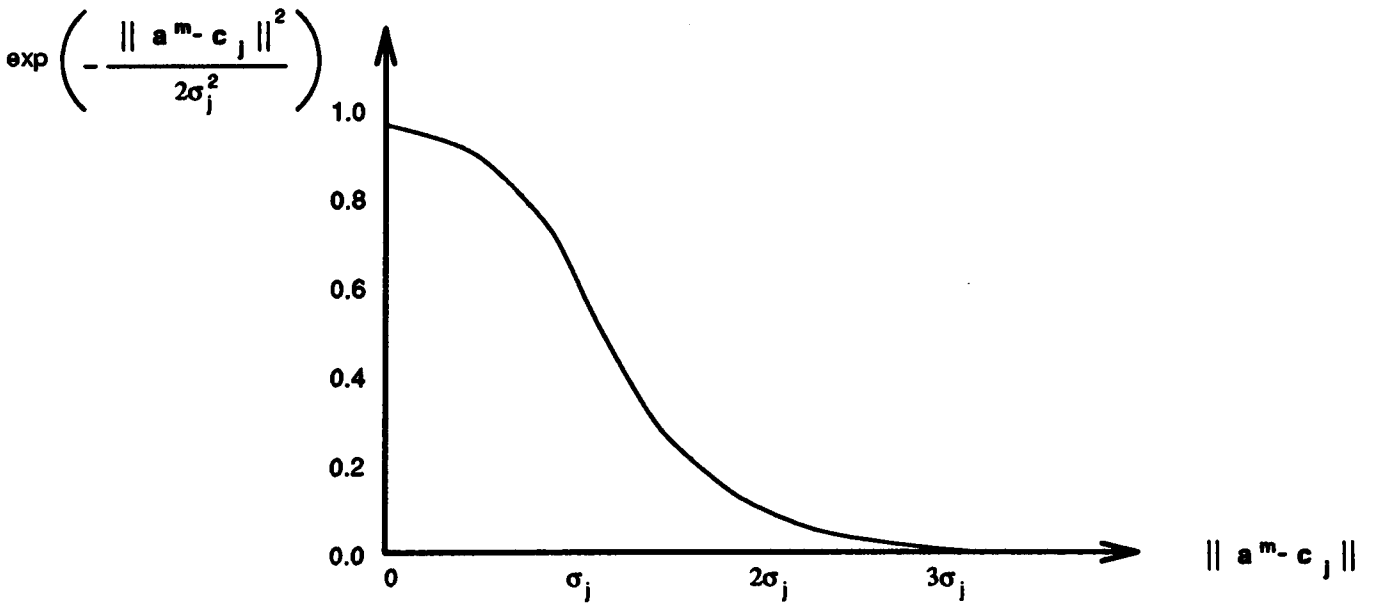


Figure 3.8: *Radially symmetric Gaussian kernel function.*

locality with which input patterns are represented by the kernel function outputs (the extent to which they respond to input patterns outside the immediate vicinity of their centres) is determined by the widths of the Gaussian kernel functions, which are normally selected

by one of several heuristic methods [36, 39, 46]. Such methods assume that the locality of representation is not critical to generalisation, provided that it is maintained within certain limits (see below). A fully adaptive optimisation method has also been suggested [36] which unfortunately suffers from greatly increased training time. These alternative strategies are briefly reviewed below.

3.4.1 Heuristic Methods

The second layer weights of the RBF are computed by inverting the matrix of kernel function outputs, $\mathbf{\Phi}$:

$$\mathbf{x}_i = \mathbf{\Phi}^+ \mathbf{b}_i \tag{3.11}$$

where $\mathbf{x}_i$ is the weight vector to the ith discriminant function, and $\mathbf{b}_i$ is its vector of target outputs. Although $\mathbf{\Phi}$ can be inverted when it is rank-deficient (using singular value decomposition), a better solution will be obtained if it is full rank. This can be ensured if kernel function widths are selected subject to the following considerations[5]:

1. If the kernel function widths are too wide, their responses become insensitive to the differences between input patterns.

2. If the kernel function widths are too narrow, their responses will be strictly localised, with the result that input patterns occurring in some regions of input space may cause no kernel functions to respond.

[5]Due to the finite accuracy with which numbers can be stored in a digital computer, any two kernel function outputs whose difference is less than that which can be represented will assume the same value, with consequent loss of information.

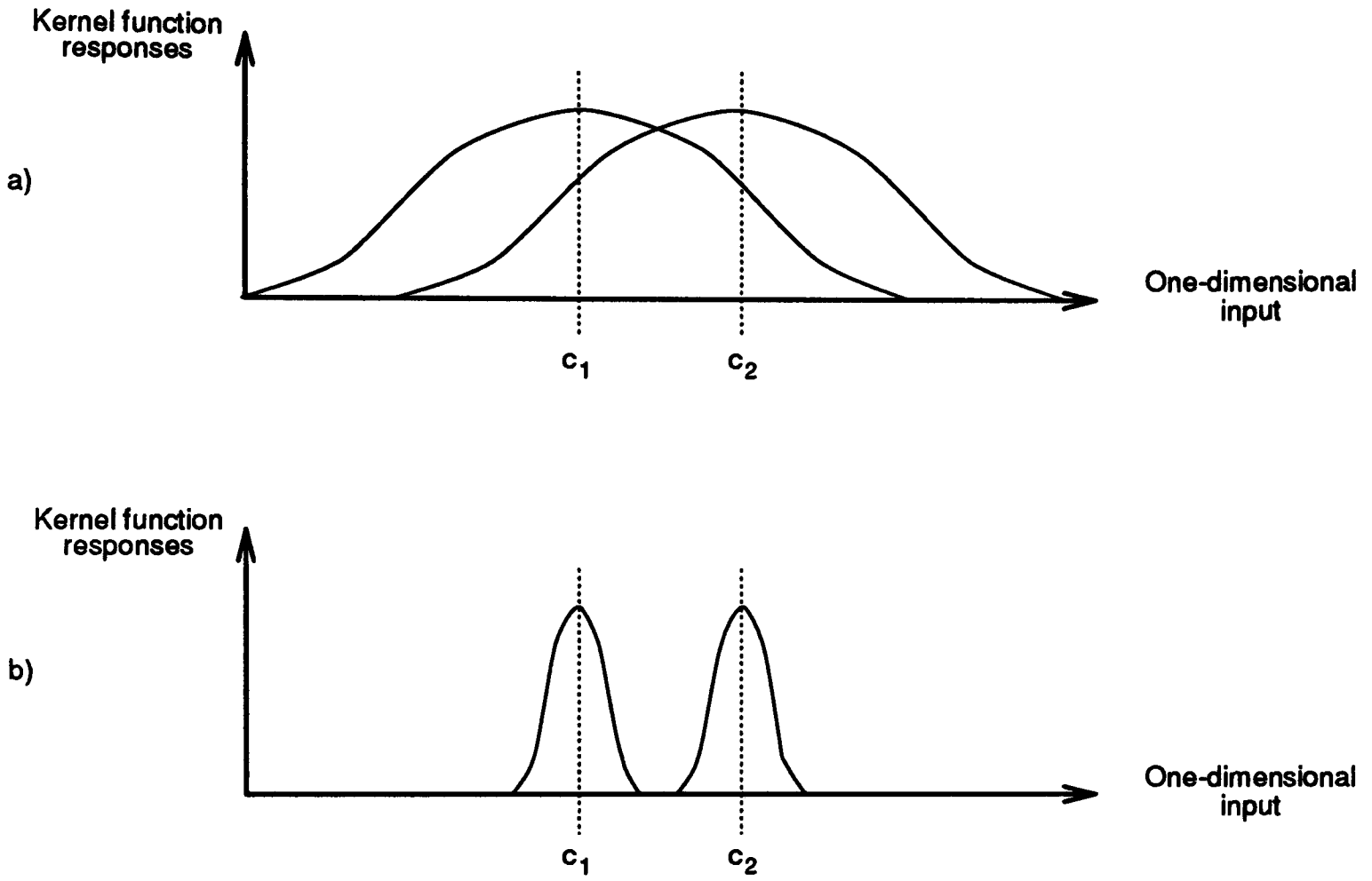


Figure 3.9: *Responses of two Gaussian kernel functions to one-dimensional input patterns. The centres of the kernel functions are labelled c_1 and c_2. a) Wide function widths. Input patterns occurring in the region between c_1 and c_2 produce very similar responses. b) Narrow function widths. Input patterns occurring between the centres produce no response at all.*

First Nearest Neighbour Heuristic

The width of the jth Gaussian kernel function is computed as the Euclidean distance between its centre, $\mathbf{c}_j$, and the nearest neighbour to its centre, $\mathbf{c}_{j^*}$:

$$\sigma_j = ||\mathbf{c}_{j^*} - \mathbf{c}_j|| \tag{3.12}$$

where:

$$||\mathbf{c}_{j^*} - \mathbf{c}_j|| \leq ||\mathbf{c}_k - \mathbf{c}_j|| \qquad 1 \leq k \leq N_K, \quad k \neq j \tag{3.13}$$

Global Farthest Neighbour Heuristic

The Euclidean distances between each centre and its farthest neighbour are computed, and the maximum value recorded. This value is then divided by the total number of centres, N_K, yielding a global width, σ, common to all kernel functions:

$$\sigma = \frac{1}{N_K} ||\mathbf{c}_{j\times} - \mathbf{c}_j|| \tag{3.14}$$

where $\mathbf{c}_{j\times}$ is the farthest neighbour to $\mathbf{c}_j$:

$$||\mathbf{c}_{j\times} - \mathbf{c}_j|| \geq ||\mathbf{c}_k - \mathbf{c}_j|| \qquad 1 \leq k \leq N_K \tag{3.15}$$

Global First Nearest Neighbour Heuristic

This heuristic computes the average width obtained using the First Nearest Neighbour heuristic, yielding a global width, σ, common to all kernel functions:

$$\sigma = \frac{1}{N_K} \sum_{j=1}^{N_K} ||\mathbf{c}_{j*} - \mathbf{c}_j|| \tag{3.16}$$

Due to the averaging process this heuristic provides a consistent value of function width even for centres derived using a random allocation.

3.4.2 Adaptive Method

The Gaussian kernel function of Equation 3.2 can be rewritten as:

$$\Phi_j^m = \exp\left(-\frac{1}{2}(\mathbf{a}^m - \mathbf{c}_j)^T \Sigma_j^{-1} (\mathbf{a}^m - \mathbf{c}_j)\right) \tag{3.17}$$

where $\sum_j$ is the *covariance matrix* associated with the jth kernel function. Radial symmetry requires that $\sum_j$ is diagonal, with each diagonal term equal to σ_j^2. However, if the constraint of radial symmetry is removed, each element of $\sum_j$ may adopt a different value. Lowe [36] outlined an adaptive non-linear optimisation method in which both the positions of the

centres and the elements of the covariance matrix are optimised so as to minimise the squared error between the actual and target outputs of the network over the training set. For the ith discriminant function this error is given by:

$$e_i = \sum_{m=1}^{M} \left(b_i^m - \sum_{j=0}^{N_K} x_{ij} \Phi_j^m \right)^2 \tag{3.18}$$

Since Φ_j^m is a continuous function it is possible to derive the partial derivatives of the squared error with respect to the centres, $\mathbf{c}_j$, and the inverse covariance matrices, $\sum_j^{-1}$ [36]:

$$\frac{\partial e_i}{\partial \mathbf{c}_k} = -2 \sum_{m=1}^{M} \left[\left(b_i^m - \sum_{n=0}^{N_K} x_{in} \Phi_n^m \right) \sum_{j=0}^{N_K} x_{ij} \frac{\partial \Phi_j^m}{\partial \mathbf{c}_k} \right] \tag{3.19}$$

$$\frac{\partial e_i}{\partial \sum_k^{-1}} = -2 \sum_{m=1}^{M} \left[\left(b_i^m - \sum_{n=0}^{N_K} x_{in} \Phi_n^m \right) \sum_{j=0}^{N_K} x_{ij} \frac{\partial \Phi_j^m}{\partial \sum_k^{-1}} \right] \tag{3.20}$$

Lowe used a second order gradient descent technique (the BFGS method) as the tool to minimise these expressions. The adaptation of the function widths was *slaved* to that of the centres; upon each adaptation of the centres, the function widths were adapted several times until a local minimum squared error solution was obtained. The centres were then adapted again and so on. A typical RBF examined by Lowe contained about 30 kernel functions. Optimisation required between 5 and 30 adaptations of the centres, each one requiring about 10 adaptations of the function widths in order to maintain a local minimum squared error. Since each adaptation requires recomputing the second layer weights, training time increased by a factor of between 50 to 300 times. This may be tractable for small networks ($N_K \leq 50$), but becomes impractical for networks of the size examined in this thesis ($64 \leq N_K \leq 1024$).

3.4.3 Proposed Optimisation Strategy

The disadvantages of the heuristic and adaptive methods are summarised below:

- The heuristic methods do not optimise the function widths. They merely constrain them to lie within certain rank-preserving limits.

- The adaptive method requires excessive training times for networks with large numbers of kernel functions (see above). Further, the local minimum squared error solution provided by the adaptive method does not ensure optimal generalisation due to the varying probability densities of the classes (see Appendix A).

Although the use of cross-validation could provide a solution to the second problem associated with the adaptive method [36] [34], the increase in training time would not be acceptable. For this reason a *compromise* optimisation method is proposed in this thesis in which the function widths are optimised in a manner similar to the selection of N_K. Specifically, the width parameter of Equation 3.2 can be modified to:

$$\sigma_j^2 = 2^L \times \sigma_{j*}^2 \tag{3.21}$$

where L is the 'locality index', and σ_{j*} is the heuristically determined function width. By varying the integer L discrete changes to the locality of representation can be imposed on the kernel function responses (note that $L = 0$ corresponds to the widthdetermined by the GFNN method). The best value of L can be discovered with the following optimisation strategy:

1. Derive a first set of function widths using a heuristic method.

2. Record the error rate on test data.

3. Increment L and repeat 2 until the error rate increases or remains constant.

Optimisation by the locality index method is not as sophisticated as the adaptive method described in Section 3.4.2, since the widths of all N_K kernel functions are identical, and increase by factors of $\sqrt{2}$ with each increment in L. It is therefore extremely unlikely that a truly optimal set of function widths will be discovered. However, the locality index method

offers a degree of flexibility not provided by the one-step heuristic methods. The flexibility is achieved without incurring the high training time penalty of a fully adaptive method, allowing it to be applied easily to large networks. Because the second layer weights must be recomputed to assess error rate for each value of L, training time is increased by the number of values of L tested. In a typical optimisation L would take the values $0, 1, 2, 3, 4$ (equivalent to multiplying the function widths by $1, 2, 4, 8, 16$), resulting in a five-fold increase in training time.

Chapter 4

RBF Optimisation: Results

4.1 Introduction

The purpose of this chapter is to introduce empirical results to demonstrate the effectiveness of the optimisation strategies proposed in the previous chapter, and to illustrate their underlying mechanisms. Experiments were conducted on problems designed using the CONNEX alphabet database, as described in Section 2.4.2. They took the form of software simulations on a 25 MHz SUN workstation. The error rates appearing in this chapter are average values computed over five independent trials, and the second layer weights were computed using the singular value decomposition technique. Table 4.1 describes each of the forthcoming experiments, indicating which variables are constant and which to be optimised in each case.

4.2 Experiment 1: Optimising the Value of N_K

The optimisation of N_K was investigated by evaluating error rate over a wide range of values of N_K, providing conditions of underfitting, optimal fitting and overfitting. In addition to Problem A1 (the full database), the results presented in Figure 4.1 were obtained on Problems A2 and A3 (the training sets of which are $\frac{1}{4}$ and $\frac{1}{8}$ the size of Problem A1 respectively) to provide different numbers of training patterns at each value of N_K. N_K is increased from 32 to 1024 in discrete steps, allowing a wide range of values of N_K to

Table 4.1: *Description of optimisation experiments.*

Experiment	Description	Problems studied	Allocation algorithm	N_K	σ_j
1	Optimising the value of N_K	A1–A3	adaptive K-means	32–1024	fixed†
2	Dependence of N_K on complexity	B,C	adaptive K-means	32–512	fixed†
3	The location of the centres	A1	variable	64,256	fixed†
4	Optimising the value of L	A1–A3,B,C	adaptive K-means	32–1024	variable

(† after initialisation with the global nearest neighbour heuristic and optimisation with the locality index method)

be assessed quickly. For the present purposes there is no need to investigate intermediate values of N_K, since the dependence of error rate on N_K is clearly very smooth. The adaptive K-means algorithm was used to allocate centres (for a justification of the use of this algorithm see Section 4.4), and kernel function widths were optimised with the locality index method. In all three problems error rate fell rapidly as N_K was increased from its initial value of 32, indicating that the training data was initially underfitted. The decline in error rate became less substantial at higher values of N_K where the training data was better fitted. In Problems A2 and A3, error rates rose when N_K was increased beyond 768 and 384 respectively, indicating that the training data was overfitted. In Problem A1, error rate continued to decline until N_K reached its final value of 1024, indicating that overfitting was never reached (unfortunately it was not possible to investigate higher values of N_K due to limited computer resources). Table 4.2 summarises the optimal range of

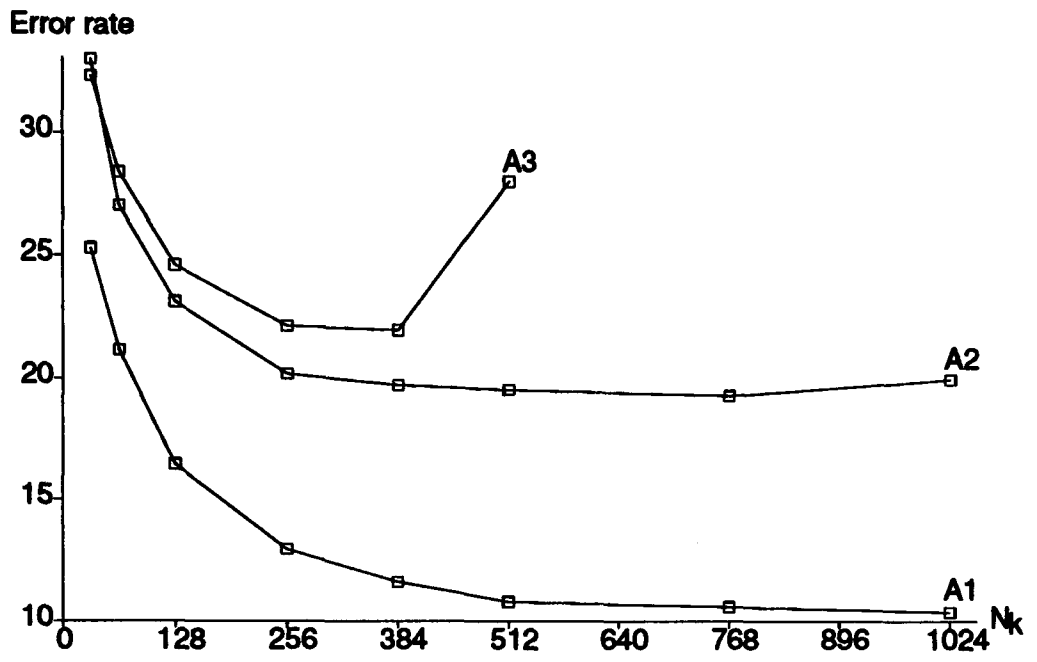


Figure 4.1: *Error rate vs N_K for Problems A1, A2 and A3.*

values of N_K for the three problems (the lower values produce underfitting, and the higher values overfitting), and compares them with the value of $\frac{M}{2}$, which was shown in Section 3.2.1 to be a theoretical estimate of its optimal value. It can be seen that there is a good

Table 4.2: *Optimal values of N_K for Problems A1–A3.*

Problem	Optimal value of N_K	$\frac{M}{2}$
A1	> 1024	2000
A2	$\geq 384 \leq 768$	539
A3	$\geq 256 \leq 384$	271

correspondence between between the observed optimal values of N_K and the value of $\frac{M}{2}$ for Problems A2 and A3. However, in Problem A1 the value of $\frac{M}{2}$ is 2000. This is clearly greater than the maximum value of N_K tested, and thus overfitting was unlikely to be

observed.

Table 4.3: *Lowest error rates achieved in Problems A1–A3.*

Problem	Lowest error rate
A1	10.38%
A2	19.34%
A3	21.98%

Table 4.3 shows the lowest error rates achieved in the three problems. The error rate obtained in Problem A1 was the lowest by a significant margin. This was to be expected because Problem A1 contains four and eight times as much training data as Problems A2 and A3 respectively. Further reductions in error rate would undoubtably follow if the CONNEX alphabet database was enlarged. It is interesting to note that the values of N_K yielding optimal fitting in this thesis were higher than those reported by several other authors on different problems. For instance Renals and Rowher [46] used values of N_K in the range 64–256 to tackle a vowel recognition problem for which $M = 750$. In this case the theoretical estimate for the optimal value of N_K is $\frac{750}{2} = 375$, which is slightly higher than the maximum value of N_K employed. However, their results show that error rate dropped steadily as N_K increased, with the minimum error rate occurring at $N_K = 256$. This suggests that further reductions in error rate may have been possible if higher values of N_K had been investigated. A similar conclusion may be drawn from the results of Moody and Darken [39]. They used values of N_K in the range 20–100 to tackle a different vowel recognition problem for which $M = 338$. In this case the theoretical estimate for the optimal value of N_K is 169, higher than the maximum value of N_K employed. As before the error rate reduced steadily as N_K was increased, and may have reduced further had higher values of N_K been investigated.

4.3 Experiment 2: Dependence of N_K on Complexity

This experiment investigates how the optimisation of N_K is affected by the complexity of a problem (the inherent linear separability of its classes). Problems B and C contain the letters 'b,c,d,e,g,p,t,v' and 'f,h,i,l,q,r,s,w' respectively. They are of equivalent size, but of markedly different complexity – a fact which is well illustrated by the significant difference in their F ratios (see Section 2.4.2). The results presented in Figure 4.2 were obtained by varying N_K between 32 and 512 to provide conditions varying from underfitting to optimal fitting. The adaptive K-means algorithm was used to allocate centres, and error rates were optimised with the locality index method. The different complexity of the two problems is

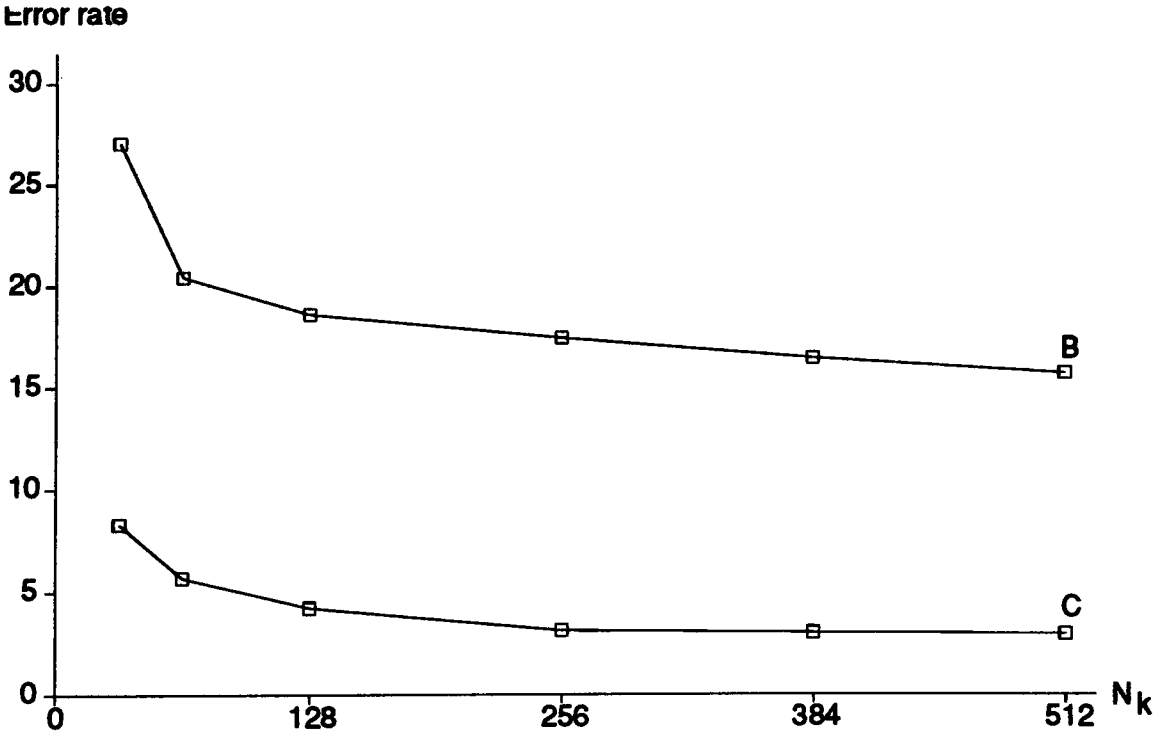


Figure 4.2: *Error rate vs N_K for Problems B and C.*

clearly illustrated by the magnitudes of the error rates obtained, the error rate of Problem C being less than 3 per cent. In this problem the error rate decreased until $N_K = 256$, after

which the addition of further kernel functions produced no further reduction. In Problem B the same effect was observed at $N_K = 512$. It is apparent therefore that larger numbers of kernel functions were required to obtain optimal fitting of the more complex problem. Table 4.4 summarises the minimum values of N_K producing optimal fitting in the two problems, and again compares them with the value of $\frac{M}{2}$. The observed value of N_K corresponds well

Table 4.4: *Optimal values of N_K for Problems B and C.*

Problem	Optimal value of N_K	$\frac{M}{2}$
B	≥ 512	620
C	≥ 256	614

with the value of $\frac{M}{2}$ for the difficult Problem B, but not for Problem C. Since both problems have equivalent numbers of training patterns, the discrepancy can only be explained by the differing complexity of the problems (see Section 3.2.3).

4.4 Experiment 3: Locations of the Centres

This experiment investigates the error rates provided by the different methods of allocating centres. In the terminology of the previous chapter, the following learning regime was used for the adaptive algorithms:

$$\eta = 0.5 \times \exp\left(\frac{-5t}{T}\right)$$

$$T = 100 \times N_K$$

Both feature map algorithms employed a centre-surround spatial decay function, and the following neighbourhood radius decay regime:

$$R(t) = R(0)\left(1 - \frac{5t}{T}\right) \qquad t \leq \frac{T}{5}$$

$$R(t) = 0 \qquad \textit{otherwise}$$

The results presented in Table 4.5 were obtained on Problem A1, using both 64 and 256 centres allocated randomly, with the adaptive K-means algorithm, and with the one-dimensional and two-dimensional feature map algorithms. Kernel function widths were optimised with respect to locality index.

Table 4.5: *Error rates for different methods of allocating centres.*

N_K	Allocation algorithm	Error rate (mean)	Error rate (min / max)	Error rate (range)
64	random	22.28%	21.32% – 22.81%	1.49%
	adaptive K-means	20.49%	20.32% – 20.66%	0.52%
	1D feature map	20.49%	20.14% – 20.87%	0.73%
	2D feature map	20.50%	20.14% – 20.79%	0.65%
256	random	14.12%	13.63% – 14.66%	1.03%
	adaptive K-means	13.24%	12.92% – 13.48%	0.56%
	1D feature map	13.00%	12.70% – 13.30%	0.50%
	2D feature map	13.12%	12.90% – 13.38%	0.48%

The results show that the adaptive algorithms provide lower mean error rates than those observed with random allocation. The reduction is significant enough to warrant the use of an adaptive algorithm when error rate is at a premium. However, in time-critical applications it may be necessary to use random allocation, since the adaptive algorithms are computationally very intensive (see Chapter 5). The difference in mean error rates between the random and adaptive allocation methods decreases for $N_K = 256$, the larger number of centres. This is because input space is better modelled by the random sampling process when the sample size is large. The range of error rates obtained with random allocation is almost twice as large as for the adaptive allocation algorithms, demonstrating the inconsistency resulting from the random process. The consistency offered by the adaptive algorithms is superior for both $N_K = 64$ and $N_K = 256$, indicating that convergence to a

locally minimum distortion yields a similar solution each time. The results also show that there is no significant difference between the error rates of the adaptive K-means algorithm and either of the feature map algorithms. As predicted, Moody and Darken's claim [39] that the adaptive K-means produces superior results to the feature map does not stand up to the empirical evidence presented here. However, the spatial ordering of the feature map does increase the computational complexity of its implementation, favouring the use of the adaptive K-means algorithm for reasons of simplicity if not superior performance. There is also no significant difference between the error rates of the two feature map algorithms, indicating that the degree of constraint on the locations of the centres within the map has no adverse effect on their locations in input space.

4.4.1 Calibrated Maps

Because centres allocated with an adaptive algorithm do not coincide with particular training patterns (as they do with random allocation) they cannot be directly associated with specific classes. However, a calibration process can uncover which class they represent best *on average* over the training set. Calibration is performed by computing the Euclidean distances between each training pattern and centre, determining which centre is closest to the training pattern in each case. A histogram of class membership is drawn up for each centre, and the class most associated with the centre taken as its label[1].

Figure 4.3 shows the results of calibrating 256 centres allocated with the adaptive K-means and two-dimensional feature map algorithms (the centres were arranged into a 16×16 data structure in both cases). The spatial ordering of the two-dimensional feature map is

[1]Calibrated maps can be used to classify test patterns directly, without requiring a second layer of connections. The Euclidean distance between the test pattern and each centre is computed, and the class of the closest matching calibrated centre taken as its class. This is actually the basis of a technique known as *learning vector quantisation*, which has been successfully applied to the recognition of phonemes in continuous speech utterances [27, 35].

a)

k	m	z	k	t	a	g	e	n	p	q	b	p	k	t	m
m	s	l	p	k	t	y	r	y	e	c	l	d	n	k	p
n	l	m	k	s	i	x	u	r	l	f	s	m	p	p	p
s	r	g	x	s	f	p	d	a	w	m	w	x	u	b	r
f	z	z	q	k	q	f	b	b	e	y	o	i	e	d	w
w	f	t	e	t	q	q	y	t	e	m	w	p	w	q	d
i	c	p	d	k	e	l	m	h	k	k	k	n	r	c	o
a	z	x	u	l	w	a	d	b	q	o	b	j	q	k	g
l	x	w	k	h	j	a	r	m	w	y	p	z	p	t	z
f	n	r	w	f	f	v	q	c	z	d	c	q	a	i	q
i	f	k	s	n	a	s	j	h	l	l	z	k	t	x	x
g	z	e	o	e	a	p	b	e	m	o	c	p	q	q	o
w	o	b	k	d	i	i	l	b	h	f	i	h	a	e	a
r	u	c	c	k	f	u	k	k	l	w	u	p	y	i	t
p	f	c	m	c	q	u	x	b	h	o	f	f	s	g	t
m	k	t	q	q	o	y	d	q	x	h	o	i	q	x	h

b)

s	x	x	x	h	h	d	t	e	e	e	d	d	t	t	g
s	x	x	x	x	h	h	e	p	e	p	p	t	t	g	g
s	s	f	f	h	h	h	e	e	e	b	p	p	t	t	t
s	s	s	f	h	h	d	d	b	p	v	w	w	a	p	k
s	s	f	f	f	f	d	e	b	p	v	w	w	a	a	k
f	s	f	f	m	p	d	b	e	p	t	a	k	k	k	j
f	f	f	f	k	k	p	b	b	e	p	a	k	k	k	j
f	i	i	i	i	a	a	b	b	b	t	j	j	k	k	j
y	i	i	i	i	a	a	a	b	p	p	g	k	k	j	j
y	y	y	i	i	w	w	w	n	n	d	j	k	k	z	z
y	y	y	i	i	w	w	n	m	m	n	j	k	v	z	z
y	y	y	i	i	o	o	m	m	q	q	q	q	k	z	z
y	y	i	i	l	o	o	m	m	q	q	q	q	z	z	z
r	r	l	l	l	o	m	m	n	u	u	u	q	c	c	c
r	r	l	l	l	o	o	n	n	u	u	u	q	c	c	c
r	r	r	l	l	o	o	n	n	u	u	u	q	c	c	c

Figure 4.3: *Representation of spoken letters by calibrated map. a) Centres allocated with the adaptive K-means algorithm. b) Centres allocated with the 2D feature map algorithm. Note that spatial ordering has occurred in the latter.*

particularly striking. Confusable subsets of letters (such as the 'e' set and the 'mn' pair) are arranged in loosely overlapping clusters over a wide region of the map, whereas easily discriminated letters (such as 'r', or 'u') arranged in tight clusters on the extremities of the map demonstrating their dissimilarity from other letters. It is apparent that geometric distances on the map do not correspond well to distances in input space as they would, for instance, if the map had been trained on a database of vowel sounds. This is evidence that the inherent dimensionality of an extensive selection of speech sounds is greater than two. There is of course no spatial ordering in the map trained with the adaptive K-means algorithm, but similar numbers of centres are assigned to the classes in both cases.

4.5 Experiment 4: Optimising the Value of L

The error rates appearing in Experiments 1, 2, and 3, were the minimum values obtained after optimising kernel function widths with the locality index method. Thus for each value of N_K, the second layer weights were recomputed for values L in the range $L = -2, -1, 0, \ldots, 8$ (equivalent to multiplying the initial function widths by $\frac{1}{4}, \frac{1}{2}, \ldots, 256$ respectively). The error rate was evaluated for each value of L to uncover its optimum value, L_{opt}. Figure 4.4 illustrates the results of the optimisation process for Problem A1, over a range of values of N_K. As usual centres were allocated with the adaptive K-means algorithm. It is clear that error rate depends strongly on the value of L, at all values of N_K. This demonstrates the importance of optimising L. Table 4.6 compares the error rates achieved using widths determined by the global nearest neighbour heuristic method ($L = 0$) with those obtained after optimising the value of L with the locality index method. The locality index method appears to be particularly effective when the training data is underfitted, since the difference in error rate between the two methods is largest at the lower values of N_K. Under

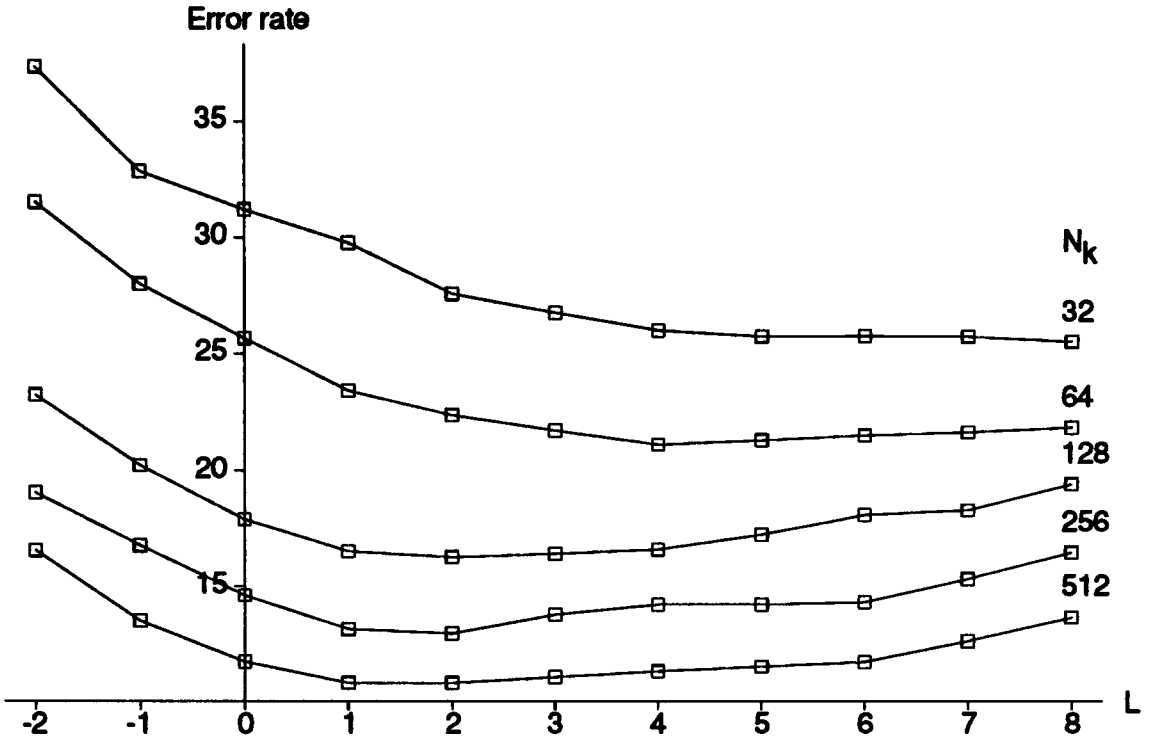


Figure 4.4: *Error rate vs L for Problem A1.*

these conditions there are too few kernel functions to adequately sample the input space, leaving large regions of it unrepresented. Underfitting can be partly offset by extending the kernel function widths into these regions of input space, reducing the likelihood that test patterns will occur in unrepresented regions. However, the benefit gained by this approach is associated with a loss in classification sensitivity (see Section 3.4.1), and consequently the problem of underfitting is not fully overcome. Table 4.7 summarises the values of L_{opt} for Problems A1–A3, B and C. The results show that L_{opt} drops steadily as the training sets become fitted more optimally. It is interesting to note that in cases where optimal fitting was achieved, $L_{opt} = 0$. The heuristic method is therefore applicable in conditions of optimal fitting, but error rate can be substantially lowered by the locality index method if this condition does not prevail.

Table 4.6: *Comparison of error rates between heuristic and locality index methods.*

N_K	Error rate (heuristic method)	Error rate (locality index method)	Difference
32	31.20%	25.80%	5.40%
64	25.67%	21.15%	4.52%
128	17.90%	16.27%	1.63%
256	14.61%	13.24%	1.37%
384	12.94%	11.62%	1.32%
512	11.72%	10.80%	0.92%
768	11.10%	10.59%	0.51%
1024	10.81%	10.38%	0.43%

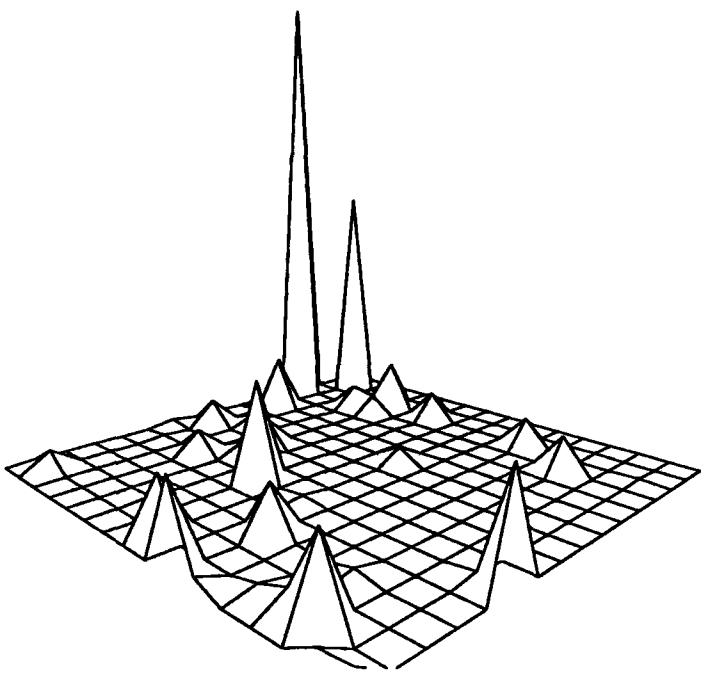
Table 4.7: *Summary of L_{opt} for Problems A1–A3, B and C.*

N_K	L_{opt} (Problem A1)	L_{opt} (Problem A2)	L_{opt} (Problem A3)	L_{opt} (Problem B)	L_{opt} (Problem C)
32	8°	2°	2°	5°	2°
64	4°	2°	1°	3°	2°
128	2°	2°	1°	3°	1°
256	2°	1°	1°	1°	0
384	2°	1°	0	1°	0
512	2°	0	1•	1°	0
768	1°	0	–	0	0
1024	1°	0•	–	–	–

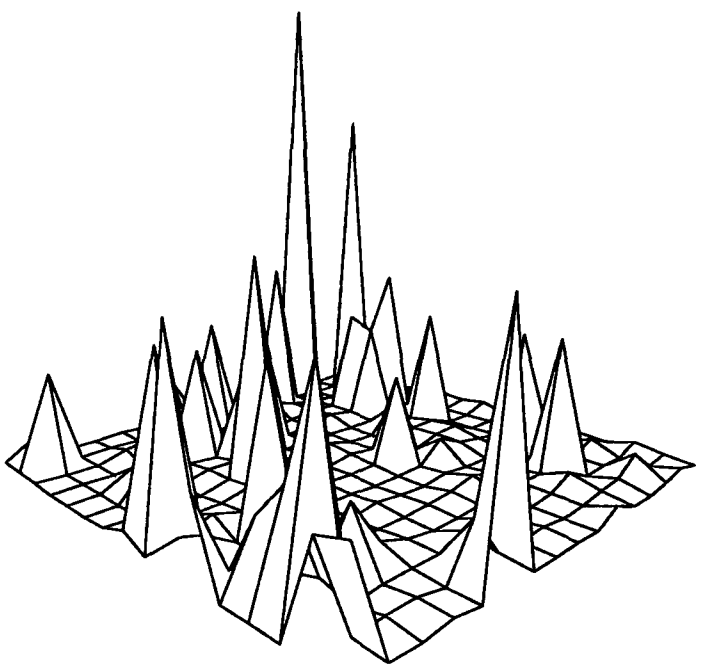
(° underfitting, • overfitting)

4.5.1 Effect of Locality Index L

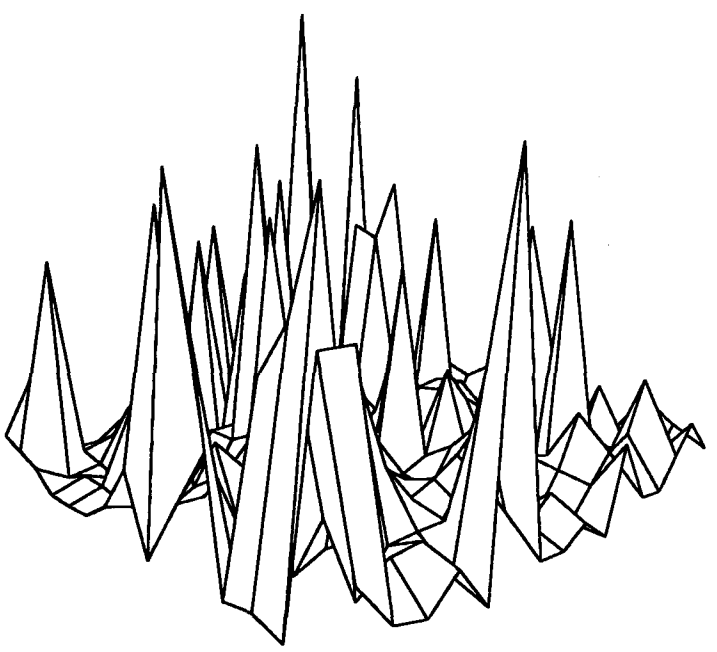
The way in which the locality of representation varies with L can be observed clearly by inspecting the responses of the kernel functions to a typical input pattern, as L is varied through different values (an advantage of the RBF over the MLP is the simplicity with which hidden layer representations can be visualised and interpreted). Figures 4.5a–c show the responses of 256 kernel functions to an utterance of the letter 's', for $L = 0, 1, 2$. Centres



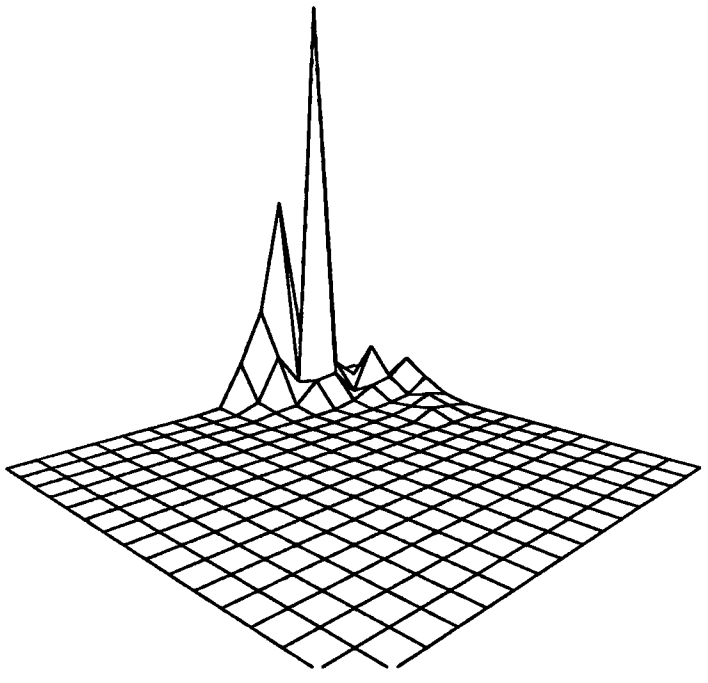
a) Adaptive K-means algorithm, $L = 0$.



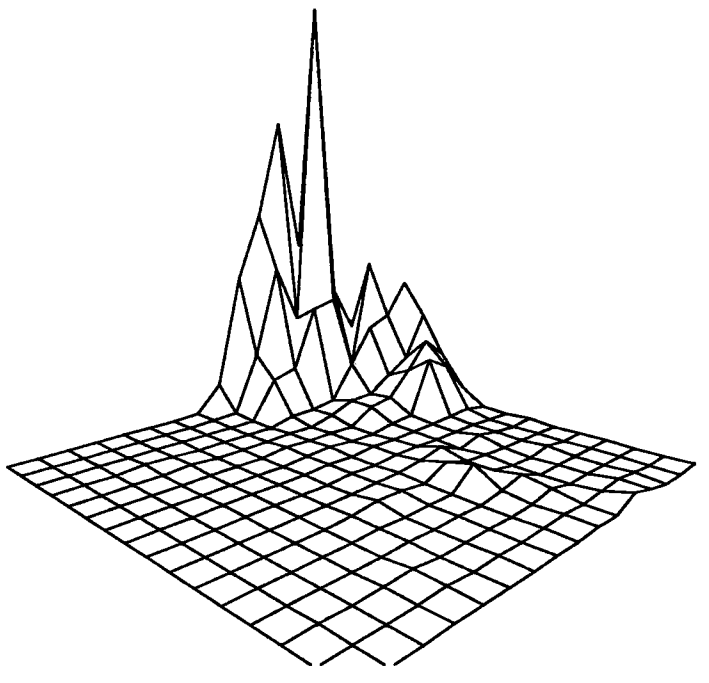
b) Adaptive K-means algorithm, $L = 1$.



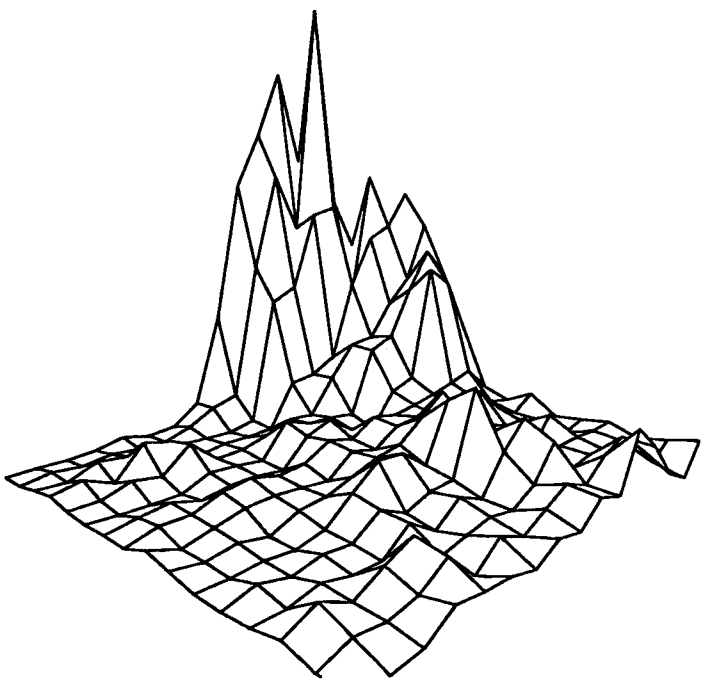
c) Adaptive K-means algorithm, $L = 2$.



d) 2D feature map, $L = 0$.



e) 2D feature map, $L = 1$.



f) 2D feature map, $L = 2$.

Figure 4.5: ***Kernel function responses to an utterance of the letter 's'.***

were allocated with the adaptive K-means algorithm and therefore show no evidence of spatial ordering. Figures 4.5d–f were produced under similar conditions, except that centres were allocated with the two-dimensional feature map and are therefore spatially ordered. The positions of the peak kernel function responses, and the positions of the centres labelled 's' in the calibrated maps of Figure 4.3 correspond closely since both were derived during the same experiment.

The value of L dictates how many kernel functions produce zero, or near-zero responses to input patterns. Such kernel functions play little part in classifying the input pattern since their contribution to the network outputs is negligible. The extent to which these kernel functions are redundant can be quantified by measuring the increase in error rate when the lowest η per cent are discarded (responses set to zero). The results presented in Figure 4.6 were recorded for $L = 0, 1, 2$, when centres were allocated with the adaptive K-means algorithm. Table 4.8 contains the exact numerical values for $\eta = 25, 50$. When $L = 0$

Table 4.8: *Increase in error rate vs discard factor, η, for $L = 0, 1, 2$.*

Discard factor, η	Locality index, L	Increase in error rate
25%	0	0.5%
	1	4.0%
	2	98.7%
50%	0	7.3%
	1	80.5%
	2	100.0%

(Figure 4.5a), input patterns are represented by very localised kernel function responses. As only a small number of kernel functions respond strongly to any input pattern, error rate is barely affected by discarding large numbers of the lowest-response kernel functions.

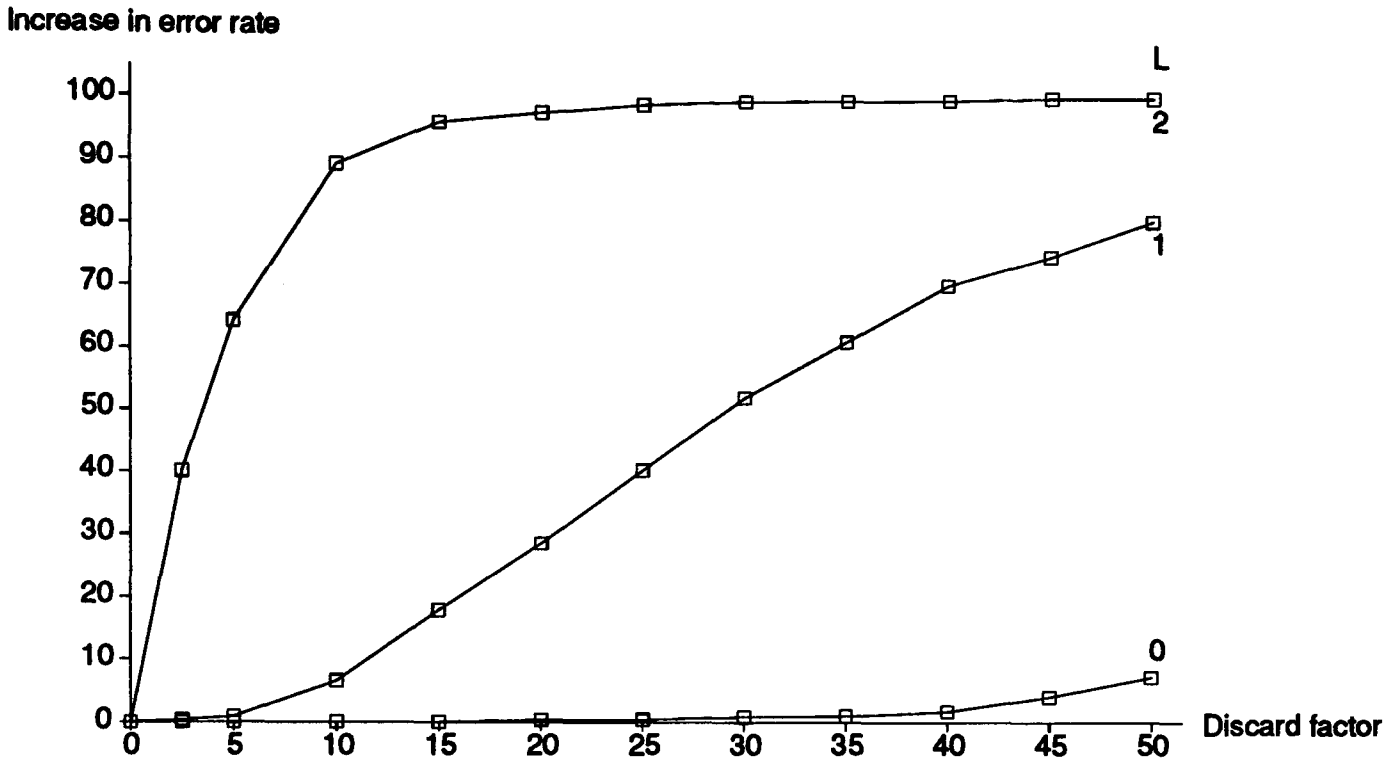


Figure 4.6: *Increase in error rate vs discard factor, η, for $L = 0, 1, 2$.*

When $L = 1$ (Figure 4.5b) input patterns are represented by a larger fraction of the kernel functions. Less kernel functions can therefore be discarded without incurring a rise in error rate. When $L = 2$ (Figure 4.5c) input patterns are represented by the responses of almost all kernel functions, and error rate rises sharply as soon as kernel functions are discarded.

4.6 Summary of Results

The results of Experiment 1 suggest that the theoretical estimate of the optimal value of kernel functions, $\frac{M}{2}$, corresponds well with the observed optimal values on Problems A2 and A3. The results of Experiment 2 illustrates that fewer than $\frac{M}{2}$ kernel functions may be employed for Problems of lower complexity, in which the effect of underfitting is less pronounced. The results of Experiment 3 show that random allocation of centres offers inferior generalisation to that provided by the adaptive allocation algorithms, but that no

significant difference in generalisation exists between the latter. The adaptive K-means algorithm is favoured since it is the simplest algorithm computationally. However, as we shall see in Chapter 7, there are instances in which it is desirable to make use of the feature map for visualisation and interpretation of data sources. Finally, the results of Experiment 4 show that generalisation is influenced by the locality of representation of input patterns in the space defined by the kernel function outputs. This effect is especially evident when fewer than optimal kernel functions are employed, as might be the case when training time or hardware specification imposes some constraint. The locality index is an efficient method for optimising kernel function widths. More flexibility is provided than with heuristic methods, without incurring the large increases in training time of a fully adaptive method.

Chapter 5

Assessment of Classifier Performance

5.1 Introduction

The ability of a classifier to generalise, as measured by its error rate on test patterns, is the major factor by which its performance is assessed. This is particularly true during the research phase when use of time and memory is traded-off against reductions in error rate in an effort to build more powerful classifiers. However, a practical problem specification might include firm limits on time and memory resources, and since every classifier has a different computational requirement performance assessment must be extended to include the following factors:

1. Training memory: The memory requirement of the training algorithm(s).

2. Working memory: The memory requirement for classification of test patterns.

3. Training time: The time required by the training algorithm(s).

4. Classification time: The time required to classify a test pattern.

As an example, it might be useful to use a near instantaneous training algorithm in a single speaker, low vocabulary speech recognition device, suitable for variable tasks under adverse environmental conditions. But if in addition such a device was to be made portable, memory

efficiency might attain a higher priority than either training time or error rate. In contrast, error rate would be the the most important factor in a voice-operated telephone enquiry system, since it would be required to recognise words from a large number of ethnically diverse people, and could be based in a highly equipped laboratory.

This chapter assesses the performance of the classifiers introduced in Chapter 1 when they are applied to Problem A1, the most extensive of the problems studied, and which, as noted in Chapter 2, was the problem selected by Woodland [61] for his study of the MLP[1]. As already mentioned in Chapter 4 the computing system used to assess the classifiers was a 25 MHz SUN workstation containing a floating point co-processor, 32 Mbytes of real memory, and 60 Mbytes of virtual memory. In this system single precision floating point numbers were represented by four bytes (32-bits), and double precision numbers by eight bytes (64-bits). Obviously, the observed values of training and classification times are heavily dependent upon the specification of the computing system, and cannot be extrapolated to other systems. However, a comparison of the *relative* training and classification times is still valid. To avoid unnecessarily high error rates it is imperative that the classifiers are optimised using an efficient strategy. For this reason the optimisation strategies of the classifiers are briefly reviewed below, following which the results of the assessment are presented.

The OLT

The OLT requires no optimisation if singular value decomposition is employed (a parameterless linear transformation of the training patterns). This method was employed here in preference to the iterative Widrow-Hoff rule, which is computationally more intensive, and

[1]The assessment of the MLP in this chapter is based on Woodland's research.

less elegant, since it requires optimisation of learning rate and adaptation time.

The KNN

A distance metric must first be selected based on prior knowledge of the data source (different metrics are usually evaluated to discover which is most effective). A Euclidean metric was employed here as it has been previously shown to be suitable for classification of mel frequency cepstral coefficients [22]. The parameter K was then optimised with respect to error rate.

The MLP

The learning parameters, choice of initial weights, and weight magnitude limits, were extensively optimised for the back-propagation algorithm of the MLP. The number of hidden units employed, N_H, was then optimised with respect to error rate[2]. Note that gross modifications to the value of N_K require that many network parameters be re-optimised.

The RBF

A Euclidean distance metric was employed in the first layer of the RBF (as with the KNN). Centres were allocated with the adaptive K-means algorithm, the learning parameters of which are determined with a set of heuristics (see Section 4.4). Singular value decomposition was used to compute the second layer weights (as with the OLT), and the values of N_K, and L were optimised with respect to error rate.

[2]There have been some attempts at selecting the value of N_H according to more formal criteria [30, 19] although such techniques were not adopted by Woodland in this case.

5.2 Error Rate

The error rates of the optimised classifiers are summarised in Table 5.1. The lowest error rate

Table 5.1: *Error rates of the optimised classifiers.*

Classifier	Error rate	Conditions
OLT	36.0%	–
KNN	16.8%	Euclidean metric $K = 10$
MLP†	11.7%	$N_H = 75$ weight magnitude ≤ 1.5
RBF	10.4%	Euclidean metric adaptive K-means algorithm $N_K = 1024$ $L = 1$

(† from Woodland [61])

was obtained with the RBF. That it was still in excess of 10 per cent indicates the complexity of Problem A1. The much higher error rate obtained with the OLT can be explained by the fact that the pattern classes making up Problem A1 are not linearly separable; single hyperplane decision boundaries in input space cannot therefore be expected to disambiguate the classes. The error rate obtained with the KNN is substantially lower than that obtained with the OLT, demonstrating that decision boundaries based on within-class distances are more effective than single hyperplanes for this problem (this is generally true for complex problems). The error rates obtained with the neural network classifiers are substantially lower again. Table 5.2 shows the reduction in error rate provided by the RBF relative to the other classifiers. Because the OLT is identical in architecture to the second layer of the RBF (both perform linear discriminant analysis), the large reduction in error rate provided by the RBF with respect to the OLT can be attributed to the non-linear *projection* of input

Table 5.2: *Reduction in error rate provided by the RBF relative to the other classifiers.*

Classifier	Reduction in error rate
OLT	25.6%
KNN	6.4%
MLP	1.3%

patterns occurring in its first layer. In contrast, the architecture of the KNN resembles that of the first layer of the RBF (one centre allocated for each training pattern). The reduction in error rate provided by the RBF with respect to the KNN can therefore be attributed to its second *discriminative* layer. Clearly, the poorer generalisation of the OLT and KNN underline the importance of a multilayer architecture for efficient classification. It is interesting to note that the error rate obtained with the MLP is almost comparable to that of the RBF, although the operation of the classifiers is fundamentally different (the difference in error rate that does exist is still worthy of note because reductions in error rate become increasingly difficult to obtain with diminishing error rate). The hidden units of the MLP perform what is essentially a feature extraction role, each unit becoming sensitised to a feature of the training patterns that is useful in discriminating the classes. In contrast, the kernel functions of the RBF constitute an alternative space in which training patterns may be represented – a space in which the classes of training patterns are more linearly separable.

It was noted in Section 1.2.2 that the error rate of the optimal Bayes classifier, γ_B, can be predicted to lie in the range $\frac{\gamma_K}{2} \leq \gamma_B \leq \gamma_K$, where γ_K is the error rate of the KNN. Using this relation in the context of Problem A1, it can be predicted that $8.4\% \leq \gamma_B \leq 16.8\%$. The error rate of the RBF is therefore very close to the theoretical minimum error rate achievable by any classifier. Figure 5.1 shows the 'confusion matrix' associated with the

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	84.4	0.7	0.0	1.3	1.3	0.0	0.0	0.0	5.2	0.0	3.2	0.0	0.0	2.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0	0.0
b	2.6	77.8	0.0	5.2	4.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.5	0.0	0.0	0.0	0.0
c	0.0	0.0	94.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.2	0.0	2.0	0.0	0.0	0.0	0.0
d	0.0	12.5	0.0	67.1	5.3	0.0	1.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.6	0.0	4.6	0.7	0.0	0.0	0.0
e	0.0	12.3	0.0	3.2	79.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.6	0.0	0.0	0.0	0.0
f	0.0	0.0	0.0	0.0	0.0	98.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7
g	0.0	0.0	0.0	2.0	1.3	0.0	88.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
h	0.0	0.0	0.0	0.0	0.0	0.0	0.0	99.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
i	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	94.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
j	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	95.5	3.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
k	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	89.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
l	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	98.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
m	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	64.1	29.5	5.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
n	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	11.0	84.5	3.9	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
o	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.3	0.7	2.6	88.9	0.0	0.0	0.7	0.0	0.0	0.0	0.0	2.6	0.0	0.7	0.0
p	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	83.1	0.0	0.0	0.0	2.6	0.0	5.8	0.0	0.0	0.0	0.0
q	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.7	0.0	0.0	95.2	0.0	0.0	0.0	2.7	0.0	0.7	0.0	0.0	0.0
r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.5	0.0	0.0	94.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
s	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0	0.0	0.0	96.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
t	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.8	0.6	0.0	0.0	83.3	0.0	2.6	0.0	0.0	0.0	0.6
u	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0	0.0	98.7	0.0	0.0	0.0	0.0	0.0
v	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.2	0.7	77.8	0.0	0.0	0.0	0.0
w	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	97.4	0.0	0.0	0.7
x	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	94.2	0.0	0.0
y	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	96.1	0.0
z	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	96.8

Figure 5.1: *Confusion matrix associated for optimised RBF. The more diagonal the matrix, the less confusion exists between the classes.*

RBF (note that in this instance the numbers refer to percentage *recognition* rates, not error rates). The confusion matrix indicates between which classes confusion, or misclassification, took place. Perfect classification (i.e. no confusion) would result in a diagonal confusion matrix. In practice however, there is significant confusion between the letters of the 'e' set and between the 'mn' pair.

5.3 Training Memory

Training memory is the memory requirement of the training algorithm. It depends on the dimensions of the classifier, the computational technique used to derive the classifier parameters, and the efficiency with which the training algorithm is implemented in software; care must be taken to free, or overwrite memory when it is no longer required. Table 5.3 summarises the minimum training memory requirements of the classifiers when trained using the specified algorithms. Note that no training is required for the KNN[3]. The back-propagation and adaptive K-means algorithms operate iteratively; training consists of the repeated application of a simple formula to training patterns. The memory required by

Table 5.3: *Training memory requirements of the classifiers.*

Classifier	Training algorithm	Memory (bytes)	Conditions
OLT	SVD	10.3 M	–
KNN	–	–	–
MLP	back-propagation	2.3 M	–
RBF	SVD	20 M	$N_K = 256$
		39 M	$N_K = 512$
		85 M	$N_K = 1024$
	adaptive K-means	1.9 M	–

[3]The KNN belongs to a group of classifiers known as 'template matching' procedures. In general such classifiers do not require a training phase since training patterns are processed directly during classification.

such algorithms is used purely for storing the training and target patterns[4]. In contrast, singular value decomposition has a very high memory requirement. In the case of the NAG library [41] version of the algorithm used in this thesis, the high memory requirement is due to the following:

1. The factorisation, $A = QDP^T$ (see explanation in Section 1.2.1), requires the storage of four large matrices (e.g. the $N_K \times M$ matrix A contains over two million elements when $N_K = 512$, and $M = 3999$).

2. A matrix the size of P is additionally required for work space.

3. Matrix elements are stored in double precision (eight bytes on the present computing system).

The training memory requirement of the RBF can be reduced to that of the MLP if singular value decomposition is replaced by the iterative Widrow-Hoff rule (see Section 1.2.1). This makes the Widrow-Hoff rule an attractive option in memory-critical applications. However, it is associated with long training times and is often terminated before the output error has converged – in which case a minimum squared error solution is not obtained. Although the memory requirement of singular value decomposition is high, it was not beyond the capabilities of the workstation used to obtain the results in this thesis. The present computing system has a total of 92 Mbytes of usable memory (real + virtual), which was sufficient to implement an RBF with a maximum of 1024 centres.

[4]It is unnecessary to store target patterns in the adaptive K-means algorithm since it is unsupervised.

5.4 Working Memory

Working memory is the memory requirement of the classifier during classification of test patterns. It is important from a practical, not theoretical point of view, since any physical implementation of the classifier will be restricted by the amount of memory it may consume. Working memory can be calculated by summing the total number of variables required by the classifier during its operation. The main variables for each classifier are given in Table 5.4. Note that the requirement of the KNN is directly related to the number of training

Table 5.4: *Number of variables required for classification.*

Classifier	Variables	Description
OLT	$(N_I+1)N_C$	weights
KNN	$N_I M$	training patterns
	M	training labels
MLP	$N_I N_H$	first layer weights
	$N_H N_C$	second layer weights
RBF	$N_I N_K$	centres
	$(N_K+1)N_C$	second layer weights

patterns, M. Since the values of N_H and N_K are selected to ensure that the training data is optimally fitted, the requirements of the MLP and RBF are indirectly related to M. With each variable represented in single precision, the actual amounts of working memory required by the classifiers are summarised in Table 5.5. It is clear that working memory is highly dependent on the choice of classifier, the requirement of the KNN being over two orders of magnitude greater than that of the OLT. However, the KNN does achieve a much lower error rate than that of the OLT, although its efficiency is lower than that of either of the neural network classifiers; the MLP and RBF provide significantly lower error rates than the KNN, with significantly less working memory. Their exact requirements depend

Table 5.5: *Working memory requirements of the classifiers.*

Classifier	Working memory (bytes)	Conditions
OLT	13 K	–
KNN	1936 K	–
MLP	15 K	$N_H = 25$
	29 K	$N_H = 50$
	44 K	$N_H = 75$
RBF	150 K	$N_K = 256$
	301 K	$N_K = 512$
	602 K	$N_K = 1024$

on N_H, and N_K respectively, with the MLP requiring roughly an order of magnitude less working memory than the RBF.

5.5 Training Time

Training time is the time requirement of a training algorithm. It is affected by the same factors which determine training memory. A full analysis of training time is only given for the OLT and RBF, since no training is required by the KNN, and the training time of the MLP is very sensitive to the learning parameters employed in the back-propagation algorithm. Both the OLT and RBF employ singular value decomposition, the most time-consuming part of which is the factorisation, $A = QDP^T$ (the subsequent computation of a solution taking negligible time in comparison). Assuming that the factorisation is performed in real memory, the time taken is proportional to the number of columns in A, and the square of the number of rows in A [41]. For the OLT, the time taken is therefore proportional to M, and N_I^2, and for the RBF it is proportional to M, and N_K^2. Unfortunately, the values of M and N_K are such that in the case of the RBF, the majority of the factorisation is performed in virtual memory. This substantially increases the time taken to perform the algorithm

as memory must be frequently exchanged between the CPU and hard disk. The time taken by the adaptive K-means algorithm to allocate centres in the RBF, is proportional to N_K^2, marginally less than the feature map allocation algorithms. Table 5.6 summarises the training times of the algorithms employed by the classifiers. The trade-off between training

Table 5.6: *Training times of the classifiers.*

Classifier	Training algorithm	Time (minutes)	Conditions
OLT	SVD	6	–
KNN	–	–	–
MLP	back-propagation	long	$\forall N_H$
RBF	SVD	26	$N_K = 256$
		110	$N_K = 512$
		540	$N_K = 1024$
	adaptive K-means	33	$N_K = 256$
		132	$N_K = 512$
		528	$N_K = 1024$

time and generalisation is highlighted clearly: the KNN and OLT require the least training time, but demonstrate the poorest generalisation, whilst the longer training times required by the RBF and MLP result in superior generalisation. If a slight reduction in generalisation can be tolerated, the training time of the RBF can be reduced considerably by allocating centres randomly. Unfortunately, no faster method is available to replace singular value decomposition (the iterative Widrow-Hoff rule requires significantly *more* training time for an equivalent solution). With the cost of memory declining rapidly, and computing power continuing to increase, training times will undoubtedly decrease in the near future. At present however, it would seem wise to limit the value of N_K to 512, as the reduction in error rate occurring between $N_K = 512$ and $N_K = 1024$ is only 0.42 per cent.

5.6 Classification Time

Classification time is the time taken to classify a test pattern. This may be broken down into a list of basic computing operations: additions (a), multiplications (m), exponentials (x) etc (subtraction and comparison require the same number of operations as addition, and division requires the same number as multiplication). The numbers[5] of such operations required by the classifiers are given in Table 5.7. The characteristics of the computing sys-

Table 5.7: *Computing operations required for classification.*

Classifier	Computing	operations	Description
OLT	$N_C(N_I+1)$	[m + a]	weighted sum
	N_C	[a]	select maximum class
KNN	MN_I	[m + 3a]	Euclidean distance
	MK	[a]	select maximum class
MLP	N_HN_I	[m + a]	weighted sum
	N_H	[x + m + a]	sigmoid
	N_HN_C	[m + a]	weighted sum
	N_C	[x + m + a]	sigmoid
	N_C	[a]	select maximum class
RBF	N_KN_I	[m + 3a]	Euclidean distance
	N_K	[x + m]	Gaussian
	$N_C(N_K+1)$	[m + a]	weighted sum
	N_C	[a]	select maximum class

tem on which the classifier is implemented also have a considerable effect on classification time. In the case of a digital computer, important factors are the processor clock rate, the amount of real and cache memory, and the compiler efficiency. However, the most important factor is whether or not a floating point co-processor is included, since classification algo-

[5] In a practical software implementation of the classifiers additional computation is required to support the main algorithm (for instance calculation of array indices, incrementing program counters, etc).

rithms in general, and neural network algorithms in particular, are multiplication intensive. Assuming a floating point co-processor is present the issue then becomes one of architecture. For instance, a pipelined architecture may allow single cycle multiplications under certain conditions, and look up tables provide a fast method of computing exponential and trigonometrical functions. If faster computation is desired, then special purpose hardware based on *digital signal processor* (DSP) chips may be designed. The architectures of DSP chips are optimised for the efficient computation of $y = a \times b + c$, since this computation is of fundamental importance in most aspects of digital signal processing. In the context of digital simulation of neural networks this computation assumes special importance, since only a small minority of other kinds of computation (such as branches) are required. In most DSP chips it may be executed in a *single* instruction cycle due to a parallel array multiplier/accumulator being integrated into the data path.

Table 5.8 summarises the classification times of the classifiers when implemented on the present computing system. The classifier with the longest classification time is the KNN, as

Table 5.8: *Classification times of the classifiers.*

Classifier	Classification time (seconds)	Conditions
OLT	1/104	–
KNN	23/20	–
MLP†	1/82	$N_H = 25$
	1/41	$N_H = 50$
	1/27	$N_H = 75$
RBF	1/8	$N_K = 256$
	1/4	$N_K = 512$
	1/2	$N_K = 1024$

(† calculated like N_H centre RBF)

M Euclidean distances must be computed for a test pattern to be classified (for techniques

to reduce this computational overhead see Section 1.2.2). Next is the RBF, which requires roughly an order of magnitude more computation than the MLP to achieve a similar level of generalisation. The OLT has the lowest classification time, but as we have already seen, its high speed of operation is not associated with good generalisation.

The average length of the utterances in the CONNEX alphabet database is 0.48 seconds. With the present classification techniques words are processed *after* they have been uttered. This produces a delay equivalent to the classification time before the results of classification are known. The only classifier having a classification time in excess of the average word length is the KNN. The fact that such low classification times are achievable with the neural network classifiers raises doubts as to the utility of special purpose hardware implementations incorporating digital signal processing chips, or fully parallel architectures (see [61] for example). Special purpose implementations may of course be desirable in situations where there is a requirement for portability or low power consumption, or where the implementation is designed as a peripheral to a host computing system. A disadvantage of most special purpose implementations is that the overwhelming majority of computation required by the neural network classifiers occurs during the training phase, yet it is not possible to train the classifier *in situ* due to insufficient availability of memory. Training time is therefore not reduced in such cases, since training is performed in the normal way on a host computer which downloads the trained weights into the hardware implementation.

5.7 Summary of Assessment

This assessment has revealed that the neural network classifiers were able to generalise significantly better than the traditional classifiers on a difficult speaker-independent spoken letter recognition problem. Although the generalisation abilities of the MLP and RBF were

almost equivalent, slightly better generalisation was obtained with the RBF demonstrating the effectiveness of the optimisation strategies developed in Chapter 3. In comparison with the MLP, the RBF required approximately an order of magnitude more working memory and classification (computation) time. The training memory requirement of the RBF did not compare favourably with the other classifiers under study, but is within the capabilities of a modern workstation. All the classifiers apart from the KNN performed classification in less time than the average length of an utterance, demonstrating their suitability to time-critical applications. The training times of the neural network classifiers were significantly greater than those of the traditional classifiers, clearly demonstrating the trade-off between training time and generalisation.

If the classifiers are implemented in dedicated hardware, floating point DSP chips currently provide the best such solution, although present architectures do not contain sufficient memory to enable the training algorithms to be performed *in situ*. Because the training times of neural network classifiers are generally large, there is some doubt as to the utility of special purpose implementations unless portability or low power is specifically required.

The classifiers considered so far in this thesis process fixed length input patterns, obtained by applying a linear time normalisation procedure to variable length templates representing utterances of the letters of the alphabet. The dimensionality of the patterns is independent of the duration of the word; the long word 'w' is represented by the same dimension pattern as the short word 'u'. Thus the time normalisation procedure eliminates information concerning word duration. The classifiers considered in the following chapter overcome this limitation.

Chapter 6

Dynamic Classification

6.1 Introduction

A major source of variation in the speech signal, and one which must be modelled for the effective recognition of speech sounds, is variation in word duration and in the relative rates of production of phonemes within a word. This is known as *temporal variation*. The static classifiers so far examined in this thesis cannot model word duration because this information is discarded during linear time normalisation. However, variation in the relative rates of production of phonemes is modelled *implicitly* by the static classifiers, since this variation is maintained in the fixed length templates in a compressed form.

Dynamic classifiers model both aspects of temporal variation *explicitly*. The two most popular dynamic classification techniques for speech recognition are dynamic time warping (DTW), and the hidden Markov model (HMM). DTW is an efficient method for *non-linear* time normalisation; the time axes of the training templates are distorted (warped) in order to calculate which most closely resembles the test template (see Section 6.2). The HMM is a stochastic approach in which each class of word is modelled by a finite number of 'states', associated with probabilities of transition into other states and of emitting a finite number of outputs corresponding to possible feature vectors. Test templates are classified using a dynamic programming algorithm which calculates the probability of each of the word models producing the sequence of frames observed in the test template. The maximum likelihood

word model is then selected as the class of the test template (for a detailed discussion of the HMM see [12]).

Most commercial speech recognition systems employ one of these dynamic classification techniques. The HMM is the most general of the two techniques[1], and probably the most effective technique yet developed for speech recognition. However, DTW is much simpler to implement and has the advantage of requiring less training data to achieve good results. It has been reported to provide equivalent error rates to the HMM in several experiments in isolated word recognition [45, 44], and, as noted in Chapter 2, has been evaluated by Woodland [61] on the CONNEX alphabet database. In this chapter DTW is compared with the RBF on Problem A1, the full speaker-independent problem, and on Problem D, a smaller multiple-speaker problem (for a description of these problems see Section 2.4.2). A new algorithm is then developed which combines non-linear time normalisation with an adaptive algorithm of the type used to allocate centres in the RBF. The result is a dynamic classifier which achieves near comparable performance to DTW with significantly reduced computational overheads.

6.2 Dynamic Time Warping

First introduced in the early 1970's [57, 50, 23], the dynamic time warping algorithm marked a breakthrough in overcoming the problem of temporal variation in speech signals. The operation of the algorithm is best explained by considering each word template as a sequence of points in feature space, $\mathbf{f}_1$, $\mathbf{f}_2$, $\mathbf{f}_3$, ...(this representation is explained in Section 2.4.1). The objective of DTW is to find which of the variable length sequences belonging to the training templates is most similar (in Euclidean distance) to that of the test template. This

[1] DTW has been shown to be a special case of the HMM, in which state transition probabilities are equal, and transitions are limited to one or less states [12].

is achieved by non-linearly distorting the time axis of each of the training templates so that it is optimally aligned with the test template. Each training template is then ranked with a *distance score*, depending on the quality of its alignment to the test template, the class of the training template with lowest distance score being taken as the class of the test template. The algorithm proceeds by computing a *cumulative* distance matrix for each training template by means of a recursive formula. For the mth training template this formula is given by:

$$\begin{aligned} C[i,j] \;=\; min[&C(i-1,j) + W_h\ d_m(i,j), \\ &C(i,j-1) + W_v\ d_m(i,j), \\ &C(i-1,j-1) + W_d\ d_m(i,j)] \end{aligned} \tag{6.1}$$

where $d_m(i,j)$ is the Euclidean distance between the ith feature vector of the mth training template and the jth feature vector of the test template:

$$d_m(i,j) = ||\mathbf{f}_i^m - \mathbf{f}_j^{test}|| \tag{6.2}$$

and $C[i,j]$ is the cumulative distance between the templates at this point. If I and J are the lengths (number of frames) of the training and test templates respectively, the total cumulative distance between the two templates, or distance score, is $C[I,J]$. For a particular test template, the magnitude of the distance score is affected by the lengths of the training templates, longer training templates producing higher distance scores. To avoid biasing classification in favour of shorter training templates the distance score should be normalised by dividing by the length of the training template. The parameters W_h, W_v, and W_d, are the weightings applied to the horizontal, vertical, and diagonal contributions to the cumulative distance respectively. The distance score can be made independent of

temporal variation if the summed weighting of the vertical and horizontal contribution is equal to that of a diagonal contribution[2]:

$$W_h = W_v = 1$$

$$W_d = 2$$

Some penalty for temporal variation has been found to reduce error rate however, and thus the weighting of the diagonal contribution is normally reduced. The following weighting scheme is used in this thesis:

$$W_h = W_v = W_d = 1$$

The recursion formula is first used to derive the consecutive series of elements $C[0,0], C[0,1], \ldots, C[0,J]$, and $C[0,0], C[1,0], \ldots, C[I,0]$, based on the element $C[0,0]$, which is the Euclidean distance between the two initial feature vectors of the templates. Following this the remaining elements of the matrix may be computed. Upon completion of the matrix it is possible to examine the path of optimal time alignment between the templates; a path is traced backwards through the matrix from $C[I,J]$ to $C[0,0]$, following the route of maximum decrease in cumulative distance at all points. Figure 6.1 illustrates the path of optimal time alignment for a test template belonging to the same class as the training template, and one belonging to a different class for comparison.

[2]It is usual to make W_h and W_v equal, since stretching and compressing are equally valid forms of time axis distortion.

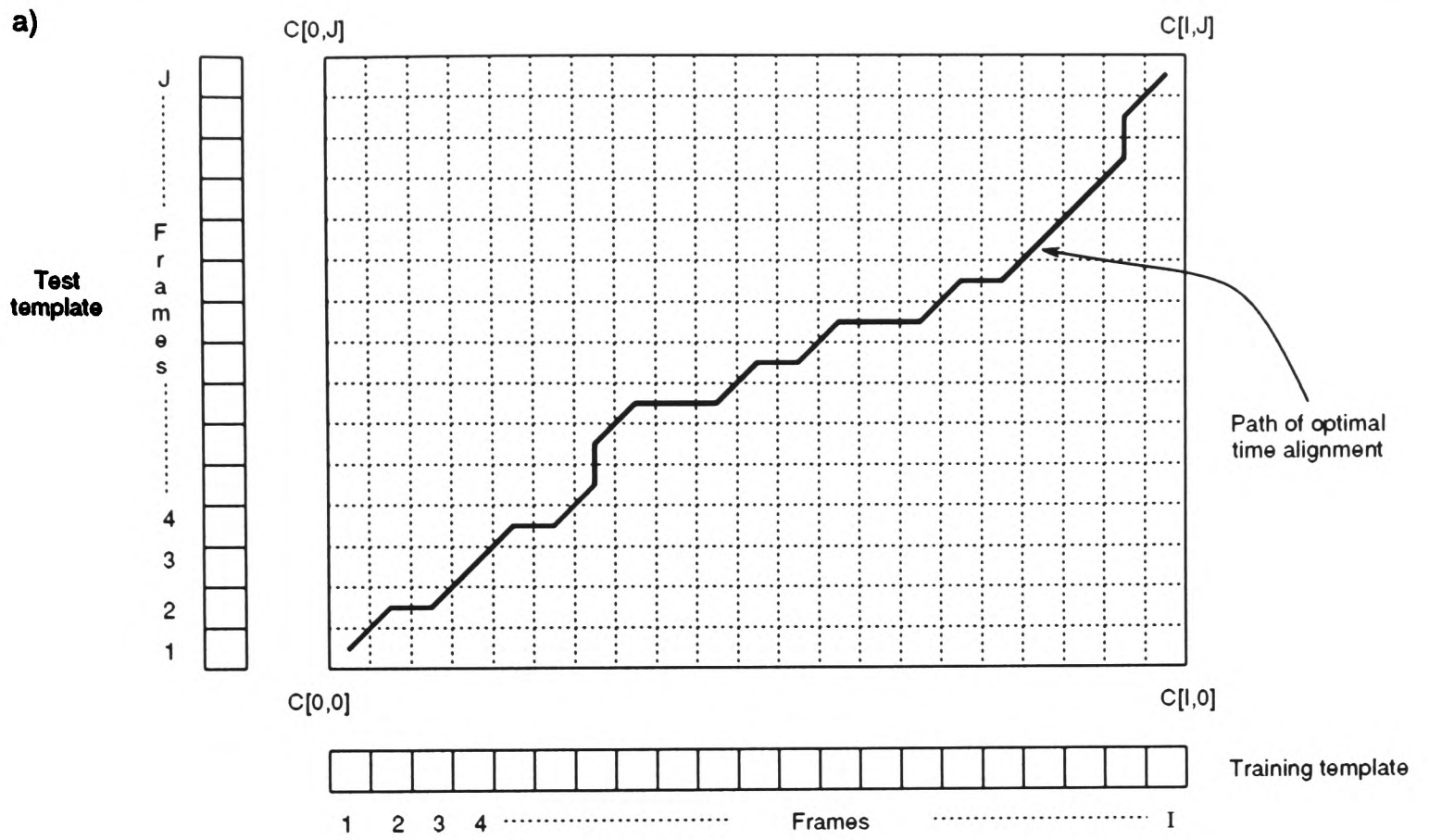


Figure 6.1: *Paths of optimal time alignment in cumulative distance matrix. a) The test template belongs to the same class as the training template. The path is nearly diagonal indicating minor temporal variation between the templates. b) The test template belongs to a different class as the training template. The path is very distorted indicating severe temporal variation between the templates.*

6.2.1 Techniques to Reduce Classification Time

Classification requires the computation of a separate $I \times J$ cumulative distance matrix for each training template. Classification time therefore depends linearly on the number of training templates and upon the square of the template lengths. DTW is a very intensive technique computationally, and several methods have been proposed to reduce computation without reducing error rate.

One class of methods makes the valid assumption that within-class temporal variation is likely to be less severe than between-class temporal variation. The path of optimal time alignment is therefore unlikely to diverge considerably from the straight diagonal path between $C[0,0]$ and $C[I,J]$, for the case of the best matching training template. The computation of cumulative distance is therefore restricted to elements occurring within a fixed or variable size 'window' around this path (for a full discussion of windowing see [42]). In problems containing very large numbers of training templates (for example speaker-independent problems such as Problem A1) windowing methods do not reduce computation sufficiently, and a *clustering* technique is usually employed. Each word is represented not by the full set of training templates belonging to its class, but by a small number of templates[3] which accurately model the acoustic and temporal variation of the class. In the case of the DTW system described by Woodland [61] each letter of the alphabet was represented by 12 clustered templates. Since the original number of templates per letter was approximately 52 (speakers) $\times 3$ (repetitions) $= 156$, clustering reduced computation by a factor of $\frac{156}{12} = 13$. Unfortunately, clustering also reduces the diversity of the training templates, resulting in a small increase in error rate [21].

[3] Derived automatically from the training templates using a criterion of maximum template diversity.

6.2.2 Comparison of Error Rate with Static Classifiers

The error rate achieved by DTW was compared with those of an optimised KNN and RBF on the small multiple-speaker Problem D, and the full speaker-independent Problem A1. Problem D was chosen because the small number of speakers, combined with the small numbers of training templates (10 speakers, $M = 520$), meant that it was a typical application for unclustered DTW. In contrast, Problem A1 was selected because its large size (52 speakers, $M = 4000$) required the use of clustered DTW to reduce computation to a manageable level. The results are summarised in Table 6.1. In both problems DTW achieved

Table 6.1: *Error rates of DTW and static classifiers.*

Classifier	Error rate (Problem D)	Error rate (Problem A1)
DTW	6.9%	14.0%†
KNN	13.5%	16.8%
RBF	10.0%	10.4%

(† from Woodland [61])

a lower error rate than the KNN. Since both are template-matching algorithms it would appear that the superiority was due to non-linear time normalisation in DTW, as opposed to linear time normalisation in the KNN. The superiority of DTW is most evident in Problem D, suggesting that DTW is capable of forming a good model of temporal variation from small quantities of training data. The KNN performs relatively better on Problem A1 for two reasons. Firstly, the implicit model of temporal variation developed by the KNN (and other static classifiers) improves with greater quantities of training data, and secondly, the clustering procedure applied to DTW in Problem A1 is likely to be detrimental to error rate. Although the RBF employs linear time normalisation it achieved a lower error rate than

DTW in Problem A1. This demonstrates that a static classifier of sufficient generalising power can overcome the limitations of linear time normalisation with a well developed implicit model of temporal variation (at least with the limited vocabulary of spoken letters considered here). In Problem D the error rate achieved by the RBF was higher than that of DTW, demonstrating the need for large quantities of training data to form a good implicit model.

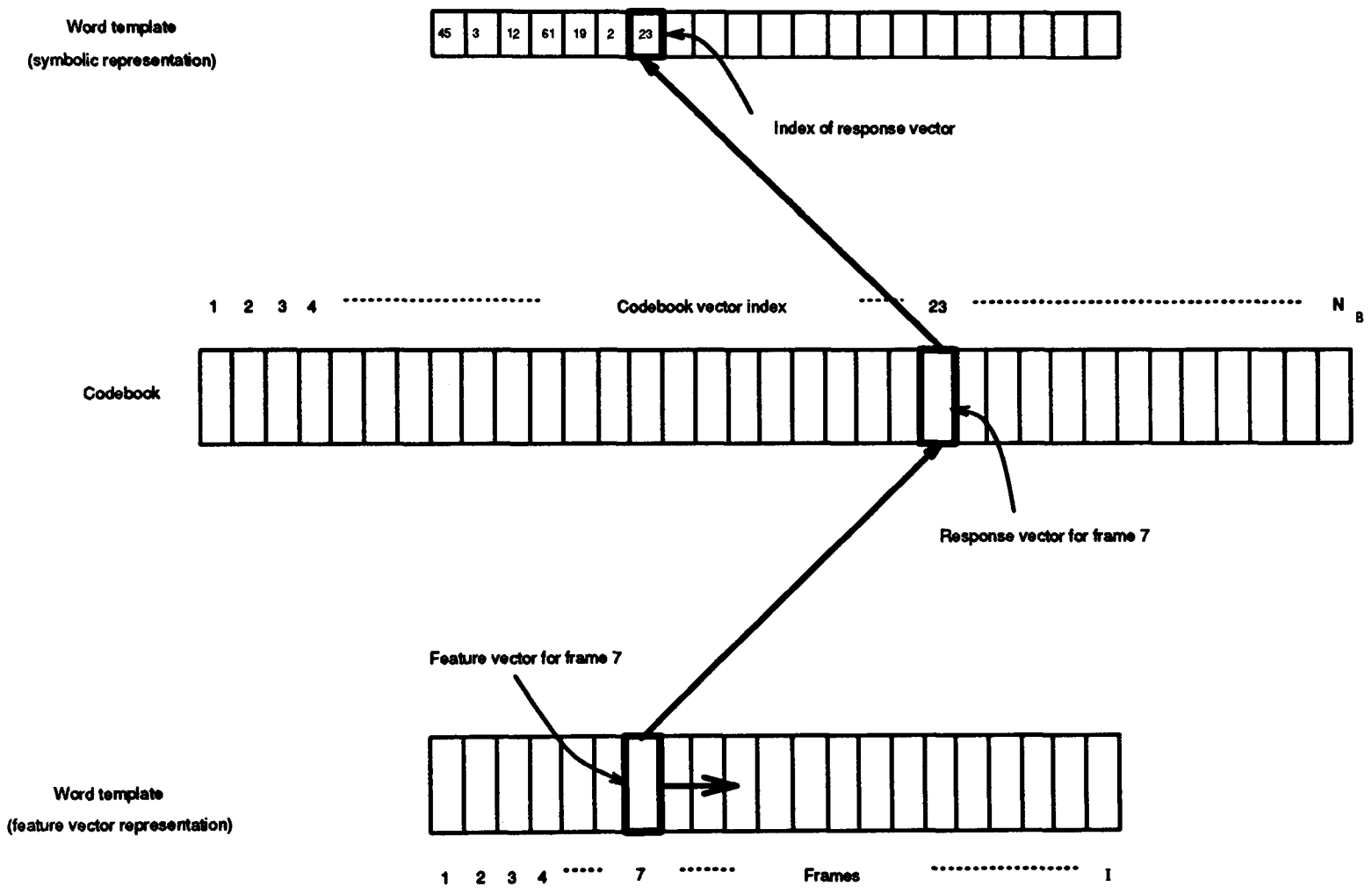


Figure 6.2: *Vector quantisation of a word template. Each feature vector in the word template is represented by the index of the codebook vector which is closest in Euclidean distance (known as the response vector).*

6.3 The SPLIT Method

Although Problem D contains relatively few training templates the mean classification time of DTW is approximately 19 seconds on the present computing system. A delay of this

length between utterance and classification is an extremely undesirable feature for a practical speech recognition device. For this reason *vector quantisation* has been combined with DTW to reduce computation[4] [53, 16]. Although developed independently, the classification method described below is similar to the SPLIT (Strings of Phoneme LIke Templates) method of Sugamura *et al* [53]. However, the method is extended significantly in Section 6.4.

Vector quantisation is a process whereby a multi-dimensional feature vector is represented by one of a set of *codebook vectors* which sample the feature space. Normally feature vectors are represented by the codebook vector which is closest in Euclidean distance, referred to in this thesis as the *response vector*. If $d'_m(i,j)$ is the Euclidean distance between the ith feature vector of the mth training template and the jth codebook vector:

$$d'_m(i,j) = ||\mathbf{f}_i^m - \mathbf{c}_j|| \tag{6.3}$$

then the index of the response vector, j^*, is such that the following condition is satisfied:

$$d'_m(i,j^*) \leq d'_m(i,j) \qquad \forall\ j\ \neq j^* \tag{6.4}$$

If the codebook consists of N_B vectors, each feature vector is therefore represented by an index in the range $1 - N_B$, and a training template by a sequence of such indices (see Figure 6.2). Sugamura *et al* used a variant of the Lindo-Buzo-Gray (LBG) clustering algorithm [32] to develop the codebook vectors. Many alternative algorithms are available however (for a full discussion see [18]). The algorithm used here was the adaptive K-means algorithm of Section 3.3.2, with the exception that 'centres' are now referred to as codebook vectors, and fixed length word templates are replaced by individual feature vectors as training patterns.

[4]Vector quantisation may be used in conjunction with windowing techniques and template clustering procedures.

The objective of the SPLIT method is to significantly reduce the computation of Euclidean distance from Equation 6.1 – the most computationally intensive task during classification. A distance matrix, $D[i,j]$, containing the Euclidean distances *between the codebook vectors* is first computed:

$$D[i,j] = ||\mathbf{c}_i - \mathbf{c}_j|| \tag{6.5}$$

where $\mathbf{c}_i$ is the ith codebook vector. This matrix is symmetric and of dimension $N_B \times N_B$. It need be computed only once after the codebook vectors have been determined and thereafter operates as a look-up-table for the Euclidean distances between response vectors. Classification occurs in two stages. The test template is firstly processed into a sequence of response indices, following which a modified version of Equation 6.1 is applied, the $d(i,j)$ terms having been replaced by the elements of the look-up-table.

6.3.1 Comparison of Classification Time with DTW

Sugamura *et al* show that the ratio of the time taken to perform the distance computation between DTW and the SPLIT method is given by:

$$\lambda = \frac{M\hat{I}}{N_B} \tag{6.6}$$

where $\hat{I}$ is the average number of frames in the training templates, which in the case of the CONNEX database is 37.5 (the average utterance length is 0.48 seconds, and frames are separated by 12.8 ms). For values of N_B in the range 64–400, the time taken to perform the distance computation is reduced by a factor of between 305 and 49. However, Sugamura *et al* failed to explain that classification time is not reduced by this ratio in practice due to the additional calculation required to support the distance computation. Table 6.2 gives a description of the main computing operations required by the two methods during classification. On the present computing system additions (a) require one processor

Table 6.2: *Computing operations required for classification.*

Method	Computing	operations	Description
DTW	$M\hat{I}J$	[a]	access element $C[i,j]$
	$M\hat{I}J$	[2a]	select minimum path
	$M\hat{I}JN_I$	[m + 3a]	Euclidean distance between feature vectors
	$M\hat{I}J$	[a]	assign element $C[i,j]$
SPLIT	$M\hat{I}J$	[a]	access element $C[i,j]$
	$M\hat{I}J$	[2a]	select minimum path
	N_BJN_I	[m + 3a]	Euclidean distance to codebook vectors
	N_BJ	[a]	select response vector
	$M\hat{I}J$	[a]	access element $D[i,j]$
	$M\hat{I}J$	[a]	assign element $C[i,j]$

clock cycle, and multiplications (m) require 16 processor clock cycles. The true ratio of computation between the two methods can therefore be expressed in terms of processor clock cycles as follows:

$$\lambda' = \frac{M\hat{I}J(1+2+19N_I+1)}{M\hat{I}J(1+2+1+1)+N_BJ(19N_I+1)} \tag{6.7}$$

Cancelling through by J (the number of frames in the test template), and substituting the constants $M = 520$, $N_I = 8$, and $\hat{I} = 37.5$, the ratio reduces to:

$$\lambda' = \frac{3042000}{97500+153N_B} \tag{6.8}$$

For values of N_B in the range 64–400, the SPLIT method reduces computation by a factor of between 28 and 19 relative to DTW. This reduction in computation was borne out in practice, classification time dropping from 19 seconds to approximately one second in the slowest case when $N_B = 400$. Clearly this reduction is less than estimated by Sugamura *et al*, but is sufficient to enable the advantages of non-linear time normalisation to be applied to large speaker-independent problems without the use of clustered training templates [53, 16].

6.3.2 Comparison of Error Rate with DTW

Because the accuracy with which the codebook vectors sample the feature space is finite, the SPLIT method cannot be expected to attain as low an error rate as DTW. However, the accuracy improves with increasing codebook size, and may be quantified by the average distortion between the feature vectors and response vectors over all training templates:

$$D = \frac{1}{M\hat{I}} \sum_{m=0}^{M} \sum_{i=0}^{I_m} d'_m(i, j^*) \tag{6.9}$$

where I_m is the length of the mth training template. Figure 6.3 illustrates the distortion for values of N_B in the range 64–400, showing that distortion decreases as N_B is increased.

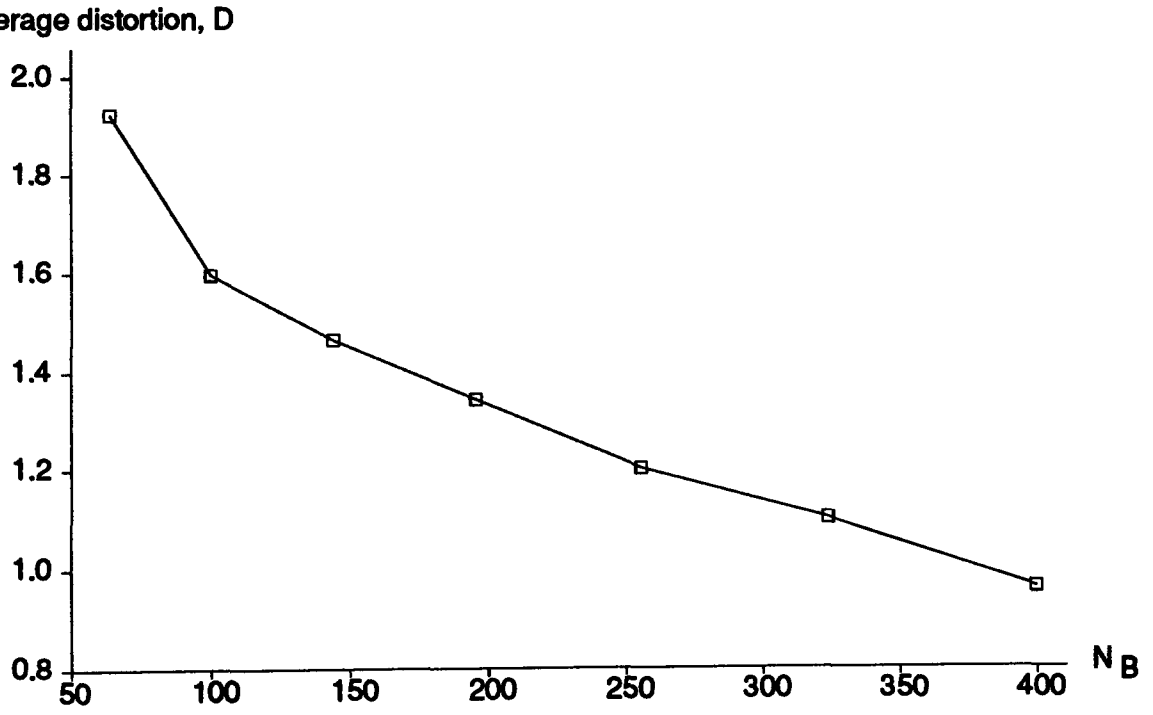


Figure 6.3: *Average distortion vs N_B for SPLIT method.*

Figure 6.4 illustrates the error rate of the SPLIT method over the same range of N_B. As expected, the distortion and error rate are strongly correlated. Comparison with Table 6.1 reveals that in Problem D, the error rate of the SPLIT method is less than that of the optimised RBF for $N_B \geq 170$.

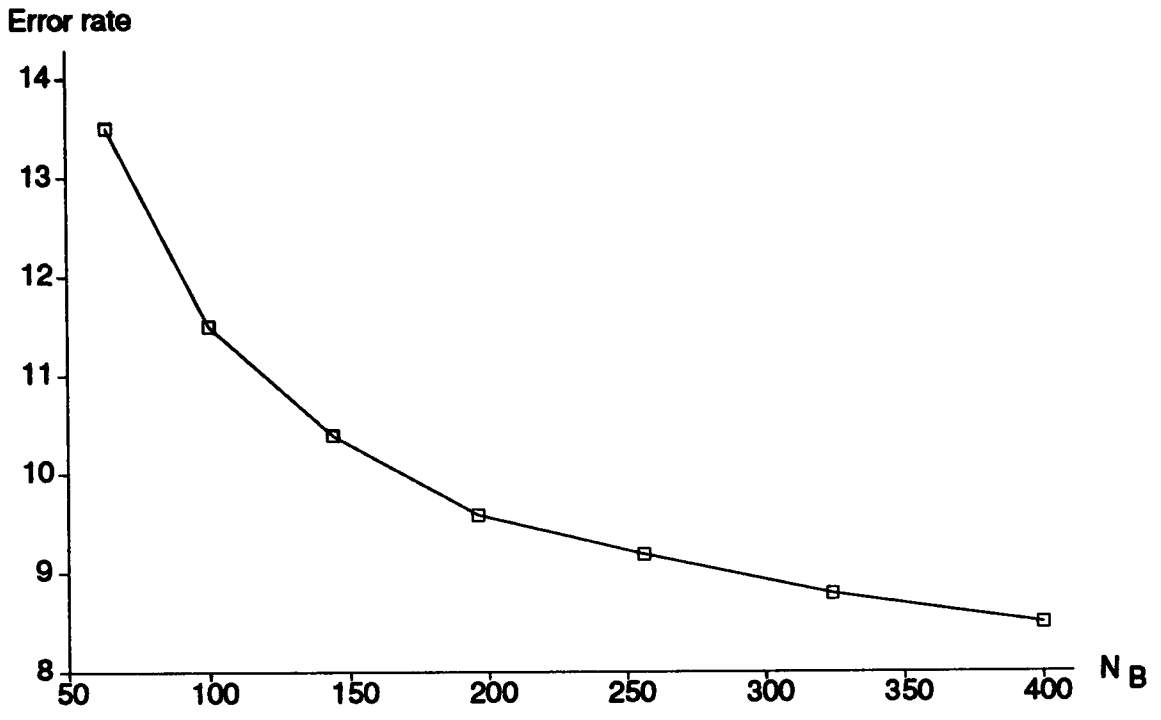


Figure 6.4: *Error rate vs N_B for SPLIT method.*

6.4 Representation of Words as Trajectories

In the system described above codebook vectors are developed using the adaptive K-means algorithm. An alternative to this algorithm is Kohonen's two-dimensional feature map (see Section 3.3.2) which was shown in Section 4.4 to sample training data with the same level of accuracy. The advantage of using the feature map is that the codebook vectors become *spatially ordered*; the acoustic similarity of the codebook vectors is reflected in their arrangement in the map data structure. Since the rate at which feature vectors are extracted is high relative to the rate of articulation of speech, the acoustic properties of the feature vectors change slowly over time. A consequence of spatial ordering is that the *positions in the map* associated with successive response vectors (i.e. the (x,y) coordinates of the codebook vectors closest in Euclidean distance to the feature vectors) change slowly, defining a smoothly varying *trajectory* in the map for each word template (see Figure 6.5).

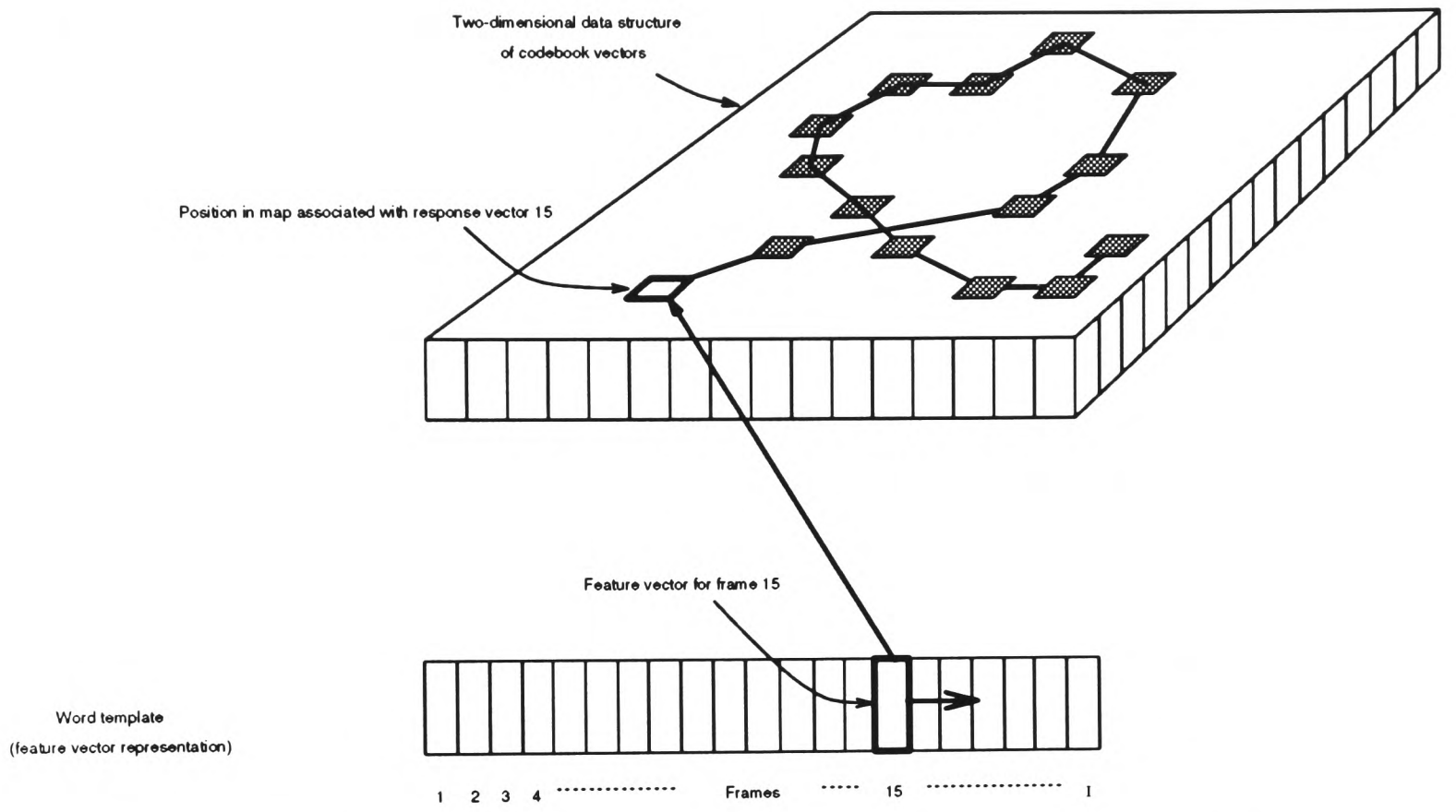


Figure 6.5: *Representation of a word template as a trajectory on the surface of a two-dimensional feature map consisting of 256 codebook vectors arranged as a* 16×16 *data structure. To enhance visualisation, shaded boxes denote the positions in the map of the response vectors and lines connect the positions of response vectors which are consecutive in time.*

6.4.1 Options for Classifying Trajectories

Two options arise for classifying the trajectories of test templates:

1. Classification is based on the trajectories directly.

2. Classification is based on the underlying feature vectors using the SPLIT method.

If the first option is selected, several possible strategies are available. One strategy is to extract the series of (x,y) coordinates describing each training trajectory, and perform DTW on the coordinates of the incoming coordinates of the test trajectory. Unfortunately this strategy is flawed since no direct correspondence exists between geometrical distance in the map and Euclidean distance in the space of the feature vectors (see Section 3.3.2). As

a result the geometrical distances between trajectories on the map only *qualitatively* reflect the similarity of their underlying feature vectors, the extent of the correspondence varying unevenly throughout the map. This effect was demonstrated by the poor performance of this method in comparison with either DTW or the SPLIT method; an error rate in excess of 20% was obtained on Problem D.

Kohonen [27] proposed an alternative strategy in which a database of labelled phoneme tokens was used to calibrate codebook vectors with one of 18 Finnish phonemes. The feature map consisted of 120 codebook vectors arranged into a 10 × 12 data structure as shown in Figure 6.6. The labels associated with the response vectors of the test trajectory were

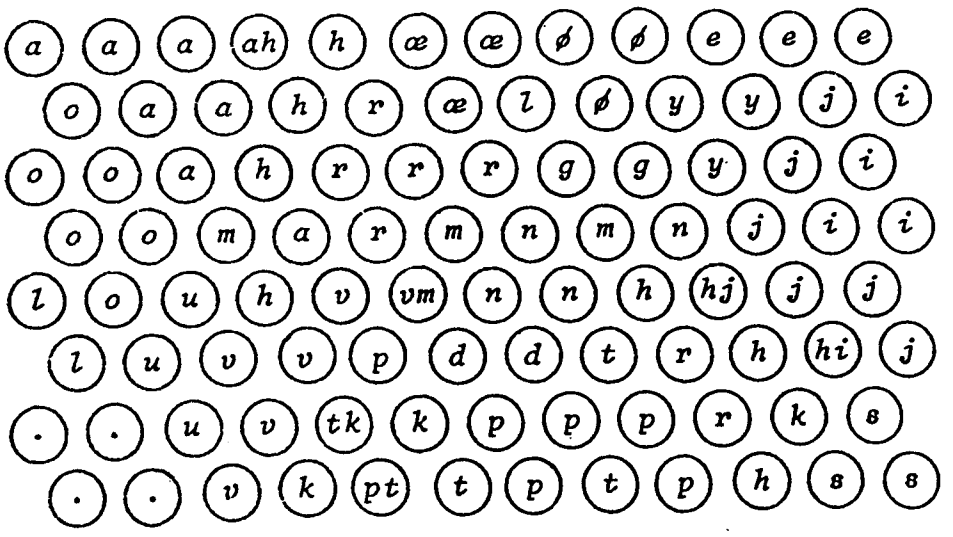


Figure 6.6: *Kohonen's calibrated map of Finnish phonemes (reproduced from [27]).*

then concatenated into a long[5]stream of phonetic symbols from which the most likely word candidate was obtained by a tree search technique known as 'dynamically expanding context' [26]. On a database of 1000 Finnish words spoken by several male speakers, Kohonen was able to achieve an error rate of under 10%. There are three important reasons why his system was able to achieve such a low error rate with this large vocabulary:

1. The Finnish language contains only 18 phonemes. Therefore the likelihood of phonetic misclassification is inherently smaller than for the English language which contains between 40 and 60 phonemes (depending on the definition adopted).

2. Finnish is an *orthographic* language. Post-processing of phoneme streams into words occurs directly, without the difficulties encountered with English spelling.

3. The codebook vectors were calibrated after extensive fine tuning with a method known as Learning Vector Quantisation, offering superior performance to the standard feature map algorithm [28].

It therefore seems clear that the excellent results achieved by Kohonen on the Finnish database would not be repeated on an English database of similar specification.

If classification is based on the underlying feature vectors (option number two at the beginning of Section 6.4.1) the error rate is similar to that obtained using the adaptive K-means algorithm (a minimum of 8.8% as opposed to 8.5%), in which codebook vectors are not spatially ordered. Thus no advantage is gained by using spatially ordered codebook vectors, with the result that the adaptive K-means algorithm is favoured since it requires marginally less computation.

[5]Some smoothing criterion is normally introduced so that a phoneme is only recognised after K successive response vectors have indicated its presence, where $1 \leq K \leq 10$.

6.5 Summary of Dynamic Classification

The superior performance of DTW relative to the KNN for both Problems A1 and D, demonstrates the value of non-linear time normalisation in template matching techniques. However, the RBF achieved a lower error rate than DTW on Problem A1 – a problem for which sufficient training data was available to develop a good implicit model of temporal variation. It is clear that DTW is especially suited to problems containing small amounts of training data (e.g. single or multiple-speaker problems). In contrast, the RBF is best suited to problems containing extensive amounts of training data (e.g. speaker-independent problems).

A dynamic classifier, known as the SPLIT method, is obtained by combining DTW with vector quantisation – a technique which is often categorised as 'neural network processing' since many algorithms for vector quantisation can be incorporated into a parallel architecture (see [55] for example). On problem D, the SPLIT method required over an order of magnitude less computation than DTW to perform classification, with near-comparable error rate. Although this reduction is significant, it is considerably less than that claimed by Sugamura *et al* [53]. With the present availability of special purpose hardware (e.g. DSP chips) for implementation of DTW, it appears that this reduction in computation is not sufficient to warrant replacing DTW by the SPLIT method. This is particularly true for small-scale problems. In larger speaker-independent problems the SPLIT method would also be ruled out in favour of a neural network classifier.

In the SPLIT method, if a standard method of vector quantisation is replaced by the two-dimensional feature map, it is possible to represent words, and indeed all types of speech sounds, as trajectories on the feature map. However, the error rate of the resulting classifier

is not improved by this modification, and for the sake of computational efficiency a standard method of vector quantisation would normally be preferred. We shall see in Chapter 7 however, that in specific applications there are convincing reasons for representing speech sounds as trajectories on a feature map.

Chapter 7

Application: Visual Feedback Therapy

7.1 Introduction

Kohonen's feature map algorithm has so far been evaluated in two contexts: as a method for allocating centres in the RBF, and as a method for allocating codebook vectors in the SPLIT method. In both cases however, the adaptive K-means algorithm was used in preference to the feature map algorithm since the former was able to achieve a similar error rate with slightly less computation. Nevertheless it was shown that generalisation was not degraded by imposing the constraint of spatial ordering. In this chapter an application is introduced for which a spatially ordered representation of speech is extremely valuable, namely the *visual interpretation* of speech by humans. Visual interpretation of speech is particularly valuable to humans suffering from a range of communication (speech and hearing) disorders, as it enables them to literally 'see' aspects of their voice. The most obvious example is the teaching of pronunciation to deaf people who have no normal means of assessing their voices, nor of adjusting their voices to be like those of normal listeners (pronunciation is learnt effortlessly by normal listeners during everyday verbal interaction).

7.2 Visual Feedback Therapy

The use of visual representations of speech in the treatment of communication disorders is referred to as *visual feedback therapy.* A number of well established techniques exist for visual feedback therapy, the most sophisticated of which is known as *electropalatography (EPG)* [20]. In this technique an artificial palate made of a hard[1] acrylic material is inserted onto the palate of the client. Embedded in the artificial palate are large numbers of silver electrodes (≈ 60) arranged in horizontal rows covering the regions of the palate most used in articulation. The client grasps an electrode to which a low voltage sinusoid is applied, and the resultant voltages are then recorded from the electrodes in the artificial palate. Electrode voltage depends on the conductivity between the artificial palate and the body, which is determined by the presence or absence of contact with the tongue. The outputs of the electrodes produce a coarse image of tongue contact during speech articulation, which is transmitted to a host computer for display (the electrodes are scanned between 100 and 200 times per second – similar to the rate required for acoustic analysis). The tongue movements of the client may be compared with those of a trained therapist in order to diagnose speech defects and suggest methods for their correction.

Unfortunately, EPG suffers from the following problems:

- It can only be used to investigate those phonemes for which tongue-palate contact occurs, which accounts for approximately half of the phonemes in the English language.

- It provides an accurate description of articulation in terms of tongue-palate *contact.* Information concerning tongue-palate proximity, or which part of the tongue is in contact with the palate, is not provided.

[1] Flexible palates are currently under development.

- The image of tongue contact has limited resolution due to the finite number of electrodes in the artificial palate.
- A unique artificial palate must be constructed for each client. Insertion of the artificial palate into the mouth can impede speech and cause discomfort. The period of habituation necessary before therapy may begin is dependent on the client, some clients rejecting use of the palate.
- The final system is complex and costly.

Despite these problems, EPG has been shown to reveal sub-phonemic defects of articulation which have not been detected by ordinary acoustic analyses, or traditional therapy techniques[2].

A less sophisticated technique known as *electrolaryngography (ELG)* [1] recovers the waveform of the fundamental frequency component of speech, referred to as the larynx excitation, or Lx, waveform. This is accomplished by recording the variation in electrical impedance between a pair of gold-plated electrodes placed on the skin at the level of the thyroid cartilage. The Lx waveform is then processed by a host computer to obtain a value of pitch (excitation frequency) of greater accuracy than can be obtained via acoustic methods. Both the Lx waveform and pitch are displayed in real time on a computer screen, enabling simple comparison between the client's and therapist's speech. Knowledge of the Lx waveform is helpful in diagnosing defects of voice control, and knowledge of pitch has been found to be a very useful aid to clients with hearing difficulties. However, such information does not constitute a full description of articulation and must be supplemented

[2]Traditional speech therapy techniques include the use of mirrors for visual feedback, explanatory diagrams of the articulatory system, mechanical methods for positioning the client's articulators, verbal descriptions of articulation, etc [20].

with traditional therapy techniques.

7.3 Spatial Ordering

In the case of a two-dimensional feature map trained on feature vectors derived from short-term acoustic properties of speech signals, spatial ordering of the codebook entries results in acoustically similar sounds being represented in similar regions of the map. The feature map may be trained on feature vectors derived from many pre-processing techniques. Combinations of time domain features (e.g. short-term energy, zero-crossing rate, pitch, degree of voicing), frequency domain features (e.g. filter-bank outputs, LPC coefficients), or others depending on which aspects of the speech signal it is desired to represent. In the system described in this chapter, feature vectors were derived using mel-frequency cepstral analysis as previously described in Section 2.4.1, with the exception that a simpler (energy-based) algorithm was used to obtain the endpoints of the words, and the frame interval was reduced to 8 ms (see Section 7.4.2). This form of analysis is particularly suitable for training feature maps for visual interpretation for the following reasons:

1. Cepstral coefficients provide a compact description of the spectral envelope of speech (see Appendix B), which is determined by the modulation of the speech signal by the vocal tract, and is largely uninfluenced by its excitation source[3]. Thus cepstral coefficients describe the *method of articulation* of speech – precisely the information valuable to the hearing or speech impaired client.

2. Cepstral coefficients offer a significant reduction in the variation between speakers of different sex, in comparison with, say, the outputs of a filter-bank, since information

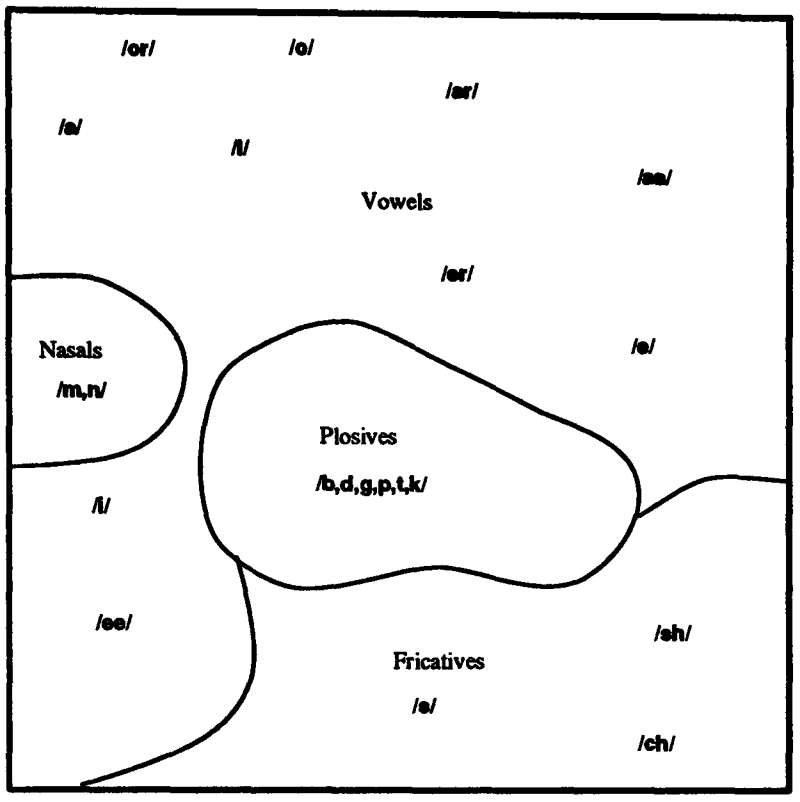
[3]Due to sub-glottal coupling the spectral envelope is influenced by the excitation source. However, the effect is only of secondary importance.

regarding the excitation source of the signal is removed. The representation of speech developed by a feature map trained on feature vectors derived from cepstral coefficients is therefore not significantly affected by speaker sex.

3. Cepstral analysis facilitates the recovery of a good estimate of pitch, degree of voicing, and short-term energy. These features constitute important supplementary sources of information to the client.

The feature map representation leads to a natural segmentation of speech signals into regions representing vowels, fricatives, nasals, glides, etc. These regions may be further segmented into areas representing individual phonemes. It should be noted that *all* phonemes are represented in the map if it is trained on a diverse selection of speech sounds which is representative of the language. A two-dimensional feature map containing 144 codebook vectors (arranged in a 12 × 12 data structure) was trained on the feature vectors obtained from single utterances (by the author) of the one hundred word database listed in Appendix D. This database provided approximately 3000 feature vectors. Since the feature vectors were unlabelled[4] it was not possible to calibrate the map using the automatic procedure described in Section 4.4.1 (note that the calibrated map presented in Chapter 4 was trained on fixed length word templates, not feature vectors, with the result that different regions of the map were representative of different classes of *word*, not different classes of *phoneme* as occurs here). A less sophisticated calibration procedure was therefore applied, in which the positions of the response vectors in the trained map were recorded when single-phoneme sounds were spoken into the system. The results of this procedure are illustrated in Figure 7.1. Note that the region representing plosives is not subdivided into phonemes since plo-

[4]Transcription of words into sequences of labelled phonemes requires the services of an expert transcriber.



Vowel	Example
/a/	who'd
/or/	hawed
/ar/	hard
/er/	heard
/aa/	had
/e/	head
/I/	hid
/ee/	heed
/i/	ball

Figure 7.1: *Representation of phonemes by calibrated map. The calibration was performed by speaking single phonemes into the system and recording the positions in the map associated with the response vectors. The map may be broadly segmented into regions representing different categories of phonemes. This calibration procedure is only approximate since appreciable overlap may occur between the regions. Note that the labelled vowel sounds are defined by example words.*

sives have transient spectra which cannot be represented at single positions in the map; all plosives are represented as short trajectories within the plosive region.

7.4 Enhancement of Word Trajectories

We saw in Section 6.4 that the sequence of response vectors generated by each word template gives rise to a unique trajectory in the feature map. Trajectories offer an instant[5] method for visualising the *acoustic-dynamics* of a word; the acoustic properties of the word are revealed by the positions in the map associated with the response vectors, and the dynamic properties by the *order* in which the response vectors appear. Figure 7.2 presents some

[5]Word trajectories were displayed in real-time on the present computing system.

examples of word trajectories which were observed on the feature map of Figure 7.1. Note that the word 'mash' is a phonetically reversed version of the word 'sham'. Consequently both words produce similar *shaped* trajectories, although they occur in opposite *directions.* The word 'whey' is included to illustrate how the trajectories of dissimilar words are quite different in shape.

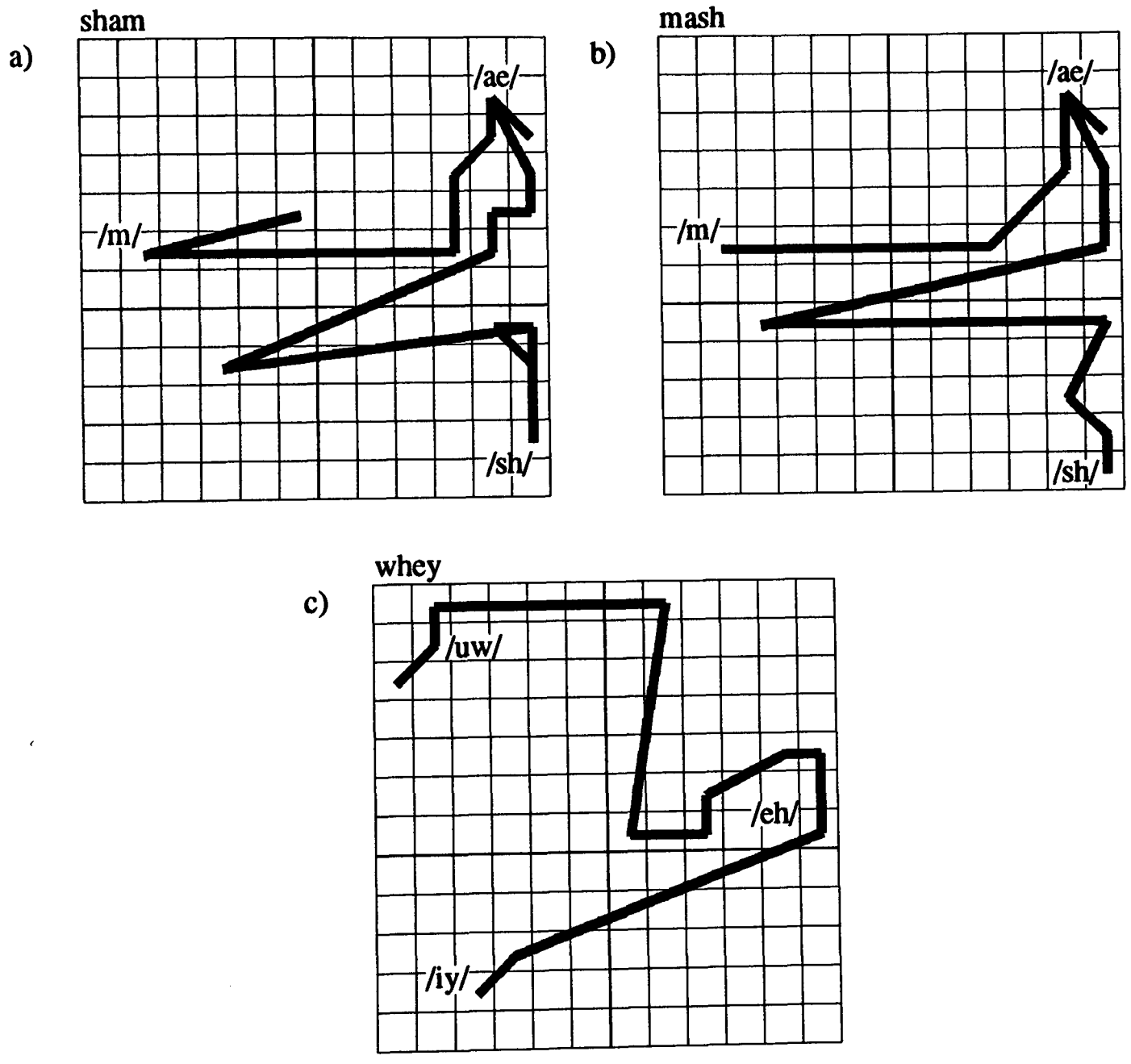


Figure 7.2: *Example trajectories for the words a) 'sham', b) 'mash', and c) 'whey'. The trajectories for 'sham' and 'mash' are similar in shape, but opposite in direction, whilst the trajectory for 'whey' is quite different in shape. Note that the trajectories are labelled phonetically to aid interpretation.*

7.4.1 Trajectory Smoothing

The trajectories examined so far are produced by linking together the sequence of response vectors associated with the feature vectors of a word template. It will be recalled that a response vector is the codebook vector with minimum Euclidean distance to a particular feature vector. If, for instance, two codebook vectors had almost equivalent Euclidean distance to a feature vector, only the position associated with the closest would affect the trajectory, despite the fact that the second codebook vector was almost as representative of the feature vector.

An alternative method for computing the trajectory is based on the relative importance of *all* codebook vectors. The Euclidean distances between a feature vector and the codebook vectors are first transformed into *responses*, in a similar manner to the transformation of input patterns in the first layer of the RBF. The response of the jth codebook vector to the ith feature vector of a word template is given by:

$$\Phi_{ij} = \exp\left(-\frac{d'(i,j)}{\nu}\right) \tag{7.1}$$

where $d'(i,j)$ is the Euclidean distance between the vectors (see Equation 6.3), and ν is a parameter affecting the *smoothness* of the trajectory. The Φ_{ij} terms are then arranged into a discrete response surface according to the positions in the map associated with the codebook vectors. The coordinates of the *centre of mass* of this surface are then computed. Figure 7.3 illustrates the trajectories obtained with the original and centre of mass methods for a single utterance of the word 'mash'. In this case the smoothness parameter, ν, was varied until the trajectories were subjectively deemed to be of highest clarity.

Figure 7.4 illustrates how the shape of the same trajectory is affected by alternative values of ν. As ν tends to zero the coordinates of the centre of mass of the response

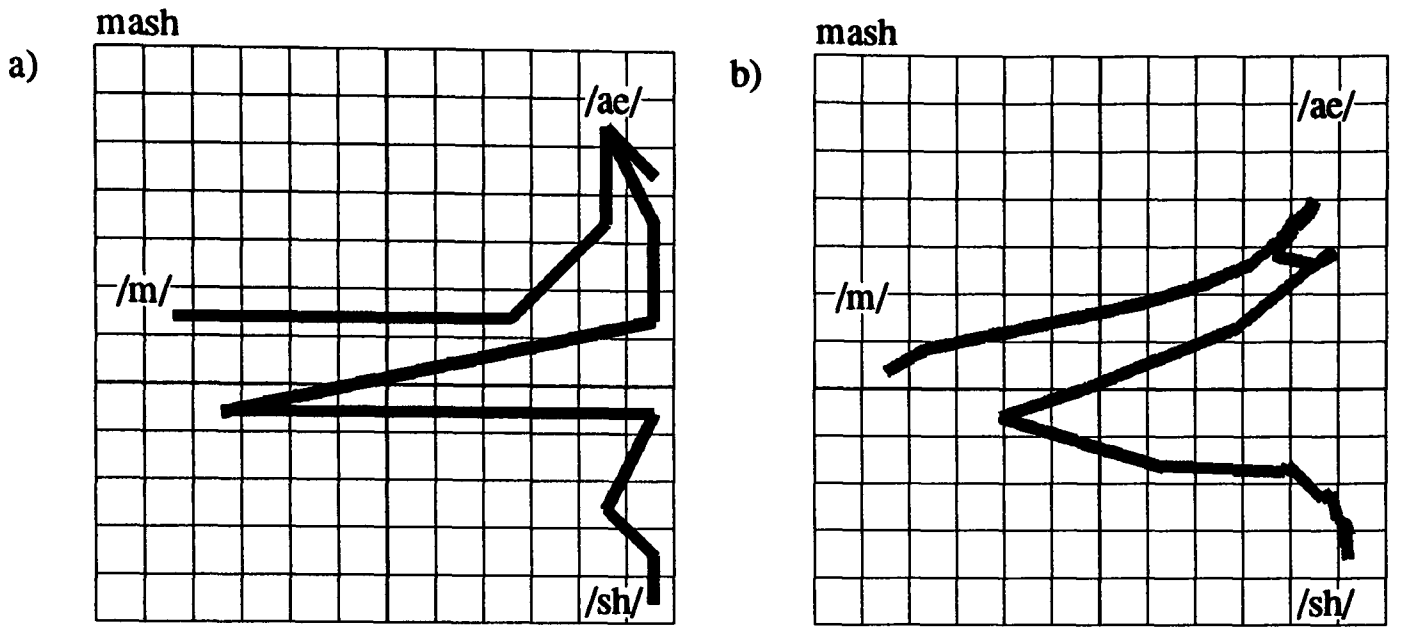


Figure 7.3: *Enhancement of trajectory shape with centre of mass method. Trajectories for the word 'mash' obtained with a) original method, and b) centre of mass method. The original method produces a more erratic, and therefore less informative trajectory than the centre of mass method.*

surface converge to the position of the peak Φ_{ij}. As this is simply the position in the map associated with the response vector, both methods produce similar trajectories for low values of ν (Figure 7.4a). As ν becomes large the response surface is flattened out, and the coordinates of the centre of mass converge to the central point of the map. Thus all trajectories are constrained to lie within a small radius of the centre of the map (Figure 7.4b).

7.4.2 Invariance to Temporal Variation

If it is desired that a single trajectory be associated with each word, trajectory shape should not be dependent on temporal variation (i.e. variation in word duration or the relative rate of utterance of phonemes). This can be achieved if feature vectors are extracted at a higher rate than is necessary to fully capture the transient properties of the speech signal (i.e. < 5 ms) [55]. Under these conditions the transient properties of the speech signal are *oversampled.* Response vectors are therefore selected for longer periods of time with slowly

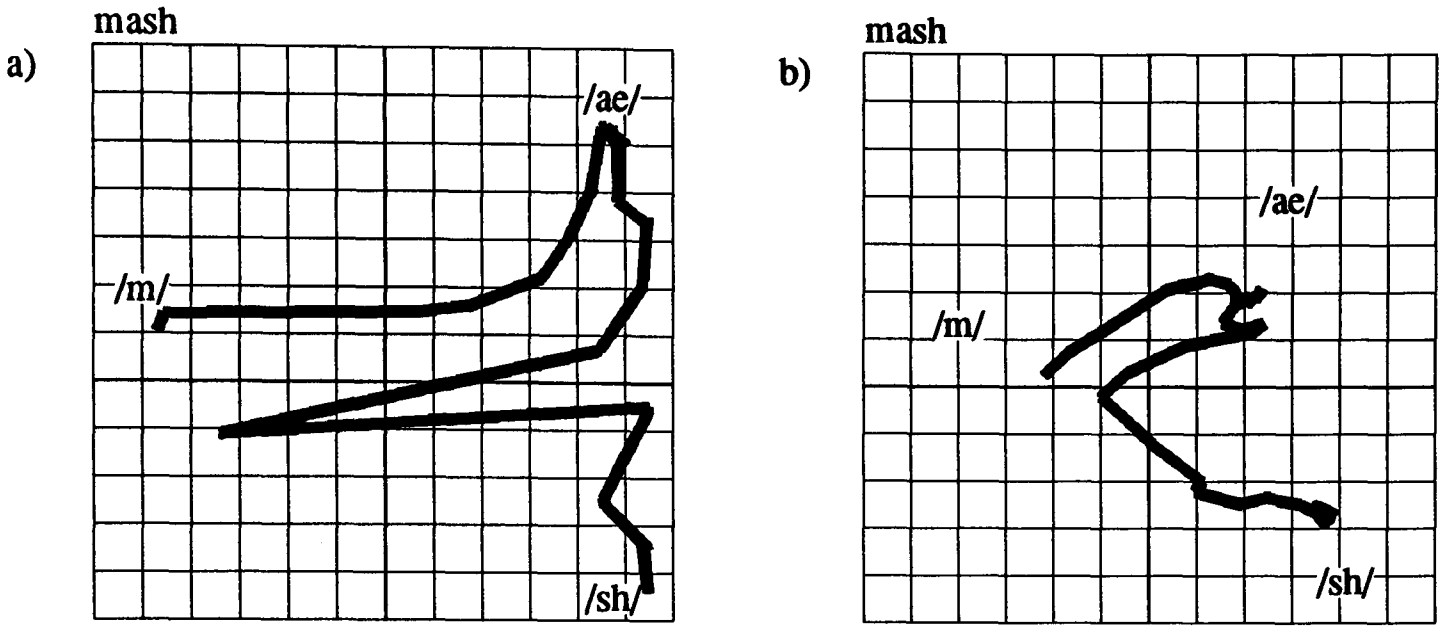


Figure 7.4: *Effect of smoothness parameter, ν, on trajectory shape. a) Undersmoothed trajectory (small ν) similar in shape to the trajectory obtained with the original method. b) Oversmoothed trajectory (large ν) constrained to lie within a small radius of the centre of the map.*

articulated speech than with quickly articulated speech, trajectory shape remaining the same in both cases.

The interval between successive feature vectors (frame interval) employed in this thesis was 8 ms, which is slightly greater than ideal. However, a significant invariance to temporal distortion was observed. Figure 7.5 illustrates the trajectories for two utterances of the word 'were' spoken at deliberately different rates by the author. Invariance to temporal variation also increases the uniformity of trajectories uttered by different speakers. Figure 7.6 shows the trajectories for two utterances of the word 'were' by one male and one female speaker (note that mel frequency cepstral analysis also plays a significant role in increasing uniformity between different speakers).

7.4.3 Intensity Coding

In order to assist the interpretation of the trajectories additional sources of information concerning the speech signal may be encoded by the *intensity* (grey-level) with which they

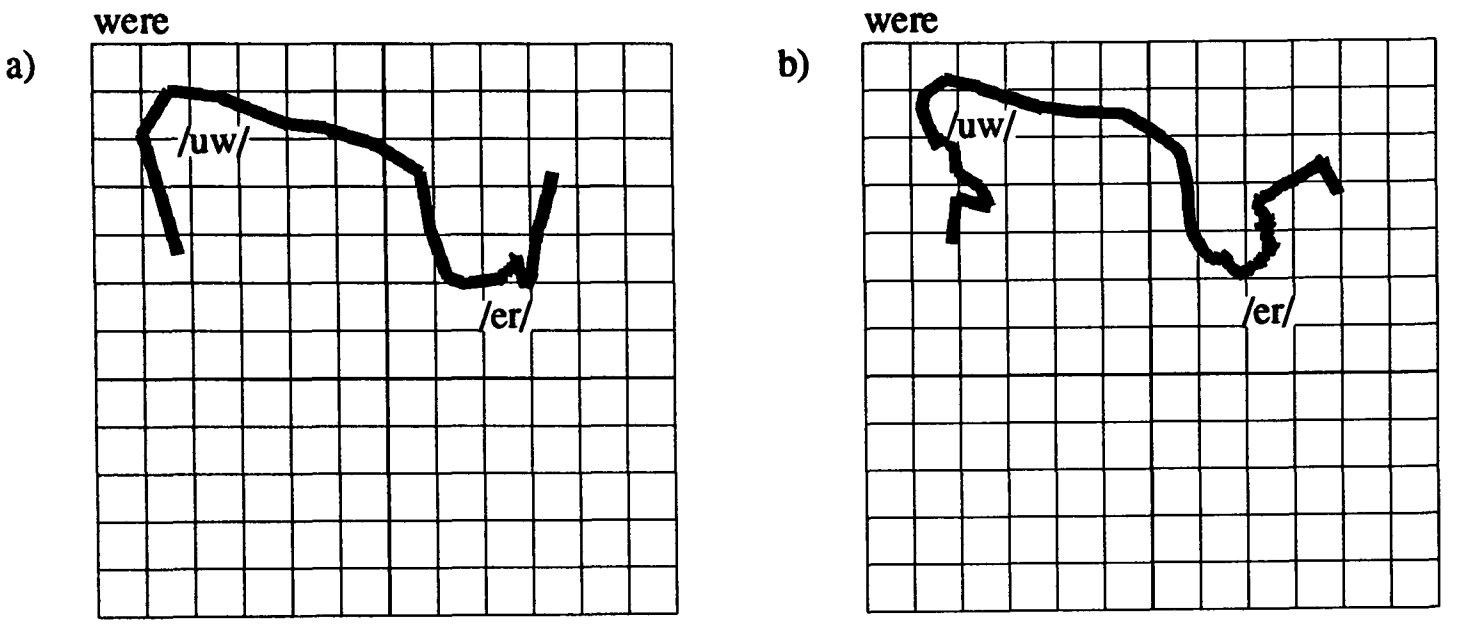


Figure 7.5: *Invariance of trajectories to temporal variation. Trajectories for the word 'were' uttered a) quickly (20 frames), and b) slowly (82 frames). Note the similarity of the resulting trajectories.*

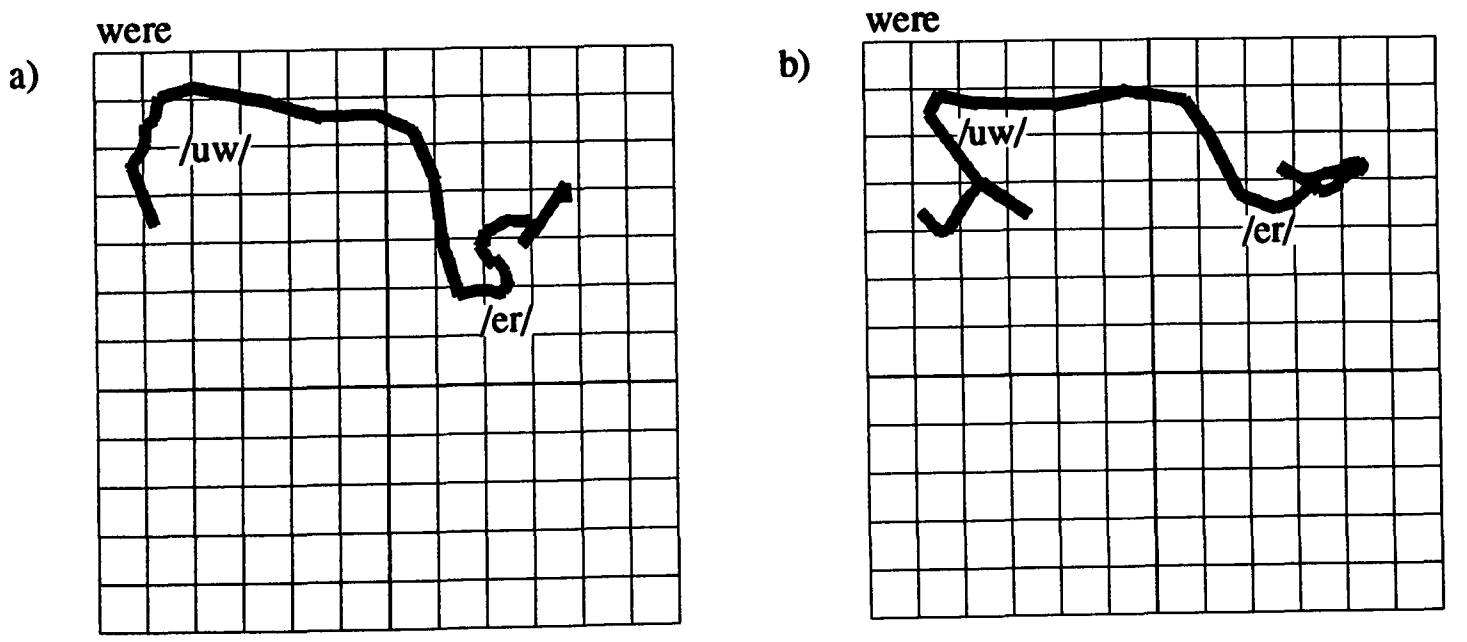


Figure 7.6: *Uniqueness of trajectory representation. Trajectories for the word 'were' uttered a) by a male speaker, and b) by a female speaker. Note the similarity of the resulting trajectories.*

are displayed. In this section intensity coding of the *short-term energy*, and *rate of spectral change* profiles of the speech sounds is examined. However, a simpler form of intensity coding is firstly considered.

Words containing a plosive phoneme contain a short period of silence preceding the plosive burst. The path of the trajectory becomes unstable during this period since the speech signal is masked by background environmental noise. A simple method for removing the silence region from the trajectory is to display the trajectory only when the short-term energy of the speech signal exceeds the energy level of the background environmental noise (see Figure 7.7). This is equivalent to imposing an intensity threshold on the generation of the trajectory, and is a particularly useful method for revealing the locations of plosive phonemes. The technique above may be extended to reveal the short-term energy profile of

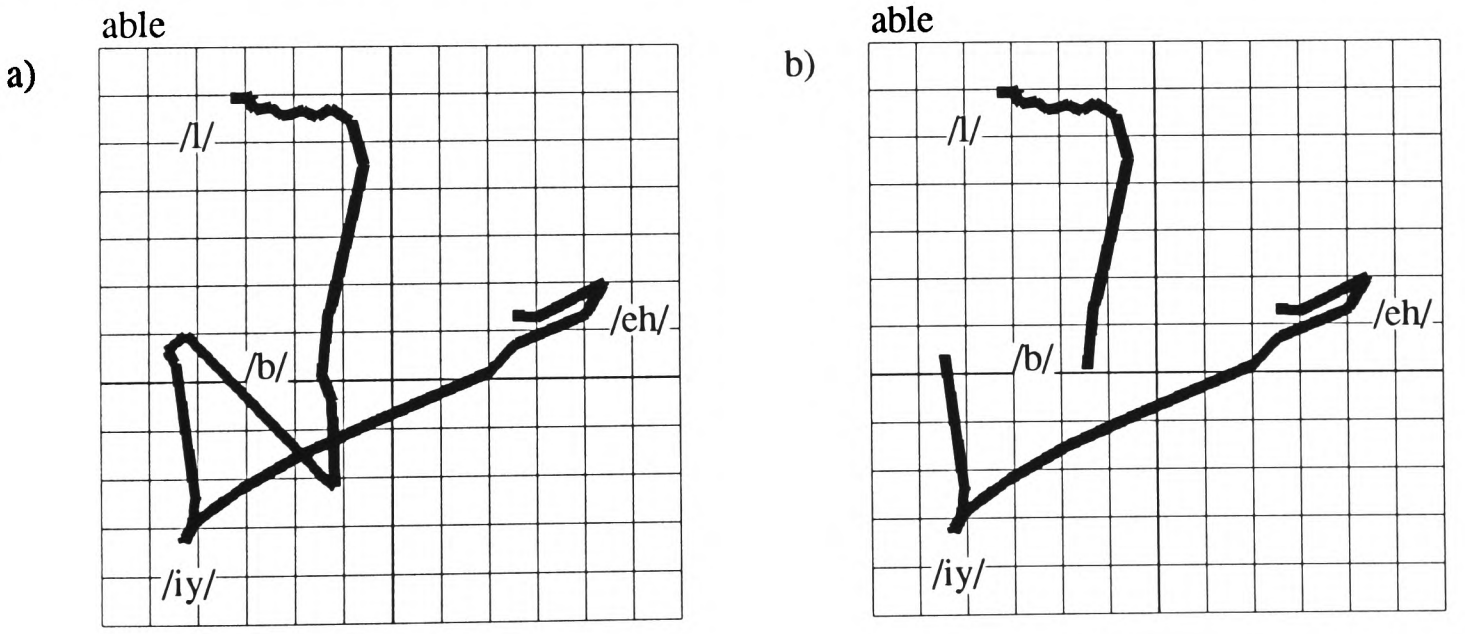


Figure 7.7: *Removal of silence region from a trajectory for the word 'able'. a) The original trajectory. b) The trajectory from which the silence region before the 'b' has been removed.*

the word, whereby the grey-level used to display the trajectory is determined by the energy (in decibels) of the speech signal within each frame (see Figure 7.8).

The rate of change of the speech spectrum may also be encoded in the intensity of the

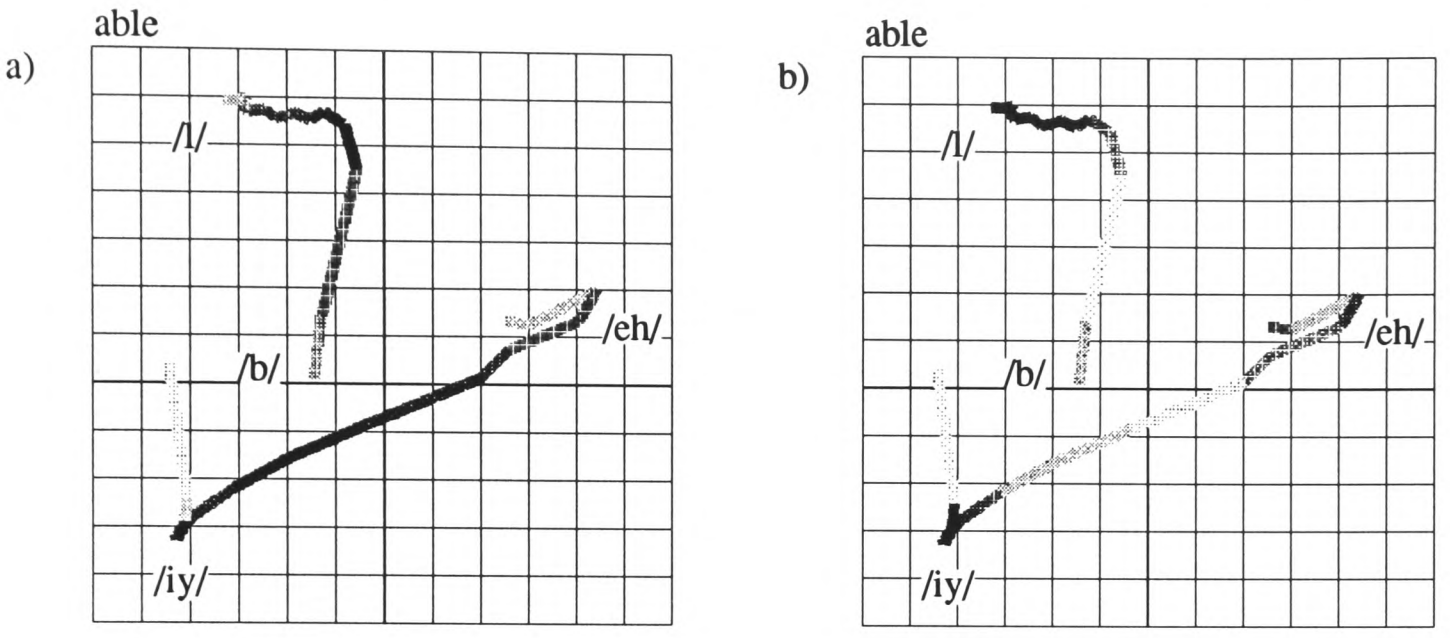


Figure 7.8: *Intensity coded trajectories. The trajectories correspond to a single utterance of the word 'able'. a) The short-term energy profile of the word is encoded in the trajectory intensity. b) The rate of change of the speech spectrum is encoded in the trajectory intensity. Note that the silence regions have been removed from both trajectories.*

trajectory, to reveal the stationary and transient portions of the speech signal – information which cannot be recovered from trajectory shape. The rate of spectral change is proportional to the Euclidean distance between successive feature vectors:

$$\Delta \mathbf{f}_i = ||\mathbf{f}_i - \mathbf{f}_{i-1}|| \qquad 2 \leq i \leq I_m \tag{7.2}$$

An effective method for displaying this information is to define an inverse relationship between $\Delta \mathbf{f}_i$ and intensity, such that periods of rapid spectral change (transient phonemes) are displayed with low intensity, and periods of slow incremental change (stationary phonemes) are displayed with high intensity. Rate of change intensity coding clearly illustrates the *quasi-static* nature of the speech signal to the viewer (see Figure 7.8b).

7.5 The Visual Ear

It is proposed that the trajectory representation of speech sounds (words, syllables, and phonemes) form the basis of a novel system for visual feedback therapy. The system will be

referred to as the *Visual Ear* as it is a method for interpreting speech visually, not aurally. The Visual Ear is a natural extension of the trajectory visualisation system developed earlier in this chapter. In particular, the centre of mass method is used to derive trajectory path, regions of the trajectory corresponding to silence are removed, and the rate of change of the speech spectrum is coded in trajectory intensity. The Visual Ear described below is based on a feature map trained with mel-frequency cepstral coefficients, but there is no reason why it could not be used to display other aspects of the speech signal if alternative features were used during training. In this application the feature map requires a large corpus of speech sounds for training, much larger than the single speaker, one hundred word database used for the examples given earlier in this chapter. The training corpus should include a diverse selection of speech sounds from speakers representing both sexes and a large range of ages and regional accents. Provided that the diversity of the training corpus is consistent with the expected diversity of the therapists and clients who will use the system, the resulting feature map should provide an accurate representation of their speech. The training corpus could equally well be derived from continuous speech utterances, or isolated words. Note that a Visual Ear could be developed for any language provided that a sufficiently large corpus of speech was available to train the feature map.

7.5.1 Therapy Technique

The facility to display *two* trajectories simultaneously has been built into the Visual Ear to allow instant comparison between the speech of a client and the speech of a trained speech therapist. The trajectory produced by the therapist is known as the *target* trajectory, and is permanently displayed on the computer screen. The client then attempts to reproduce this trajectory with their own voice. Two steps are taken to improve the clarity of the display:

1. The client and target trajectories are displayed in different colours[6].

2. Each utterance by the client is preceded by the removal of their previous utterance.

Using traditional techniques of speech therapy the therapist informs the client how to improve their pronunciation, the Visual Ear offering an instant qualitative technique by which the progress of the client can be monitored. When the client is able to reproduce the target trajectory of a speech sound reliably, it is a good indication that their pronunciation is correct. Figure 7.9 illustrates the results of a short therapy session with the Visual Ear. Target trajectories for the word 'whey' are displayed together with the trajectories produced by a partially deafened speaker with a noticeable pronunciation difficulty.

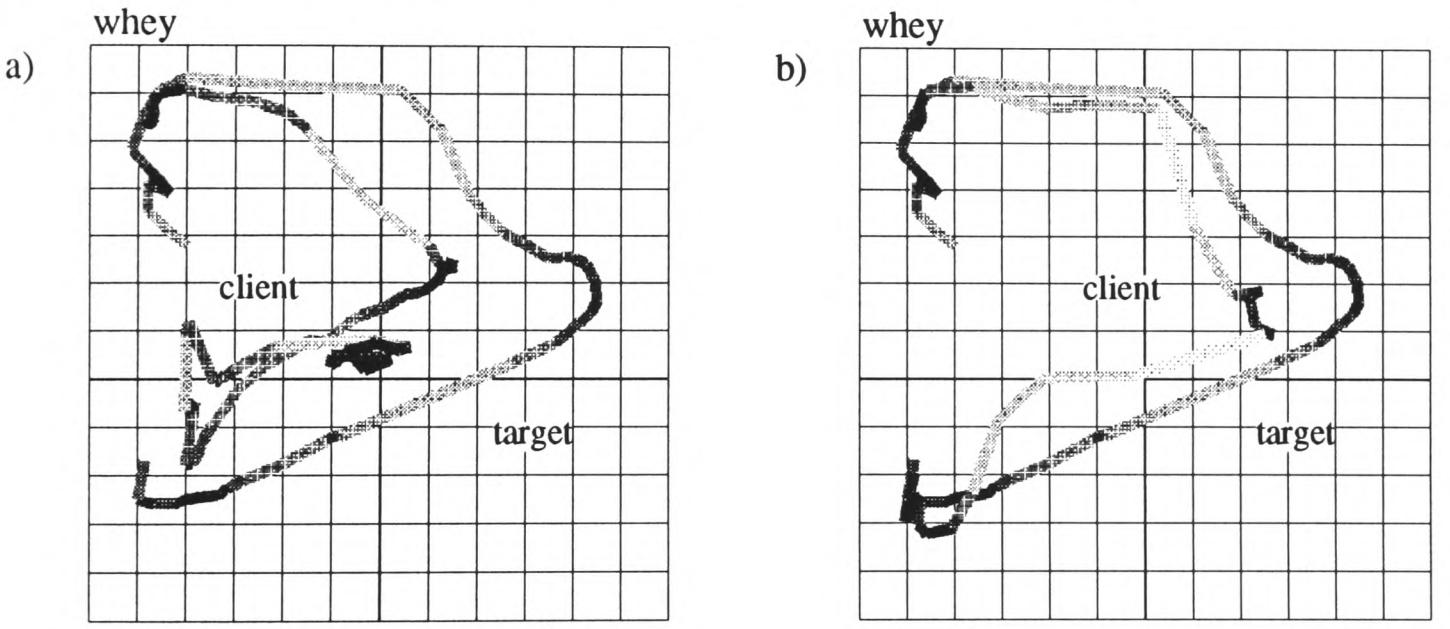


Figure 7.9: *Results of a short therapy session with the Visual Ear. The client attempts to reproduce a target trajectory for the word 'whey' after a) no therapy, and b) a short period of therapy with the Visual Ear.*

The extent to which the client and target trajectories correspond can be quantified by performing DTW on the sequences of (x,y) coordinates from which the trajectories are composed (see Section 6.4.1). The distance score obtained refers to the geometrical distance

[6]Assuming the computing system supports a colour display.

in the map, not their Euclidean distance in feature space. This is desirable since the former corresponds to the difference between the trajectories observed by the viewer. During the course of a therapy session the distance score provides a quantitative assessment of the improvement in pronunciation of the client.

7.5.2 Self-learning Pronunciation

An attraction of the Visual Ear is its suitability for *self-learning* pronunciation. Clients who possess some linguistic knowledge and command of pronunciation (for instance the post-lingually deafened) can practise pronunciation by attempting to reproduce a variety of target trajectories which are derived from a large 'on-line' dictionary of speech sounds assembled by a normal speaker. With self-experimentation the client can explore the different regions of the feature map with their voice until they are able to reproduce arbitrarily shaped target trajectories at will. The online dictionary could be divided into lessons to encourage a structured approach to learning pronunciation.

7.5.3 Foreign Language Pronunciation

The Visual Ear has so far been considered for teaching pronunciation to clients with a communication disorder. However, the Visual Ear is potentially useful to anyone wanting to improve their pronunciation. An obvious application is self-learning foreign language pronunciation using a feature map trained on a corpus of speech of the desired language. Since the majority of clients using the system would have normal hearing, it would be advantageous to accompany trajectories by audible versions of the sounds they represent. This system would then match the performance of conventional methods such as learning by audio cassette, with the added benefit of instant visual assessment.

7.6 Summary of Application

Ball [1] listed several clinical requirements of a computer-based system for visual feedback therapy, a summary of which is presented below:

- A simple visual representation of the acoustically complex speech signal that is clear, unequivocal, and intuitive (i.e. conforming to normal human expectations of how the speech signal behaves).
- Reliable, simple-to-use software that is flexible enough to display arbitrary features of speech (or combinations of features) in real-time.
- The simultaneous display of the client's speech with target speech from a trained speech therapist.
- The ability to determine what is wrong with the client's speech.

The Visual Ear meets the above requirements and demonstrates several further attractive properties. Not least, the Visual Ear is *non-invasive* (information enters the system through a microphone), and unlike EPG it is possible to investigate all phonemes. The trajectory representation is undoubtedly a very clear method for displaying speech sounds. It is also an intuitive representation (the trajectories *look* like they have been derived from speech) because the rate at which speech is uttered into the system is mirrored by the rate at which the trajectory is produced. Although novel in design, the Visual Ear is implemented using several reliable computational techniques:

- Cepstral analysis is a well-established speech pre-processing technique.
- The feature map algorithm is similar in operation to traditional clustering algorithms.

- Computing trajectory path with the centre of mass method is an elementary operation.

Trajectories can be displayed in real-time on the present computing system. However, it would be necessary to adopt special purpose hardware if the the host computer was based on, say, a typical IBM PC AT[7]. In this case, the computer would be used purely for display purposes.

The simultaneous display of client and target trajectories provides an instant method for assessment of the client's speech. It is anticipated that therapy with the Visual Ear will encourage concentration in the client due to the positive feedback derived when it is attempted to reproduce target trajectories with one's own voice. In particular, younger clients may find this a rewarding method of learning, especially if colour is used extensively in the display. By means of an online dictionary of target speech sounds, the Visual Ear is suitable as a self-learning system for clients who possess basic linguistic and verbal abilities. A separate system could be designed for any language, conditional on there being an extensive corpus of speech sounds available for training the feature map. Any type of speech may be represented as a trajectory, providing flexibility in the design of therapy procedures.

The Visual Ear is *not* able to diagnose precise articulation defects, although such defects are highlighted by differences between client and target trajectories, and can therefore be investigated using alternative therapy techniques. Neither is the Visual Ear a system for quantitative analysis of speech. However, information such as pitch, short-term energy, degree of voicing, spectral balance, etc, can also be obtained during the derivation of cepstral coefficients, and could be displayed simultaneously with the trajectories.

In summary, the Visual Ear constitutes a new method for visual feedback therapy – a method in which human speech is graphically represented as a moving trajectory on a

[7]Note that this is the case with commercial systems for EPG and ELG.

computer screen. Although the Visual Ear is primarily designed for assisting pronunciation in the speech and hearing impaired, it may also find application in the learning of foreign language pronunciation. As with EPG and ELG, the Visual Ear does not constitute a whole environment for speech therapy by itself. However, the ease of use and intuitive nature of the visual display may allow it to play an important role in combination with other methods.

Chapter 8

Conclusions and Future Research

8.1 Conclusions

The objectives laid out in Chapter 1 were, respectively:

- To develop an effective method for optimising the RBF classifier on a difficult speaker-independent spoken letter recognition problem.
- To assess the performance of the RBF classifier relative to other classifiers such as the OLT, KNN, and MLP.
- To investigate what advantages may be obtained from dynamic classification techniques, within the context of the same speech recognition problem.

8.1.1 Consideration of Objectives

The following strategy was evolved to optimise the RBF for use on large speaker-independent speech recognition problems. It reflects a compromise between heuristic choice of network parameters and a fully adaptive approach:

- *Employ a maximum of $\frac{M}{2}$ kernel functions, the optimal number depending on the complexity of the problem (where M, as before, is the number of training patterns).*
- *Use the adaptive K-means algorithm for allocating centres (especially if the number of kernel functions is small).*

- *Employ the locality index method for optimising the kernel function widths.*

It was found that spatially ordered centres (as obtained with the feature map algorithm) were able to provide an equivalent level of generalisation to that of the adaptive K-means algorithm, demonstrating that the constraint of spatial ordering does not affect the development of the centres in input space. It was also found that generalisation was sensitive to the locality of representation of input patterns in the space of the kernel function outputs, the optimum locality of representation differing considerably from that determined by heuristic methods.

The lowest error rate for Problem A1 was obtained with an optimised RBF containing 1024 centres[1]. At 10.4% this error rate is close to the theoretical minimum error rate of 8.4% of an optimal Bayes classifier (see Section 5.2). This is an indication of both the effectiveness of the RBF optimisation strategy outlined above, and the complexity of Problem A1. Both the neural network classifiers achieved significantly lower error rates than the traditional classifiers, illustrating the trade-off between computational investment in training and degree of generalisation.

In the case of template matching procedures for speech recognition there is clearly an advantage in dynamic classification techniques. However, neural network classifiers overcome the constraint of linear time normalisation by developing a sophisticated implicit model of temporal variation from large numbers of training patterns. It was demonstrated that the RBF can achieve a lower error rate than DTW on Problem A1, but that the reverse is true on Problem D – a problem for which considerably fewer training patterns were available.

An examination of the SPLIT method revealed that the reduction in computation rel-

[1] In this problem it was not possible to investigate an RBF with $\frac{M}{2}$ centres due to limited availability of memory. However, it is unlikely that the error rate would be significantly reduced as their is little difference in error rate between an RBF with $\frac{M}{8}$, and $\frac{M}{4}$ centres respectively.

ative to DTW was not as significant as previously claimed. This method was consequently abandoned in favour of DTW for small-scale problems and in favour of the RBF for speaker-independent problems. Experiments incorporating the two-dimensional feature map algorithm into the SPLIT method confirmed that speech sounds could be represented as trajectories in the feature map, yielding an intuitive method by which humans can visualise speech. Several techniques were established for enhancing the clarity and information content of the trajectories, culminating in the development of a novel system for visual feedback therapy known as the Visual Ear.

8.1.2 General Comments

The fact that the RBF achieved the lowest error rate of the static classifiers examined in this thesis, and exceeded the performance of a well established dynamic classifier, is of importance to the design of systems for isolated word recognition. If such systems are to be applied to real world problems (e.g. systems for accessing information over the telephone, or for control of domestic appliances) they must be capable of speaker-independent operation. In this respect the results for the RBF are encouraging: a 10% error rate for spoken letter recognition problem should translate into lower error rates for less confusing vocabularies, as testified by the 3% error rate of Problem C. However, this level of error rate is still above that which would be required for a marketable speech recognition device (e.g. less than 1% might be acceptable for a telephone-based directory enquiries system in which the name and address of the desired telephone holder was communicated letter by letter).

The conception of the Visual Ear was prompted by the incorporation of the feature map algorithm into the SPLIT method. This shifted the purpose of the system from *classification* of speech sounds into *representation* of speech sounds, with the result that the goal of

classification was dropped altogether. It is therefore important not to view neural networks purely in terms of pattern classification; the self-learnt representations of neural networks may be desirable in themselves.

It was noted at the outset of this thesis that many of the computational techniques applied in pattern recognition were first developed in the 1960's. Only recently however, in the guise of neural networks, have these techniques been successfully applied to real world problems. It may be concluded therefore, that the recent interest in neural networks is well founded in the sense of furthering the goals of pattern recognition, but that the original biological inspiration of neural networks is now of little relevance to the solution of real-world engineering problems.

Although the performance of the RBF and MLP classifiers was almost equivalent when applied to Problem A1, the RBF was judged to be superior due to the simplicity with which its operation may be interpreted; the first layer of the RBF projects the input patterns into a space in which the linear separability of the pattern classes is increased, and the second layer performs a linear classification of the projected input patterns into the pattern classes. This interpretation arises from the fact that the two layers of the RBF are trained *independently.* In the case of the MLP the two (or more) layers are trained in a single procedure, and it is difficult to interpret the functions of the weights in the different layers as a result.

8.2 Future Research

It would be profitable to concentrate future research efforts in the two areas discussed below.

8.2.1 Automatic Feature Selection in the RBF

The kernel function used in this thesis was the radially symmetric Gaussian function, in which equal weight is given to each input component during the computation of kernel func-

tion response. Although this did not prevent the RBF from achieving excellent results on the problems examined, further improvement may be possible if the components of the input patterns were weighted according to their relative merit as features for the disambiguation of the pattern classes. Although the weightings could be derived by canonical analysis of the training patterns, it would be preferable to incorporate feature selection directly into the training algorithm of the RBF. Automatically weighted input components would be particularly useful in cases where the components of the input patterns are obtained from different sources (data fusion problems), such as short-term energy, zero-crossing rate, degree of voicing, as well as cepstral coefficients. Lowe [36] has already presented a method for automatic feature selection, although it was rejected in this instance due to excessive training time, and reliance on the squared output error as the criterion for adjustment of network parameters. A better optimisation criterion is the error rate recorded on the training set, or a cross-validation set of input patterns. The problem of unconstrained adaptation in Lowe's method could also be eliminated if centres were allocated with the adaptive K-means algorithm, thereby restricting the adaptation procedure to the kernel function widths. It would be interesting to compare the error rate obtained with Lowe's fully adaptive optimisation method with that of both the optimisation strategy employed in this thesis, and the new method outlined above. However, for training times to be manageable the comparison would have to be conducted on a small-scale problem initially.

8.2.2 Systematic Evaluation of the Visual Ear

The Visual Ear has not yet been evaluated by expert speech therapists in the context of their work. This must be achieved before its effectiveness as a method for visual feedback therapy can be determined. Such an evaluation requires access to an extensive corpus

of training data (see Section 7.5 for a specification). Ideally the speech sounds in the corpus should be labelled phonetically to enable an automatic calibration procedure to be applied to the feature map. A possible corpus is the SCRIBE database, presently under compilation at the Royal Signals and Radar Establishment, Malvern, the Centre for Speech Technology Research, Edinburgh, and University College, London. The evaluation should provide insight into the following areas:

- The usefulness of intensity coding as a means of providing additional information to the viewer.
- The value of feature maps developed with feature vectors derived from different pre-processing techniques.
- The consistency of the shape of trajectories representing plosive phonemes.

The final point exposes a deficit in the design of the Visual Ear; plosives are not represented as stable shaped trajectories. A possible method for stabilising plosive trajectories is to train the feature map on multiple feature vectors formed by the concatenation of several successive feature vectors, smoothing the instability resulting from periods of rapid spectral change. Alternatively, a variable smoothness parameter may be employed to increase smoothing in regions of the map representing plosive phonemes.

Appendix A

Non-optimality of Minimum Squared Error Solution

Upon presentation of the mth training pattern to the inputs of a single linear discriminant function, the output is given by:

$$o^m = \sum_{j=1}^{N_I} x_j a_j^m + x_0 \tag{A.1}$$

where a_j^m is the jth component of the training pattern, and x_j is jth weight connected to it. Training the discriminant function minimises the squared error between this output and the corresponding target output, b^m, over the M patterns in the training set:

$$e = \sum_{m=1}^{M} (b^m - o^m)^2 \tag{A.2}$$

The purpose of this appendix is to show that the solution obtained using the minimum squared error criterion is not necessarily coincident with the solution yielding maximum generalisation. Figure A.1 shows the two-dimensional training patterns of a simple linearly separable two class problem, which may be solved with a single discriminant function. If the two classes were represented by target outputs of 0, or 1, during training, test patterns will be classified with the following rule:

$$\begin{aligned} o^{test} &\leq \frac{1}{2} \qquad class\ 0 \\ &> \frac{1}{2} \qquad class\ 1 \end{aligned} \tag{A.3}$$

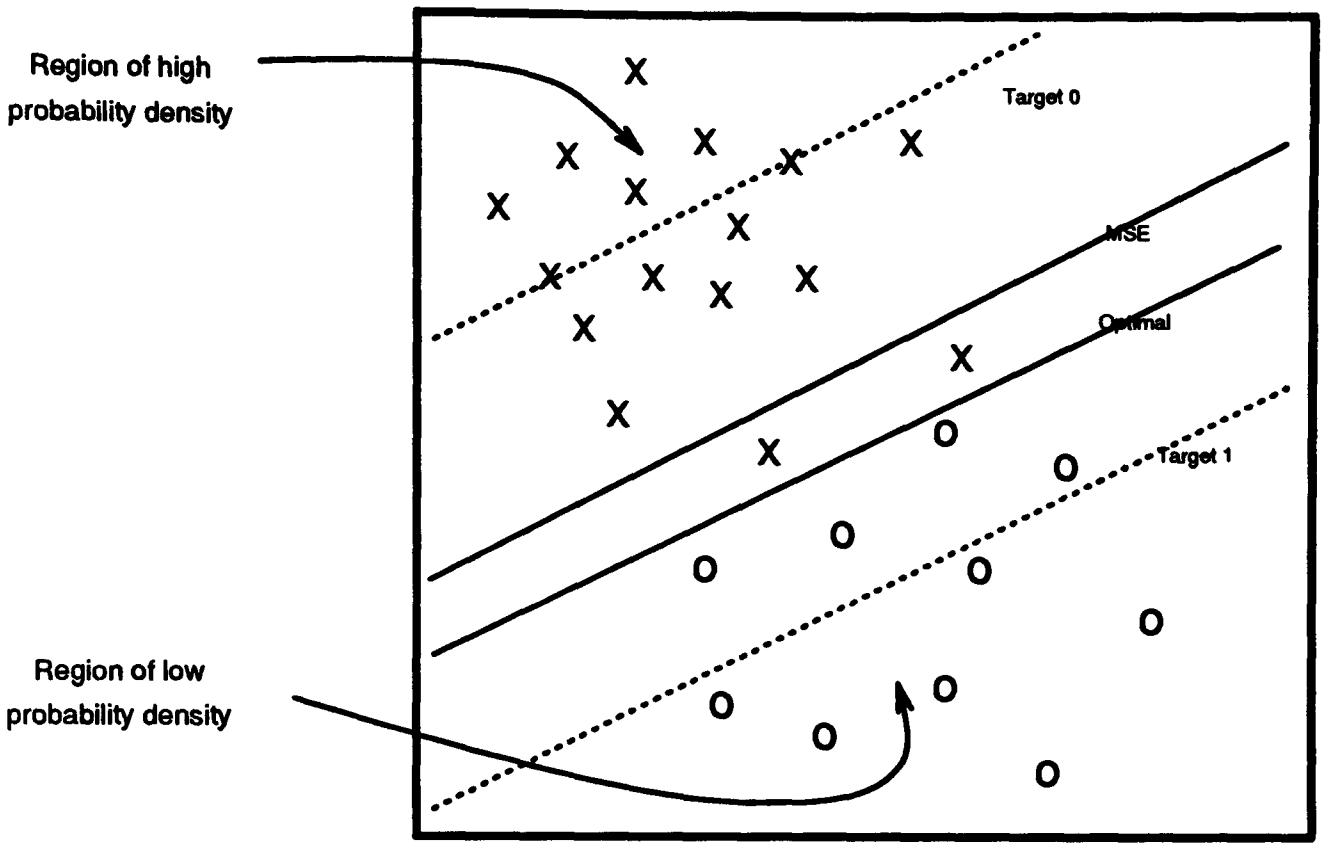


Figure A.1: *Non-optimality of decision boundary resulting from minimum squared error fit to the training patterns. Note that the target lines pass through the regions of high probability density of training patterns, which therefore dominate the positioning of the decision boundary. The position of the decision boundary resulting in optimal generalisation is shown for comparison.*

Substituting Equation A.1 into this rule yields the formula for the decision boundary (see Figure A.1):

$$a_1 x_1 + a_2 x_2 + a_0 = \frac{1}{2} \tag{A.4}$$

Formulas for lines representing the target outputs can be obtained in a similar manner (see Figure A.1):

$$a_1 x_1 + a_2 x_2 + a_0 = 0,\ 1 \tag{A.5}$$

Evidently, the decision boundary is placed *midway* between the target lines in this simple two class problem. The distances between the training patterns and the target lines represent the *errors* appearing in Equation A.2. The target lines are therefore positioned so as to minimise these errors, affecting the position of the decision boundary in the process. Regions of input space with a high probability density of training patterns dominate the positioning of the decision boundary, since they contribute the greatest proportion of the total error. In contrast, outlying training patterns influence the positioning of the decision boundary to a much lesser extent, although attention to these training patterns is more likely improve generalisation. The positioning of the decision boundary will therefore favour those classes with a high probability density of training patterns, and is therefore unlikely to provide optimal generalisation.

Appendix B

Theory Behind Cepstral Analysis

In the excitation-modulation model of speech, the spectrum of the speech signal, $X(f)$, is related to the excitation spectrum, $G(f)$, and spectrum envelope, $H(f)$, by:

$$X(f) = G(f)H(f) \tag{B.1}$$

Taking logarithms of the magnitudes of these spectra we get:

$$\log|X(f)| = \log|G(f)| + \log|H(f)| \tag{B.2}$$

which informs us that the log magnitude spectrum is composed of two superimposed components. These can be clearly seen in Figures B.1a and b, which show the log magnitude spectra of a typical voiced and unvoiced sound respectively. In the case of the voiced sound, there is a rapidly varying periodic component due to the periodic nature of the excitation, and a slowly varying component due to the spectrum envelope. This component is also apparent for the unvoiced sound, but the rapidly varying component is now random due to the turbulent (aperiodic) nature of the excitation. The spectrum envelope is very useful in discriminating speech sounds since it is determined by the positions of the organs used during speech articulation. The excitation spectrum carries information concerning the pitch, and degree of voicing of the speech waveform. This is considered subsidiary to the human speech recognition process, since monotonic and whispered speech can be recognised easily under normal listening conditions (the pitch, or intonation profile of speech,

does play a crucial *semantic* role in the English language however). Cepstral analysis is a way of separating the two components by computing the spectrum of the log magnitude spectrum, termed the *cepstrum.* Figures B.2a and b show the cepstrum of a typical voiced and unvoiced sound respectively. In both cases the lower coefficients represent the spectrum envelope since it forms the lowest frequencies in the log magnitude spectrum. The higher coefficients represent the excitation spectrum. The cepstrum of the voiced sound contains small peaks corresponding to its excitation harmonics (fundamental, second harmonic, etc). These are not detected in the cepstrum of the unvoiced sound because its excitation source is not periodic. The majority of the variance in the cepstrum occurs in the first eight or so coefficients representing the spectrum envelope. It is these coefficients which are extracted for the purposes of recognition.

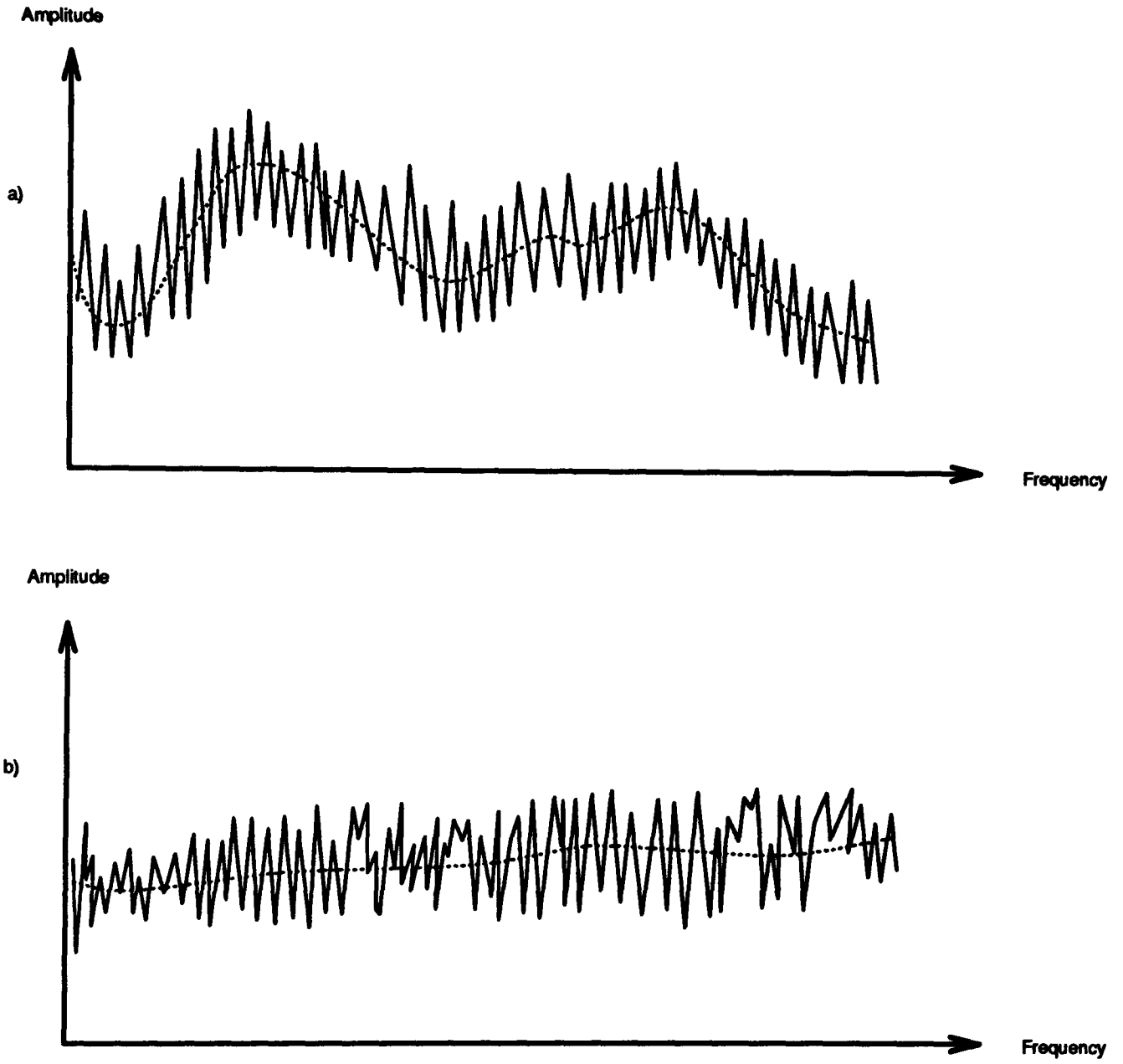


Figure B.1: *Log magnitude spectra of typical voiced and unvoiced sounds. a) Voiced sound. Note the periodic nature of the spectrum peaks, corresponding to harmonics of the fundamental frequency. b) Unvoiced sound. Note that the spectrum is much flatter, with small peaks occurring randomly due to the turbulent source of excitation.*

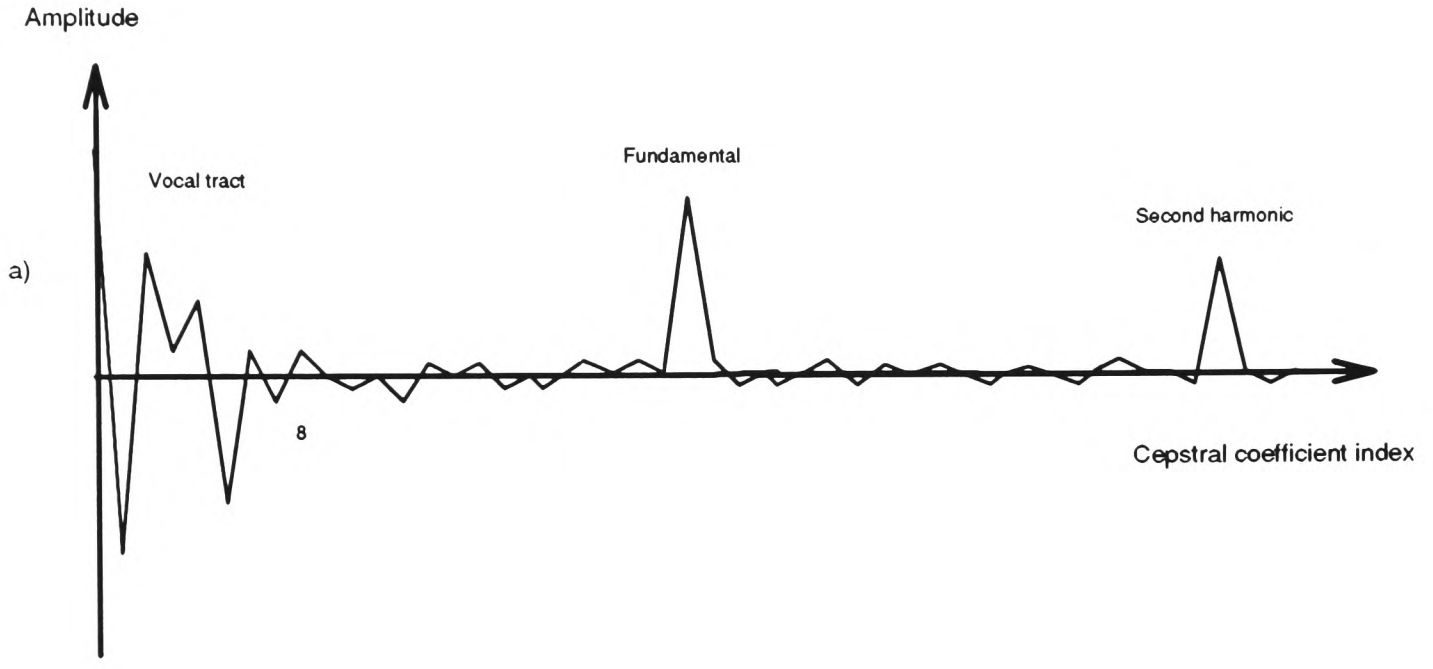


Figure B.2: *Cepstrum of typical voiced and unvoiced sounds. The low coefficients represent the spectrum envelope, and higher ones the source of excitation. a) Voiced sound. b) Unvoiced sound.*

Appendix C

The F Ratio

For a simple problem containing two classes and one-dimensional input patterns, Fisher's discriminant function, or F ratio, is given by:

$$F = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{C.1}$$

where μ_1 and μ_2 are the means of the two classes over the training set, and σ_1^2 and σ_2^2 are their variances. The F ratio may be extended to an N_C class problem with arbitrary dimension input patterns:

$$F = \frac{\sum_{i=1}^{N_C} ||\mu_i - \mu||^2}{\sum_{i=1}^{N_C} \sigma_i^2} \tag{C.2}$$

where μ_i is the mean of the ith class over the training set, σ_i^2 is its variance, and μ is the overall mean input pattern. If $\mathbf{a}_i^m$ is the mth of M_i training pattern belonging to the ith class, these terms may be defined as follows:

$$\mu_i = \frac{1}{M_i} \sum_{m=0}^{M_i} \mathbf{a}_i^m \tag{C.3}$$

$$\sigma_i^2 = \frac{1}{M_i} \sum_{m=0}^{M_i} ||\mathbf{a}_i^m - \mu_i||^2 \tag{C.4}$$

$$\mu = \frac{1}{\sum_{i=0}^{N_C} M_i} \sum_{i=0}^{N_C} \tag{C.5}$$

$$sum_{m=0}^{M_i} \mathbf{a}_i^m \tag{C.6}$$

(C.7)

F can be expressed more succintly in terms of the *between class* scatter matrix, B, and the *within class* scatter matrix, W, of the input patterns [42]:

$$F = \frac{tr(B)}{tr(W)} \tag{C.8}$$

where the matrices B and W are given by:

$$B = \frac{1}{N_C} \sum_{i=1}^{N_C} (\mu_i - \mu)(\mu_i - \mu)^T \tag{C.9}$$

$$W = \frac{1}{N_C} \sum_{i=1}^{N_C} \frac{1}{M_i} \sum_{m=0}^{M_i} (\mathbf{a}_i^m - \mu_i)(\mathbf{a}_i^m - \mu_i)^T \tag{C.10}$$

The definitions in Equations C.2 and C.8 are entirely equivalent.

Appendix D

One Hundred Word Database

The author made recordings of single utterances of the following list of one hundred words. The words were selected randomly from *A Teacher's Book of 30,000 Words* [56], with the condition that each contained just four letters.

glad	want	ogle	wear	mush	easy	mean	wall	bask	free
bull	puma	pave	huge	hurl	much	fern	loss	moot	thaw
note	sand	scow	gush	shoe	espy	reek	buff	melt	pose
gall	duel	jilt	scan	chef	post	roam	weld	maul	vial
isle	rice	mast	spry	feat	crab	tilt	comb	node	dumb
soak	acme	most	wilt	rust	heat	skin	jeer	seem	gave
mole	mere	size	wife	frog	lief	oath	rack	lacy	heir
same	helm	edge	soda	sire	hind	word	flax	cone	wren
fete	bulk	slip	duty	goes	grip	crop	clog	wend	stir
shun	volt	grin	neat	none	rail	akin	dame	boot	amen

Bibliography

[1] V. Ball. Computer-based tools for assessment and remediation of speech. *British Journal of Disorders of Communication*, Vol.26:95–113, 1991.

[2] E. Baum and D. Haussler. What size net gives valid generalisation ? *Neural Computation*, 1(1):151–160, 1989.

[3] W. Bialek et al. Reading a neural code. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 36–43, 1990.

[4] P. Brauer and P. Knagenhjelm. Infrastructure in Kohonen maps. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 647–650, 1989.

[5] J. Bridle et al. Comparison of neural and conventional classifiers on a speech recognition problem. In *Proc. 1st IEE International Conference on Artificial Neural Networks*, pages 86–89, 1989.

[6] D. Broomhead et al. A parallel architecture for non-linear adaptive filtering and pattern recognition. In *Proc. 1st IEE International Conference on Artificial Neural Networks*, pages 265–69, 1989.

[7] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems 2*, pages 321–355, 1988.

[8] D. Broomhead, D. Lowe, and A. Webb. A sum rule satisfied by optimised feed-forward layered networks. Technical Report Memorandum 4341, Royal Signals and Radar Establishment, St. Andrews Rd, Malvern, England, 1989.

[9] S. Chen et al. Orthogonal least squares learning algorithm for radial basis functions. *IEEE Transactions on Neural Networks*, Vol.2, No.2:302–309, 1991.

[10] R. Cole and J. Jakimik. A model of speech perception. In R. Cole, editor, *Perception and Production of Fluent Speech*. Erlbaum, Hillsdale, 1980.

[11] T.M. Cover. Geometric and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, Vol.14:326–334, 1965.

[12] S.J. Cox. Hidden Markov models for automatic speech recognition: Theory and application. In C. Wheddon and R. Linggard, editors, *Speech and Language Processing.* Chapman and Hall, 1990.

[13] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4), 1989.

[14] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol.28, No.4:357–366, 1980.

[15] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[16] S. Furui. A VQ-based preprocessor using cepstral dynamic features for speaker-independent large vocabulary word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol.36, No.2:980–987, 1988.

[17] R. Gorman and T. Sejnowski. Analysis of hidden units in a layered network trainied to classify sonar targets. *Neural Networks*, Vol.1:75–89, 1988.

[18] R.M. Gray. Vector quantization. *IEEE ASSP Magazine*, Vol.1, No.2:4–29, 1984.

[19] M. Gutierrez, Wang. J., and R.O. Grondin. Estimating hidden units for two-layer perceptrons. In *Proc. 1st IEE International Conference on Artificial Neural Networks*, pages 120–124, 1989.

[20] W.J. Hardcastle, F.E. Gibbon, and W. Jones. Visual display of tongue palate contact: Electropalatography in the assessment and remediation of speech disorders. *British Journal of Disorders of Communication*, Vol.26:41–74, 1991.

[21] J. Holmes. *Speech Synthesis and Recognition.* Van Nostrand Reinhold, UK, 1988.

[22] M. Hunt and C. Lefebvre. A comparison of several acoustic representations for speech recognition with degraded and undegraded speech. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 262–265, 1989.

[23] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions Acoustics, Speech, and Signal Processing*, Vol.23, No.1:67–72, Feburary 1975.

[24] M. James. *Classification Algorithms.* Collins, London, 1985.

[25] P. Kanerva. *Sparse Distributed Memory.* MIT Press, Cambridge, 1988.

[26] T. Kohonen. Dynamically expanding context, with application to the correction of symbol strings in the recognition of continuous speech. In *Proc. 8th International Conference on Pattern Recognition, Paris*, pages 1148–1151, 1986.

[27] T. Kohonen. The neural phonetic typewriter. *Computer*, pages 11–22, March 1988.

[28] T. Kohonen. *Self-Organisation and Associative Memory.* Springer-Verlag, Berlin, third edition, 1988.

[29] T. Kohonen. The self-organizing map. *Proc. IEEE*, Vol.78, No.9:1464–1480, 1988.

[30] Y. Le Cun, J.S. Denker, and S.A. Solla. Optimal brain damage. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605, 1990.

[31] Y. Lee and R. Lippmann. Practical characteristics of neural network and conventional pattern classifiers on artificial and speech problems. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 168–177, 1990.

[32] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, Vol.28:84–95, 1980.

[33] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, Vol.4, No.2:4–22, 1987.

[34] R. Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, pages 47–64, November 1989.

[35] R. Lippmann. Review of neural networks for speech recognition. *Neural Computation 1*, pages 1–38, 1989.

[36] D. Lowe. Adaptive radial basis function nonlinearities and the problem of generalisation. In *Proc. 1st IEE International Conference on Artificial Neural Networks*, pages 171–175, 1989.

[37] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics 5*, pages 115–133, 1943.

[38] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, 1969.

[39] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation 1*, pages 281–294, 1989.

[40] S. Nowlan. Max likelihood competition in RBF networks. Technical Report CRG-TR-90-2, University of Toronto Connectionist Research Group, 10 Kings College Road, Toronto, M5S 1A4, Canada, 1990.

[41] Numerical Algorithms Group Ltd, NAG Ltd, Wilkinson House, Jordan Hill Road, Oxford OX2 8DR. *NAG Fortran Library Introductory Guide*, mark 14 edition, 1990.

[42] T. Parsons. *Voice and Speech Processing.* McGraw-Hill, 1986.

[43] M. Powell. Radial basis functions for multivariable interpolation: A review. In *IMA conference on 'Algorithms for the Approximation of Functions and Data', RMCS Shrivenham*, 1985.

[44] G. Quenot et al. A dynamic programming processor for speech recognition. *IEEE Journal of Solid State Circuits*, Vol.24(2):349–357, 1989.

[45] L. Rabiner et al. On the application of vector quantisation and hidden Markov models to speaker-independent, isolated word recognition. *BSTJ*, Vol.62, No.4:1075–1105, 1983.

[46] S. Renals and R. Rohwer. Phoneme classification experiments using radial basis functions. In *Proc. International Joint Conference on Neural Networks*, pages I.461–I.467, 1989.

[47] F. Rosenblatt. *Two theorums of statistical separability in the perceptron*, pages 421–456. HM Stationary Office, London, 1959.

[48] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York, 1962.

[49] D. Rumelhart and J. (eds) McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of cognition.* MIT Press, Cambridge, 1986.

[50] H. Sakoe and S. Chiba. A dynamic programming approach to continuous speech recognition. In *Proc. International Congress on Acoustics, Budapest, Hungary*, pages Rep. 20–C–13, 1971.

[51] T. Sejnowski and C. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, Vol.1:145–168, 1987.

[52] J.A. Simmons. Acoustic-imaging computations by echolocating bats: Unification of diversely-represented stimulus features into whole images. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 2–9, 1990.

[53] N. Sugamura, K. Shikano, and S. Furui. Isolated word recognition with phoneme-like templates. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 723–726, 1983.

[54] L. Tarassenko, J. Tombs, and J. Reynolds. Neural network architectures for content-addressable memory. *Proc. IEE, Part F*, Vol.138, No.1:33–39, 1991.

[55] G. Tattersall, P. Linford, and R. Linggard. Neural arrays for speech recognition. *British Telecom Tecnology Journal*, Vol.6, No.2:140–163, 1988.

[56] E.L. Thorndike and I. Lorge. *The Teacher's Book of 30,000 Words.* Publisher Unknown, New York: Teachers College, Columbia University, 1944.

[57] V.M. Velichiko and N.G. Zagoruiko. Automatic recognition of 200 words. *Int. Journal of Man-Machine Studies*, pages 223–234, 1970.

[58] E.A. Wan. Neural network classification: A Bayesian interpretation. *IEEE Transactions on Neural Networks*, Vol.1, No.4:303–305, 1990.

[59] J. Wilpon et al. An improved word detection algorithm for telephone quality speech incorporating both syntactic and semantic constraints. *AT and T Bell Labs Technical Journal*, Vol.63, No.3:479–498, 1984.

[60] D. Wolpert. A benchmark for how well neural nets generalize. *Biological Cybernetics*, Vol.61:303–313, 1989.

[61] P. Woodland. Isolated word speech recognition based on connectionist techniques. *British Telecom Technology Journal*, Vol.8, No.2:61–66, 1990.