Speech and Neural Network Dynamics

Stephen John Renals

Doctor of Philosophy

University of Edinburgh

1990

Abstract

This thesis is concerned with two principal issues. Firstly the radial basis functions (RBF) network is introduced and its properties related to other statistical and neural network classifiers. Results from a series of speech recognition experiments, using this network architecture, are reported. These experiments included a continuous speech recognition task with a 571 word lexicon. Secondly, a study of the dynamics of a simple recurrent network model is presented. This study was performed numerically, via a survey of network power spectra and a detailed investigation of the dynamics displayed by a particular network.

Word and sentence recognition errors are reported for a continuous speech recognition system using RBF network phoneme modelling with Viterbi smoothing, using either a restricted grammar or no grammar whatsoever. In a cytopathology task domain the best RBF/Viterbi system produced first choice word errors of 6% and sentence errors of 14%, using a grammar of perplexity 6. This compares with word errors of 4% and sentence errors of 8% using the best CSTR hidden Markov model configuration. RBF networks were also used for a static vowel labelling task using hand-segmented vowels excised from continuous speech. Results were not worse than those obtained using statistical classifiers.

The second part of this thesis is a computational study of the dynamics of a recurrent neural network model. Two investigations were undertaken. Firstly, a survey of network power spectra was used to map out the temporal activity of this network model (within a four dimensional parameter space) via summary statistics of the network power spectra. Secondly, the dynamics of a particular network were investigated. The dynamics were analysed using bifurcation diagrams, power spectra, the computation of Liapunov exponents and fractal dimensions and the plotting of 2-dimensional attractor projections. Complex dynamical behaviour was observed including Hopf bifurcations, the Ruelle-Takens-Newhouse route to chaos with mode-locking at rational winding numbers, the period-doubling route to chaos and the presence of multiple coexisting attractors.

Declaration

This thesis has been composed by myself and contains original work of my own execution. Some of the work reported in this thesis has previously been published:

S. Renals. Radial basis functions network for speech pattern classification. *Electronics Letters*, 25:437–439, 1989.

Steve Renals and Richard Rohwer. A study of network dynamics. *Journal of Statistical Physics*, 58:825–847, 1990.

Steve Renals and Richard Rohwer. Learning phoneme recognition using neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 413–416, Glasgow, 1989.

Steve Renals and Richard Rohwer. Phoneme classification experiments using radial basis functions. In *Proceedings of International Joint Conference on Neural Networks*, volume I, pages 461–468, Washington DC, 1989.

Steve Renals and Richard Rohwer. Neural networks for speech pattern classification. In *IEE Conference Publication 313, Artificial Neural Networks*, pages 292–296, London, 1989.

Steve Renals. Chaos in neural networks. In L. B. Almeida and C. J. Wellekens, editors, *Neural Networks*, volume 412 of *Lecture Notes in Computer Science*, pages 90–99, Springer-Verlag, Berlin, 1990.

Acknowledgements

My principal thanks must go to Richard Rohwer, my advisor, for his patience, encouragement and unflagging support. His willingness to listen to my latest ideas and to distill the essence from them has been of great importance throughout this work. Working with Richard has taught me much about what it is to be a scientist. My thanks also go to David Wallace, my other advisor, whose support has been invaluable these past three years.

I performed most of this work at the Edinburgh University Centre for Speech Technology Research, where I have been made a welcome guest for the duration of my PhD studies. Without their willingness to allow me to use up disk space and CPU cycles and the provision of ready access to their speech databases most of this work couldn't have been done. Particular thinks must go to Yasuo Ariki, Art Blokland, Alan Crowe, Jonathan Dalby, Greg Filz, Fergus McInnes, David McKelvie and Andrew Sutherland who shared their insights, results and programs with me. Tim Bradshaw performed wonders with LaTeX and created this thesis style. Steve Isard proofread this thesis and made several valuable comments. Tim, Andie Ness, Alex Zbyslaw, Bob Anstruther and Euan Purves all provided computer support.

This work has been guided by discussions that I've had with several people. David Broomhead and Sara Solla were particularly helpful when I first started on the work that was to become chapters 5 and 6. The Connectionist Workshop at Edinburgh has always been source of many ideas and fertile discussion — Peter Dayan, David Willshaw, Richard Rohwer, David Wallace, Marcus Frean, Jay Buckingham, Steve Finch and Martin Simmen have all been valuable participants in this group. And the Shotgun Reading Group added much excitement to Monday mornings in the spring of 1990!

I'd like to thank my parents for encouraging me to do what I wanted to do and for always being supportive. There are many people who have helped to make my time spent in Edinburgh such a good experience: Tom provided somewhere to live, couscous and companionship; Mike provided cheeseburgers; Toby provided sartorial elegance; Nature's Gate provided tofu and soya dessert; Jonathan Richman uplifted my spirits in the days when I was writing this; and Jane was here at the beginning, returned near the end and was always an inspiration.

Contents

List of Figures

List of Tables

1

INTRODUCTION

The original motivation for the study of neural networks was to develop models of biological neural systems that displayed similar behaviour to that observed in neurophysiological and psychological experiments (e.g., see papers collected in [184]). A principal goal of this research programme was to develop models of memory, learning and adaptation. Many of the neural network models that have been developed bear only the scantiest resemblance to biological systems. However they have proved to be extremely interesting systems in their own right. The study of these systems has focussed on their intrinsic physical properties (dynamics and statistical mechanics) and on their ability to learn from examples and to generalise to new data. These two threads of research are closely intertwined: the development of learning algorithms and the design of networks that generalise adequately is considerably facilitated by an understanding of the physical properties of the neural network system under investigation.

The neural network models studied in this thesis consist of networks of simple processing units (nodes) coupled by weighted connections. Each node receives an input vector, processes it using some predefined transfer function and produces a scalar output. Depending on the form of the weight matrix, a network model may be classed as *static* or *dynamic*. Static networks may be defined as those which are bound to converge to a fixed point attractor. That is, after a period of transient dynamical behaviour, the outputs of the network reach an equilibrium point, with the

outputs of each node in the network assuming a stable value. Transitions between fixed point attractors may be achieved by external inputs to the network. Dynamic networks, however, are not guaranteed to converge to a fixed point attractor: they may converge to a fixed point, limit cycle or chaotic attractor, depending on the exact values of each weighted connection, the initial states of each node and any external input.

Examples of static network models include all feed-forward networks (there can be no dynamical activity without feedback), and those recurrent networks for which a Liapunov function[1] may be defined. Examples of static, recurrent network models include asynchronously updated symmetric networks, with sum and threshold units [79] or similar networks with analog nodes (where a sigmoid or soft threshold function replaces the hard threshold) [35].

Static networks are more amenable to theoretical analysis compared with their dynamic counterparts; however dynamic networks are computationally more powerful. This computational power is useless unless learning algorithms are available to train a network to perform a particular computation. Such algorithms have been devised for static networks (e.g. [175, 1, 95]); learning algorithms for dynamic networks are at a more primitive state.

1.1 Learning

Neural network models may be programmed to perform some computation (input-output mapping) by setting the elements of the weight matrix to appropriate values. For small simple tasks (such as implementing simple Boolean functions) the weights may be easily set by hand. However, this is not a means by which neural networks may be generally engineered to perform the task at hand. Representations of the problem are distributed through the whole network and the particular values

1 A Liapunov function is a monotonically decreasing function which equals 0 at the fixed point of a dynamical system and is greater than 0 at all other points.

required by each weight may not be obvious. In this case the network weights should be set by some automatic learning procedure (see [78] for a good review of learning in neural network models).

In static network models in which the desired output values for each node of the network are known, then it is relatively easy to devise an algorithm to set the values of each weight. Such algorithms include the perceptron convergence procedure for single layer feed-forward networks (perceptrons) [173] and Hopfield's non-iterative prescription for storing vectors in binary threshold symmetric networks [79]. However, these networks lack the computational power to tackle many problems [131].

More recently, learning procedures have been introduced for networks with *hidden* nodes — i.e. nodes for which there is no desired state. The principal problem of designing learning algorithms for such networks is that of credit (or blame) assignment. That is, given that the network produces an incorrect output for particular input data, which weights should be adjusted to improve its behaviour. In the case of multiple layer feed-forward networks in which the input vector is propagated through one or more layers of hidden units before reaching the output layer (multi-layer perceptrons), a solution to the credit assignment problem has been the back-propagation algorithm [208, 145, 175]. This algorithm assigns errors to each weight in the network by back-propagating the overall error from the output layer to the input layer via the hidden layers using the chain rule for differentiation.

So far, we have considered the case in which there is an external "teacher" which is able to offer the correct network output, given a particular input. Such *supervised* learning is not the only type of neural network learning procedure. In *reinforcement* learning there is also feedback from the environment; however, this is not corrective feedback, but merely a signal indicating the cost or benefit of a particular output given the input. Learning may also occur in the absence of environmental feedback. *Unsupervised* learning procedures are used to enable networks to learn the structure and regularities of the input data. In these processes,

the network learns to self-organise in accordance with some general organising principle.

It seems likely that such procedures of learning from examples are necessary to solve ill-posed problems for which no general specification exists: all that is available are examples of correct (and incorrect) behaviours. Examples of such problems include speech recognition and vision. Although some prior knowledge of the problem may be available, it is not known how to construct an algorithm from this knowledge to solve the problem at hand or whether this knowledge is sufficient to solve the problem. The method of learning from examples is not specific to neural network approachs; learning from examples has been used in control engineering [29], adaptive signal processing [209], computational geometry [45] and geophysical modelling [76]. Many of the algorithms applied in these disciplines have been rediscovered as "new" neural network learning algorithms.

However, the application of such algorithms to neural network systems has been performed with the explicit realisation that these are learning algorithms, and this has acted as a unifying factor between various disciplines (as diverse as neurobiology, control engineering and physics). This realisation has enabled existing algorithms to be viewed from a new perspective. It has also acted as an incentive to obtain a deeper understanding of neural network learning algorithms. For example, neural networks have been applied to speech recognition problems for some time. This approach was originally motivated by the idea that neural networks model the brain, the only known general continuous speech recogniser. Subsequently, neural network learning for speech recognition has moved from a vague and naïve biologically-inspired methodology (see Minsky and Papert for a detailed criticism [131]) to a sophisticated mathematical framework (e.g. [103, 109, 26]).

Neural network learning has been applied to many prediction, identification and classification problems. These problems may be classified as static or dynamic, depending on whether they possess an intrinsic

time-dependency. Problems such as image restoration and biochemical structure determination are static problems, requiring only spatial modelling. However many problems have an explicit temporal component which must be modelled. Such problems include speech recognition and production, motor control and stock market prediction.

1.2 Speech recognition

The recognition of continuous speech by a computer is an important, but difficult problem. Current state of the art systems are capable of good performance in a limited domain, but unconstrained speaker-independent continuous speech recognition has not yet been approached. Most state of the art systems use sophisticated stochastic models of speech dynamics; these models utilise simplistic assumptions, but their performance in limited domains does indicate that they capture some of the essence of speech dynamics. Their principal advantage over most current (static) neural network systems is that they are able to model the time-dependent nature of speech. Stochastic models too are trained from examples and can generalise to new data.

However, many speech recognition researchers have become interested in using neural network models for speech recognition. There are two principal reasons for this interest:

- static neural network models offer a natural way to discriminatively train a speech recognition system;
- dynamic neural network models offer the promise of modelling speech dynamics in a more sophisticated way than current methods allow.

Discriminatively training a speech recogniser means that in addition to increasing the likelihood estimate of the correct class (word, phoneme, etc.) on presentation of a training example, the likelihood estimates of the incorrect classes are lowered. This is in contrast to maximum likelihood training methods in which each item of training data is used

to refine the model of its assigned class. Thus discriminative training of a speech recogniser is designed to encourage the correct choice between competing classes, whereas maximum likelihood training is designed to build the best possible model of each class from the data. If the search space includes the true model for each class (e.g. if each class is Gaussian, and the search space is the space of Gaussians), then maximum likelihood training is an optimal strategy. If it is not (which is the case for most real world problems such as speech recognition) then discriminative training is preferable.

There have been many experimental systems that have applied neural network models to speech recognition. Most of these models have been static models that have regarded the speech signal as a series of (overlapping) static segments. Extensions have been added to these models to provide a crude model of speech dynamics. These extensions have included the use of time-delays (thus offering a limited memory of previous events) and of time-dependent post-processing of the outputs of the static network.

1.3 Dynamic networks

Dynamic networks are those for which convergence to a fixed point attractor is not guaranteed. There are several motivations for their study:

- dynamic networks seem more appropriate than static networks to model dynamic processes such as speech;
- dynamic networks are computationally more powerful than their static counterparts;
- neurobiological experiments indicate that biological neural systems have non-trivial dynamic behaviour (e.g. [58, 47, 69]);
- dynamic networks are interesting physical systems.

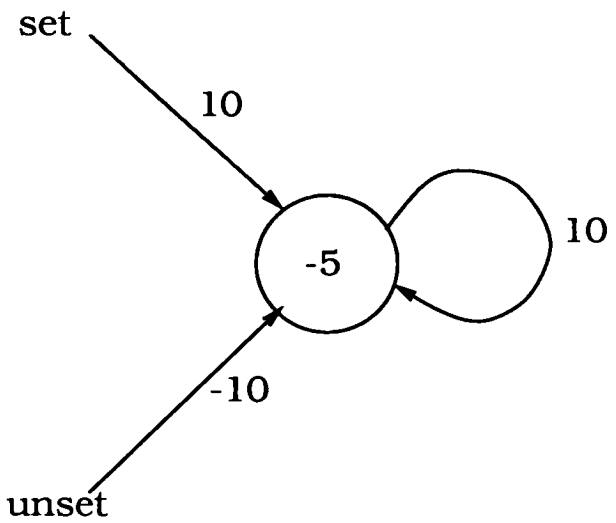


Figure 1–1: A simple "neural" implementation of a flip-flop, requiring a recurrent connection. The node is a weighted summer with a sigmoid transfer function (the model is defined in (5.4)). The weights of the connections and unit bias are shown.

However, research in dynamic networks is still at an early phase. The problem of credit assignment in learning is more difficult for dynamic networks compared with their static counterparts. Unlike static networks, each timestep may not be considered in isolation, as the activity of a network propagates forward through time. The state of a network at a given time is dependent on its state at previous timesteps. For example, if a dynamic network is trained using the back-propagation algorithm, the back-propagation of error must be applied through time as well as through the network [175]. Indeed, there is still considerable ignorance about the dynamical behaviour that may be expected from the simplest dynamic networks.

Dynamic networks offer more computational power than static networks. For example, simply implementing a flip-flop in a neural network requires a recurrent connection (figure 1–1). Recurrent connections, producing dynamic behaviour, impart a memory into a network, allowing a notion of internal state. This allows a network to respond to distant temporal events, for instance. An example of this is given in figure 1–2.

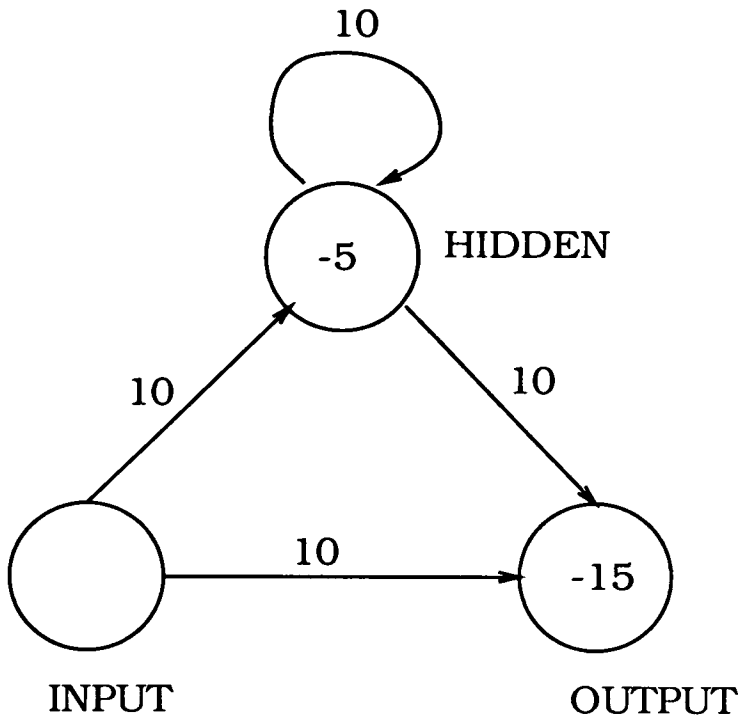


Figure 1–2: A three node network (using the same neural network model as figure 1–1) that can respond to temporally distant input. The input node is clamped to the value of some external signal. The first firing (1) of the input node acts as a priming signal, by setting the hidden node to "on". A second signal will cause the output node to fire. An arbitrary amount of time may pass between the first and second firings of the input node.

In this work, the principal motivation for studying dynamic networks (apart from their intrinsic interest) is the hope of developing a better model of speech dynamics. Presently, the most powerful models of speech are based on hidden Markov modelling. These are very simplistic models, with the units of speech being represented as piecewise stationary systems. Their advantage is that they are well understood algorithms, with a powerful associated learning algorithm. However, hidden Markov model (HMM) systems suffer fundamental limits, since an arbitrary finite state machine cannot be built out of a first order HMM; HMMs do not have a memory, thus they cannot be trained to learn long-distance temporal dependencies. Additionally the piecewise stationarity of HMMs limits their accuracy as models of speech dynamics. By developing dy-

namic networks and associated learning algorithms, it is hoped that more accurate models of speech dynamics may be developed.

1.4 Overview of thesis

This thesis has two major parts. Firstly a particular feed-forward network was studied and applied to various problems in speech recognition. Various additions were made, such as the incorporation of delays and a dynamic programming post-processing of the network's output, to enable the system to be used to perform single-speaker continuous speech recognition in a limited domain. Secondly, a network model with feedback was studied, with the aim of understanding more about the temporal complexity that this model might exhibit. This computational investigation was extensive involving a large survey of network power spectra and a detailed study of the dynamics exhibited by a particular network.

In chapter 2 the radial basis functions network is defined and a learning algorithm is described. This network is developed from work in functional interpolation and its operation is defined in these terms. The radial basis functions network is shown to be a general model for pattern recognition and its close relation to several statistical methods and neural network models is examined.

Chapter 3 contains a review of neural network models used for speech recognition. Starting with work performed using static networks for static speech pattern classification problems, this chapter describes the various means which have been employed to make neural networks better models of speech dynamics. The later sections of the chapter demonstrate that neural network models and stochastic models are not distinct approaches to speech recognition, by discussing hybrid approaches and "neural" implementations of stochastic models. Indeed, HMMs may be regarded as a subset of neural network models.

The results of some speaker-dependent speech recognition experiments carried out using a system based on a radial basis functions network are

reported in chapter 4. The first experiments were static pattern classification experiments, in which vowels excised from continuous speech were classified using feed-forward networks. This work was then extended to another static pattern classification experiment in which 5 ms frames of speech were classified according to their phonetic labels. A post-processor assuming a first order Markov model was then applied to the output of this second set of networks, in an attempt to crudely model the speech dynamics. The output of passing a speech signal through this phoneme modelling system was a lattice of probabilistically scored phoneme hypotheses, which was further processed by the CSTR language modelling system, to produce word and sentence hypotheses. All these experiments were compared to the results obtained using "traditional" statistical methods.

Chapter 5 considers dynamic networks. After a review of previous studies of network dynamics, a dynamic neural network model is defined and its dynamics (in a four-dimensional phase space defined by the symmetry of the weight matrix, the gain of the nodal transfer function, the network size and the discretisation constant used to digitally simulate the continuous equation of motion) investigated. This study took the form of a survey of network power spectra, where a network power spectrum (the power spectrum of the time history of outputs of a node in the network) was taken to be informative about the temporal behaviour of the network. By summarising these power spectra into a few statistics a map of the dynamical behaviour of this network model's behaviour (in the given phase space) was constructed.

The study of this model is continued in chapter 6. Here the dynamics of a particular network were investigated relative to this phase space. The methods used here were those that have been developed to characterise nonlinear dynamical systems, and it was found that this network exhibited complex behaviour, including chaos, that could be described with few dimensions.

Finally the work is summarised and concluded in chapter 7.

2

RADIAL BASIS FUNCTIONS

2.1 Learning from examples

2.1.1 Functional interpolation

The basic problem of learning an input-output mapping from examples has been formulated in several different ways. In control theory the problem is that of system identification and estimation; in statistical pattern recognition the problem may be approached parametrically where forms for the probability distributions are assumed or non-parametrically where assumptions about the input probability distributions are not required; in neural network modelling the problem of learning from examples has been framed as the construction of an associative memory which retrieves an appropriate output when presented with an input.

It is apparent from a consideration of these methodologies that a crucial concept applying to learning from examples is *generalisation*[1]. Here, we shall find it convenient to express notions of generalisation in terms of *functional interpolation*; learning an input-output mapping from examples may be understood as a process of curve-fitting, in which a high-dimensional surface is constructed from the input data. Generalisation

1 A notion of generalisation was formalised by Vapnik and Chervonenkis [201] and Valiant [199] in what is commonly called the PAC (Probably Approximately Correct) learning model. In PAC learning the aim is to construct a function which will produce incorrect outputs from future unseen inputs at most ϵ of the time with a probability $\geq 1 - \delta$. In this model the training input and all future examples are assumed to be drawn from the same, arbitrary probability distribution (thus excluding any information about the prior distribution).

is then an interpolation between known points on this surface [27, 102].

It is easy to see how this notion of learning and generalisation is instantiated by a feed-forward neural network. In a feed-forward network an n-dimensional input space is mapped to an m-dimensional output space; that is the network executes a function $\mathcal{F} : \mathbf{R}^n \rightarrow \mathbf{R}^m$. Consider a feed-forward network with a single hidden layer. If the input vector is given by $\mathbf{x}$, then

$$\mathbf{y} = \mathcal{F}(\mathbf{x}) = W\mathbf{f}(A\mathbf{x}) \tag{2.1}$$

where $\mathbf{f}$ is the vector of non-linear transfer functions of the hidden nodes, $\mathbf{y}$ is the output vector, A is the first layer weight matrix and W is the second layer weight matrix. The surface that is fitted to the data is constrained to be a linear combination of the non-linear hidden unit transfer functions. Clearly additional hidden layers (or more hidden units in a single layer) allow more complex surfaces to be produced. A fitting procedure or learning algorithm is used to choose the weights parameterising the network (W and A). The operation of such an algorithm may be viewed as the reduction of the volume of accessible weight space [62, 197] given the training examples. There are two criteria that should be considered: the volume of accessible weight space[2] (which measures the flexibility of the set of functions that can implement the input-output relations presented in training); and the ability of the learning algorithm to meet the constraints and generate a set of weights within the accessible weight space. If the volume of accessible weight space is small, then good generalisation is likely as the weight sets in this volume are likely to be of low complexity, corresponding to smooth interpolation surfaces. However, the problem of searching for a weight set in this volume (addressed by the learning algorithm) may be difficult and an acceptable weight set may not be found in a reasonable time. If the volume of ac-

2 An issue here is the appropriate measure to use to quantify the volume of accessible weight space. For example, one may not place the same importance on volumes at $|W| \rightarrow \infty$ as volumes at $|W| \rightarrow 0$, as it is generally desirable to have weight matrices of low magnitude.

cessible weight space is larger, then the learning algorithm may quickly converge to a weight set in this volume, but generalisation is likely to be poorer as the set of accessible functions available to the network is larger. This increase in flexibility means that the resultant solutions are more likely to have modelled fluctuations in the training data. Achieving good generalisation in reasonable time necessitates a trade-off between these two criteria of flexibility and ease of learning. In terms of a feed-forward network the decision is made when deciding on the network architecture (i.e. the number of free weights) [16]; in engineering applications problem-dependent knowledge is crucial in designing minimal networks capable of representing the desired function [104].

2.1.2 Radial basis functions

The method of radial basis functions (RBFs) [153, 151] is a powerful and general method of constructing surfaces to solve the problem of high-dimensional interpolation. In this section an outline of the method and important associated theoretical results will be presented.

The multi-dimensional interpolation problem being studied may be stated as follows: Consider an n-dimensional input space $\mathbf{R}^n$ and a scalar output space $\mathbf{R}$. Given P different points X_p in $\mathbf{R}^n$ together with their corresponding outputs Y_p in $\mathbf{R}$, the problem is to construct a function $\mathcal{F}$ such that:

$$\mathcal{F}(X_p) = Y_p \qquad p = 1, 2, \ldots, P. \tag{2.2}$$

The method of radial basis functions chooses $\mathcal{F}$ to be a linear combination of radially symmetric functions centred on the input points X_p. The functions used are the so-called "radial basis functions", $\Phi(\|X - X_p\|)$, so $\mathcal{F}$ becomes:

$$\begin{aligned} \mathcal{F}(X) &= \sum_{p=1}^{P} w_p \Phi(\|X - X_p\|) \\ \Phi &: \mathbf{R}^+ \rightarrow \mathbf{R} \end{aligned} \tag{2.3}$$

where $\mathbf{R}^+$ denotes the non-negative real numbers. This system of equations will have a unique solution provided the matrix

$$y_{pq} = \Phi(\|X_p - X_q\|)$$

is non-singular, where p and q range over the input data points. Micchelli [130] proved that this is indeed so (provided the data points are non-degenerate) for a large choice of Φ including:

$$\Phi(r) = r \tag{2.4}$$

$$\Phi(r) = (r^2 + c^2)^\beta \quad 0 < \beta < 1 \tag{2.5}$$

$$\Phi(r) = r^2 \ln r \tag{2.6}$$

$$\Phi(r) = \exp\left(\frac{-r^2}{2\sigma}\right). \tag{2.7}$$

The radial basis functions method is closely related to surface spline methods of interpolation [99]. The method of thin plate splines derived by Duchon [45] is derived from a variational approach to interpolation. The solution to this problem is given by:

$$\mathcal{F}(X) = \sum_{i=1}^{P} w_p \Phi(\|X - X_p\|) + \sum_{p=1}^{k} v_p \rho_p(X) \tag{2.8}$$

where the $\rho_p(X)$ are additional polynomial basis functions and $\Phi(r) = r^2 \ln r$ (2.6).

The multiquadratic interpolation method [76] is a particular case of a general RBF method given by (2.3). Here $\Phi(r) = \sqrt{r^2 + c^2}$ (2.5); this method has been used extensively in surface (2-dimensional) interpolation.

2.2 Network implementation

2.2.1 Approximated radial basis functions

Broomhead and Lowe [27] demonstrated that the method of radial basis functions may be regarded as a feed-forward network with a single hidden layer (figure 2–1). Regarding RBFs as hidden units with radi-

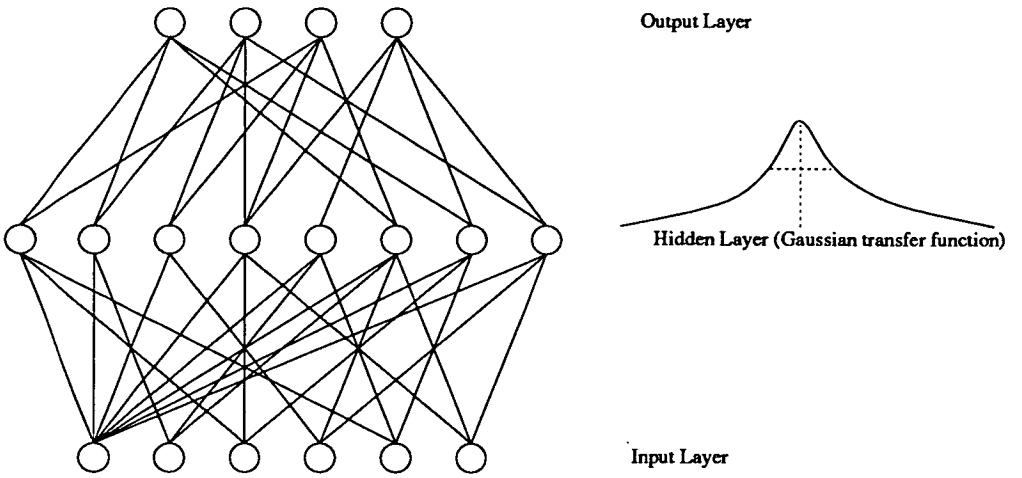


Figure 2–1: Sketch of a radial basis functions network (not all connections are shown). This is a 2 layer feed-forward network, with the hidden units having Gaussian transfer functions. The input-to-hidden weights may be regarded as centres in input space; the hidden units also have a width associated with them.

ally symmetric fields is a compelling analogy, particularly in the case of Gaussian RBFs. These functions are centred on a particular input point and exponentially decay to zero; additionally they may be factored into a product of two-dimensional Gaussians, which lends some neurobiological plausibility [151]. The output units are simply linear summers with no nonlinearity. An RBF network may be viewed alternatively as a linear network with an input preprocessor. In this interpretation the RBFs are used to transform the input data into a different space, usually of higher dimension.

The above discussion concentrated on the strict interpolation problem: a set of basis functions was constructed, with each member of the set of data points acting as the centre for a basis function. When there is a very large set of training examples it is clearly infeasible to attempt a strict interpolation. For example, a continuous speech recognition frame labelling problem reported later (chapter 4) involved 140,000 frames of training data. The inversion of a 140,000 $\times$ 140,000 matrix would be computationally massive (requiring centuries of computation on a Sun 4 computer); additionally the probability of a matrix being ill-conditioned

grows with size of the matrix [43]. Broomhead and Lowe [27], in their presentation of the radial basis functions method as a feed-forward network, relax this constraint, allowing the number of RBFs R to differ from the number of training examples P (Poggio and Girosi refer to this as an approximated RBF method [151]). Other extensions included extending the output space from a scalar quantity to a vector (so $\mathcal{F} : \mathbf{R}^n \rightarrow \mathbf{R}^m$) and adding a bias ($w_{i0}$) to the linear network (thus allowing hyperplanes to be constructed away from the origin). $\mathcal{F}$ is then a vector-valued function given by:

$$\mathcal{F}_i(X) = w_{i0} + \sum_{r=1}^{R} w_{ir}\Phi(\|X - c_r\|) \qquad i = 1, 2, \ldots, m \tag{2.9}$$

where c_r is an RBF centre (or "knot") in the input space and w_{ir} are the weights of the linear network.

This is an over-constrained system and the matrix of RBF outputs y_{pr} is now a $P \times R$ matrix with the optimal solution, in the least squares sense, being given by the Moore-Penrose pseudoinverse (see [95] for a good discussion). The pseudoinverse of y, y^+, is given by[3]:

$$y^+ = (y'y)^{-1}y' \tag{2.10}$$

provided $y'y$ is non-singular, where y' denotes the transpose of y.

Poggio and Girosi [151] extended Micchelli's theorems guaranteeing non-singularity to this situation, given that the centres chosen are a subset of the set of data points. This method has been used by several researchers, e.g. [27, 30, 139, 207]. In summary, a system with a greater number of centres than data points is guaranteed to be singular; Micchelli showed (for many forms of Φ) that a strict interpolation system with the same number of centres as data points is guaranteed non-singular; Poggio and Girosi made a simple extension to Micchelli's result to the over-constrained situation (fewer centres than data points), guaranteeing non-singularity to the same class of functions, Φ.

3 The pseudoinverse may also be found using a direct pseudoinverse routine such as singular value decomposition or by applying Greville's theorem [95].

2.2.2 An alternative training method

In the work reported here, an alternative (but equivalent) training method was used [169]. Consider a linear network with a single layer of weights, whose input layer corresponds to the outputs of a set of radial basis functions. For each training example p, a real number y^H_{rp} is output by each RBF r. The "activation" or "potential" y^T_{ip} on each target node i will be computed using the (as yet undetermined) weights w_{ir}:

$$y^T_{ip} = \sum_r w_{ir} y^H_{rp}. \tag{2.11}$$

Let the desired activation value for exemplar p on target node i be called Y^T_{ip}. The optimal weights are defined as those for which the error measure:

$$E = \frac{1}{2} \sum_{ip} (y^T_{ip} - Y^T_{ip})^2 \tag{2.12}$$

or

$$E = \frac{1}{2} \sum_{ip} \left(\sum_r w_{ir} y^H_{rp} - Y^T_{ip} \right)^2 \tag{2.13}$$

is smallest. This minimum may be found by a pseudoinverse method (see above) or by finding where the derivatives

$$\frac{\partial E}{\partial w_{kl}} = \sum_r w_{kr} \left(\sum_p y^H_{rp} y^H_{lp} \right) - \sum_p Y^T_{kp} y^H_{lp} \tag{2.14}$$

vanish. Let M be the correlation matrix of radial basis function outputs, summed over training examples:

$$M_{rl} = \sum_p y^H_{rp} y^H_{lp} \tag{2.15}$$

and let M^{-1} be the matrix inverse of M. Then the optimal weight matrix W^* lies where the gradient vanishes:

$$w^*_{ir} = \sum_k \left(\sum_p Y^T_{ip} y^H_{kp} \right) M^{-1}_{kr} \tag{2.16}$$

Thus the problem may be exactly solved by inverting a square $R \times R$ matrix, where there are R radial basis functions.

It may be demonstrated that this method is indeed equivalent to the direct pseudoinverse method. In that method the optimal weights are given by:

$$w^* = Y^T y^{H+}. \tag{2.17}$$

If identity (2.10) is applied to (2.17) then

$$w^* = Y^T y^{H'} (y^H y^{H'})^+ \tag{2.18}$$

which is just (2.16) in matrix form. Formula (2.18) is obtained in [207], but it is explicitly stated that (2.17) is used in computer simulations.

Rough calculations [169] show that exact methods such as this require the same amount of computation as a small number of gradient evaluations of E; hence this is preferable to iterative schemes of minimisation. An advantage of iterative minimisation methods (such as the LMS algorithm [209]) is that they allow online training[4]. Although the training method described here involves a matrix inversion, the dimensions of the matrix to be inverted are only dependent on the number of RBFs and not the number of training examples; a typical application may require 100 RBFs, requiring the inversion of a 100×100 matrix — this takes 6 seconds to compute on a Sun 4. This training method may then be described as a "semi-online" method, as the matrices M and $(Y^T y^{H'})$ may be computed in an online fashion. This is not true of the direct pseudoinverse method: the dimension of the matrix to be pseudoinverted is dependent on the number of training patterns, hence the bulk of the computation is performed after all the training data has been presented. Additionally the direct pseudoinverse method is computationally more expensive, if $P >> R$. The computation time required to solve (2.17) directly is approximately $\mathcal{O}(P^{3/2})$, whereas solving the form given in (2.18) is approximately $\mathcal{O}(P)$. Genuine online problems (such as the adaptive

4 Training a network may be classified as *online* or *batch*. In online training the weights of a network are adjusted after each training pattern has been presented; in batch training the weights are not adjusted until after the presentation of the entire set of training patterns.

improvement of a channel equaliser) cannot be solved using these non-iterative methods, so an adaptive algorithm (e.g. the LMS algorithm) must be used.

2.3 Related statistical methods

The generality of the radial basis functions formalism becomes apparent by demonstrating that various statistical pattern recognition methods may be regarded as RBF methods. In particular, density estimation using Parzen windows [146], the method of potential functions [2] and Bayesian learning using Gaussian (or mixture Gaussian) density functions may all be described using the RBF formalism. In the case of Gaussian classifiers and Parzen windows the RBFs are used to construct probability density functions (PDFs); each RBF is committed to a particular class. In statistical pattern recognition, methods of probability density estimation using basis functions centred at points in the input space are termed *kernel* methods[5]. In the method of potential functions, the RBFs are not probabilistically interpreted, and are used as interpolating functions. A good text covering these statistical pattern recognition methods is the one by Duda and Hart [46].

2.3.1 Gaussian classifiers

The application of Bayes' rule is a fundamental of statistical pattern recognition. In a classification problem, a classification into class c_i given input data x is required. This may be achieved by maximising the posterior probability $p(c_i|x)$. Clearly this conditional probability is

5 Poggio and Girosi [151] made a connection between radial basis functions and the kernel functions of mathematical physics. Specifically, they treated the problem of functional approximation using the methods of regularisation theory. The solution to this regularisation problem has a kernel given by a Green's function of a constraint differential operator. (This operator expresses prior constraints on the form of the approximating functions and also any prior knowledge of the problem.) It was shown that if the Green's function is radial, then the regularisation theory solution is an expansion into radial basis functions.

not directly available; however we may apply Bayes' rule:

$$p(c_i|x) = \frac{p(x|c_i)P(c_i)}{p(x)}. \tag{2.19}$$

The probabilities $p(x|c_i)$ may be estimated from the data, e.g. by compiling a histogram. The prior probabilities $P(c_i)$ denote the probability of each class occurring without referring to the input data and reflect prior knowledge of the problem. Training then becomes equivalent to learning the conditional probability density functions (PDF) $p(x|c_i)$.

Here we shall assume that the PDFs for each class take on the form of the d-dimensional normal density:

$$p(x|c_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\right) \tag{2.20}$$

where μ_i is the mean of class c_i and Σ_i the corresponding covariance matrix. Learning corresponds to estimating values for these parameters. This is the most popular PDF to have been investigated: not only is it analytically tractable, but it also models the situation where the input vectors corresponding to a given class are noisy versions of a single typical vector.

This classifier may be expressed as an RBF network (figure 2–2). In such a network the number of hidden units (RBFs) is specified by the number of classes in the problem. there is one hidden unit for each class, representing a PDF (2.20). Each hidden unit has just one output connection, leading to its corresponding output unit, its weight w_{ii} being the prior probability of class c_i.

This may be contrasted with a general RBF network which features a fully connected hidden-to-output weight matrix. This gives a clear illustration of the different methods of training such a network. The Gaussian classifier is trained by a maximum likelihood method: a model is built for each class and optimised to maximise the likelihood of the data given the class. This is in contrast to the discriminative least-squares type of training usually applied to RBF networks, when in addition to maximising the output of the desired class, the output of the other classes

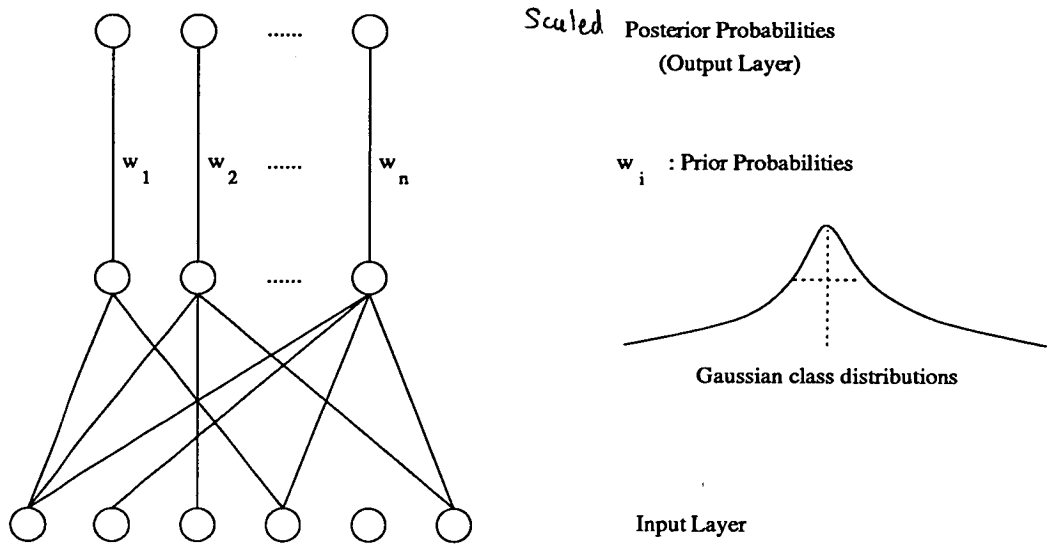


Figure 2–2: Sketch of a Gaussian classifier expressed as an RBF network. There is one hidden unit for each class, with each RBF centre corresponding to the estimated class mean and the covariance of an RBF corresponding to the estimated class covariance. The (diagonal) hidden-to-output weight matrix depends on the prior probabilities of each class.

is minimised. Choosing maximum likelihood or discriminative training (or a mixture of the two) is a design decision and is discussed in greater detail in chapter 4.

An alternative means of adding discrimination to a Gaussian classifier has recently been suggested by Yau and Manry [212]. In this work, the prior probabilities were neglected (i.e. assumed to be uniform), and a neural network was designed that implemented the Gaussian conditional probabilities, $p(x|c_i)$. This network was a sigma-pi network[6], obtained by taking logs of the Gaussians. The weights of the network were obtained

6 A sigma-pi network features second-order connections; the output of a particular unit is given by:

$$y_i = f\left(w_{i0} + \sum_j w_{ij}y_j + \sum_{kl} w_{ikl}y_k y_l\right).$$

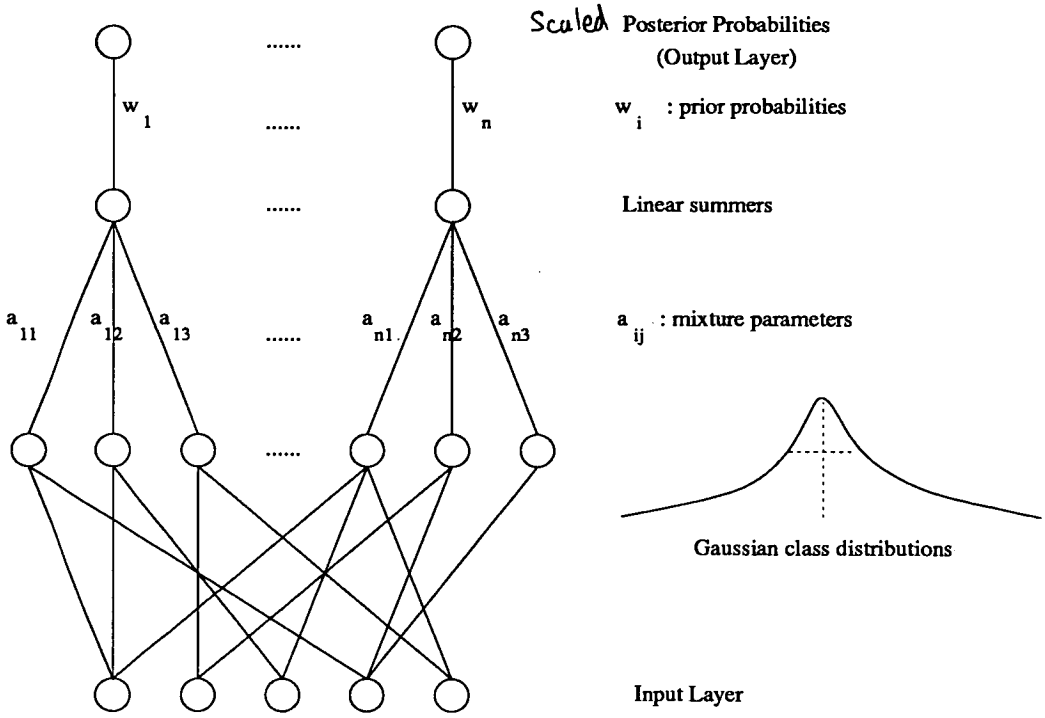


Figure 2–3: Sketch of a mixture of Gaussians expressed as a RBF network. The prior probabilities and mixture coefficients are shown as two separate sets of weights (leading out of and into a linear summer) for extra clarity; it is trivial to collapse the priors and mixture coefficients into a single weight set.

from the means and covariances of the Gaussian classifier. In Yau and Manry's system, the Gaussian network was initialised using the sample means and covariances. The classifier was further improved by discriminative training using the back-propagation algorithm. In experimenting with this system, it was found that the discriminatively trained classifier, performed better than the maximum likelihood Gaussian classifier (to which the former was initialised) on problems in which the means and covariances were not known accurately (as in most real world problems).

Multivariate Gaussian PDFs are limited to representing unimodal distributions. Presuming that one wishes to retain the advantages of a parametric method, a more general form of PDF is required. A suitable

function to choose is a mixture of Gaussians:

$$\begin{aligned} p(\mathbf{x}|\Theta_i, c_i) &= \sum_j a_j p_j(\mathbf{x}|\theta_{ij}, c_i) \\ \sum_j a_j &= 1. \end{aligned} \tag{2.21}$$

Here a class c_i is described by a parameter vector Θ_i, which is a vector of the means and covariances of the individual Gaussians making up the mixture. The mixture parameters a_j define how the Gaussians are linearly combined to produce the final mixture density. These parameters are obtained via a maximum likelihood optimisation performed using the EM algorithm [44, 158]. Again this method may be represented as an RBF network (figure 2–3).

2.3.2 Parzen windows

The Parzen windows approach to non-parametric estimation of probability densities [146] is probably the most frequently used kernel method of density estimation. The basis of the method involves fixing a set of volumes in input space (centred at the data points) and counting the number of example points in each volume. This hypercube or hypersphere window function may be extended to a more general class of functions — Gaussians are typically used. In this case the basic equation for Parzen windowing is similar to (2.3), where Φ is the Parzen window function. The sum in (2.3) is then normalised by the number of data points to give an average of the probability densities centred at the samples. A further parameter used in this method is referred to as the smoothing parameter (h). This is used to specify the width or variance of each window and is normally constant over all windows.

This method is a non-parametric method and may be contrasted with a Gaussian classifier. In a Gaussian classifier, a particular, Gaussian, form is chosen for each output PDF. Each training data point serves to "tune" the parameters (means and variances) of its corresponding PDF. In the Parzen windows method, however, the addition of a new training

data point alters the function space from which the resultant PDF is selected; a new training point introduces a new window (and thus more parameters). Additional training points do not merely refine an existing set of parameters, and the final functional form of the estimated PDF is dependent upon the training data.

If the estimated density function for a class c_i is written:

$$p(x|c_i) = \frac{1}{N_i h} \sum_{k=1}^{N_i} \Phi\left(\frac{\|x - x_k^i\|}{h}\right), \tag{2.22}$$

where there are N_i examples of class c_i, then a discriminant function separating two classes c_i and c_j may be written:

$$D(x) = \frac{1}{(N_i + N_j)h}\left(\sum_{k=1}^{N_i} \Phi\left(\frac{\|x - x_k^i\|}{h}\right) - \sum_{k=1}^{N_j} \Phi\left(\frac{\|x - x_k^j\|}{h}\right)\right). \tag{2.23}$$

This approach to pattern recognition is sometimes termed kernel discriminant analysis [75].

A good deal of theoretical work has been undertaken to determine optimal forms for the kernel function (RBF) Φ and the smoothing parameter h used to scale the influence of each window away from its central point [75]. In the case of Parzen windows the most popular choice of kernel function has been the Gaussian normal density, and various *ad hoc* methods for choosing h have been employed. Again, a choice must be made whether to optimise parameters using a maximum likelihood process ("goodness of fit") or discriminative training process ("minimal misclassification").

2.3.3 Potential functions

Rather than interpolating between data points to construct a density function, the method of potential functions performs an interpolation to estimate a discriminant function in input space. Consider a two class problem: if each training example is regarded as being a charged point in input space, whose charge is unit positive or negative, depending on its class, then the resulting surface of zero potential serves as a discriminant

function. If the charge at a point x in input space due to point charge x_i is denoted by $K(x, x_i)$, then the potential at x due to n point charges is:

$$\Psi(x) = \sum_{i=1}^{n} c_i K(x, x_i) \tag{2.24}$$

In classical physics the potential varies inversely with $(\|x - x_i\|)$, but as Aizerman et al. [2] discuss, a wide range of forms for $K(x, x_i)$ are suitable. Functions that have been used include:

$$K(x, x_i) = \frac{\sigma^2}{\sigma^2 + \|x - x_i\|^2} \tag{2.25}$$

$$K(x, x_i) = \exp\left(-\frac{1}{2\sigma^2}\|x - x_i\|^2\right). \tag{2.26}$$

Choices for $K(x, x_i)$ have generally tended to possess a maximum when $x = x_i$ and to monotonically decrease to 0 as $\|x - x_i\|$ tends to ∞. In the case when K is a radial function, i.e. $K(x, x_i) = \Phi(\|x - x_i\|)$ (as in (2.25) and (2.26)), it may be identified with the RBF Φ in (2.3).

Potential functions research has tended to concentrate on iterative algorithms (especially perceptron training) to choose the weights c_i, which is in contrast to work on radial basis functions, which have used efficient non-iterative means of solving the output linear network. A typical iterative rule used to optimise $\Psi(x)$ is:

$$c_i(t + 1) = c_i(t) + V(Y(t), \Psi(x(t)) K(x(t), x_i), \tag{2.27}$$

where V is a function of the error dependent on the target output $Y(t)$ and the actual output $\Psi(x(t))$.

2.3.4 Vector quantisation

Vector quantisation (VQ) is a process in which the input space is partioned into a set of disjoint regions. A real valued input vector may thus be transformed into a symbol corresponding to the region in which it is located. This process may be regarded as an RBF method using hard-limiting hyperspherical basis functions in (2.3):

$$\begin{aligned} \Phi(x) &= 1 \quad \text{if } \|x - x_i\| \leq d \\ \Phi(x) &= 0 \quad \text{otherwise} \end{aligned} \tag{2.28}$$

Φ is a hypersphere of radius d centred at x_i. However, most vector quantisers have more complicated forms of Φ than this, generally hard-limited polygons. $\Phi(x)$ for each RBF is determined relative to the other RBFs present, using a nearest neighbour metric. The RBF for each centre defines the region of input space which is closer to that centre than any other centre. This partioning of the input space is often referred to as a Voronoi tessellation.

Several algorithms have been proposed for computing the centres (or codewords) of a VQ codebook [122]. A commonly used algorithm is the iterative k-means clustering algorithm [121]. After initialising (e.g. at random data points) a Voronoi tessellation is computed and each codeword is repositioned at the centroid of the set of data points within its partition. This process is iterated until convergence.

The advantage of VQ is that it allows computation to proceed using a finite number of discrete probability distributions, rather than computing the continuous PDFs (e.g. Gaussian mixture densities). However a drawback is that the VQ operation will undoubtedly distort the input data, owing to the hard partioning of the input space. The RBF formalism suggests a compromise solution that may mix the advantages of both discrete and continuous modelling. By using continuous valued RBFs for the VQ, the VQ codebook may be modelled as a mixture of continuous functions which are overlapped rather than disjoint. This method was suggested by Huang [82] in the context of hidden Markov modelling, (semi-continuous hidden Markov modelling) (see section 4.5.3.2).

2.4 Related neural network models

The RBF framework is quite general, with several other neural network architectures having close relationships to it. RBF networks clearly have a close relationship to multi-layer perceptrons [175], since both are layered, feed-forward networks. Additionally, there is a large group of neural network architectures that are based upon a preprocessing of the

input into a higher dimensional space. These include localised receptive fields [133], the modified Kanerva model [155], the random cells model of Gallant and Smith [61], Kanerva's memory model [89], the restricted Coulomb energy network of Reilly et al. [159], Boolean networks based on n-tuple sampling [4], Kohonen's feature maps [95] and the memory-based reasoning system of Stanfill and Waltz [191].

2.4.1 Multi-layer perceptrons

A radial basis functions network may be developed from a standard multi-layer perceptron (MLP) in a natural way. Consider a MLP with a single layer of hidden units; the input units are clamped to external input values (X_k) and the output units (y_i) are linear. The output of such a network is given by:

$$\begin{aligned} y_i &= \sum_{j=0}^{M} w_{ij} f\left(\sum_{k=0}^{N} w_{jk} X_k\right) \qquad (2.29) \\ f(x) &= \frac{1}{1+\exp(-x)} \qquad \text{(for example)}, \end{aligned}$$

where w_{ij} represents a connection weight and w_{i0} the bias. The argument to the sigmoid nonlinearity f defines a hyperplane:

$$\sum_{k=1}^{N} w_{ik} X_k = -w_{i0}. \qquad (2.30)$$

The desired class boundaries are modelled by the hyperplanes defined by the hidden nodes. Too many hidden nodes cause the boundaries to be significantly influenced by the sample distribution; too few hidden nodes lead to an inadequate modelling of the class boundaries.

MLPs may be trained using the back-propagation of error algorithm [175]. This algorithm adjusts the weights, w_{ij} of the network according to the supervised output error, E:

$$\Delta w_{ij} \propto \frac{\partial E}{\partial w_{ij}} \qquad (2.31)$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial w_{ij}}. \qquad (2.32)$$

The gradient of the error with respect to the (supervised) output units may be simply obtained, provided the error function is differentiable. To compute this gradient for the hidden units the error must be back-propagated from the output units, via a linear network, with the required gradients being computed by the chain rule of differentiation.

A feed-forward network to be trained by back-propagation need not feature weighted-sum nodes with a sigmoid nonlinearity. As long as the gradient of the error with respect to the weights can be computed, the back propagation algorithm may be performed. Consider altering the transfer function of the hidden nodes from a sigmoid to a Gaussian:

$$y_i^H = \exp\left(-\sum_j \frac{(y_j - c_{ij})^2}{2\sigma_{ij}^2}\right) \tag{2.33}$$

Here the hidden units are computing hyperellipses with centre coordinates c_{ij} and covariances σ_{ij}[7]. The required gradients for the back-propagation algorithm may be computed for a network of this type:

$$\frac{\partial y_i^H}{\partial c_{ik}} = \frac{y_i^H(y_k^I - c_{ik})}{\sigma_{ik}^2} \tag{2.34}$$

$$\frac{\partial y_i^H}{\partial \sigma_{ik}} = \frac{y_i^H(y_k^I - c_{ik})^2}{\sigma_{ik}^3} \tag{2.35}$$

where the centres and covariances of the Gaussians correspond to the adaptive weights. Such a network may be considered as a type of RBF network. An important difference is that the centres and covariances are adaptive; the expansion into radial basis functions space may not be regarded as a preprocessor prior to learning in this case. Robinson and Niranjan [166, 139] have experimented with networks of this type. Poggio and Girosi [151] have developed a solid theoretical foundation for networks of adaptive radial basis functions using approximation and regularisation theory, a foundation that they claim is not yet shared by MLPs.

7 Lapedes and Farber [102] have suggested that MLPs with 2 layers of sigmoid hidden units compute "bumps" in input space. These may be more easily modelled using a single layer of hyperellipses. Of course, MLPs may form more complex classification boundaries than bumps.

Non-adaptive RBF networks have several advantages relative to MLPs. The non-adaptivity leads to (much) faster training times, because only a solution to a linear network is required. A unique solution may be found (the least squares optimum); hence there are no problems with local minima. Furthermore, RBF networks are closely related to well understood interpolation and statistical pattern recognition methods, thus offering a clear insight into their functioning.

2.4.2 Input dimensionality expansion methods

A characteristic of RBF networks shared with several neural network models is the notion of expanding a vector into a high-dimensional space. These methods take advantage of a theorem by Cover [38] which states that a classification problem cast into a high-dimensional space is, in a specific sense, more likely to be linearly separable than would be the case in a lower dimensional space. Cover considered the partioning of a d-dimensional space by a hyperplane. If there are n sample points (in general position[8]) then there are a maximum 2^n possible dichotomies[9]. The probability that a random dichotomy is linearly separated, $P(n, d)$, is given by:

$$\begin{aligned} P(n,d) &= 1 && \text{if } n \leq d+1 \\ P(n,d) &= \frac{2}{2^n}\sum_{k=0}^{d}\binom{n-1}{k} && \text{if } n > d+1. \end{aligned} \tag{2.36}$$

(All dichotomies of d or fewer points are linearly separated.) Plotting $P(n, d)$ vs. $\frac{n}{d+1}$ for various values of d (figure 2–4), it is apparent there is a limiting value of $\frac{n}{d+1} = 2$, below which the probability that a random dichotomy is linearly separated is 1, above which it is 0. Cover described this value as the capacity of a hyperplane. This theorem implies that lin-

8 A set of points in a d-dimensional space are said to be in *general position* if no subset of $d+1$ points lies in a $(d-1)$-dimensional subspace.

9 A *dichotomy* is a partition of a set of points into two subsets.

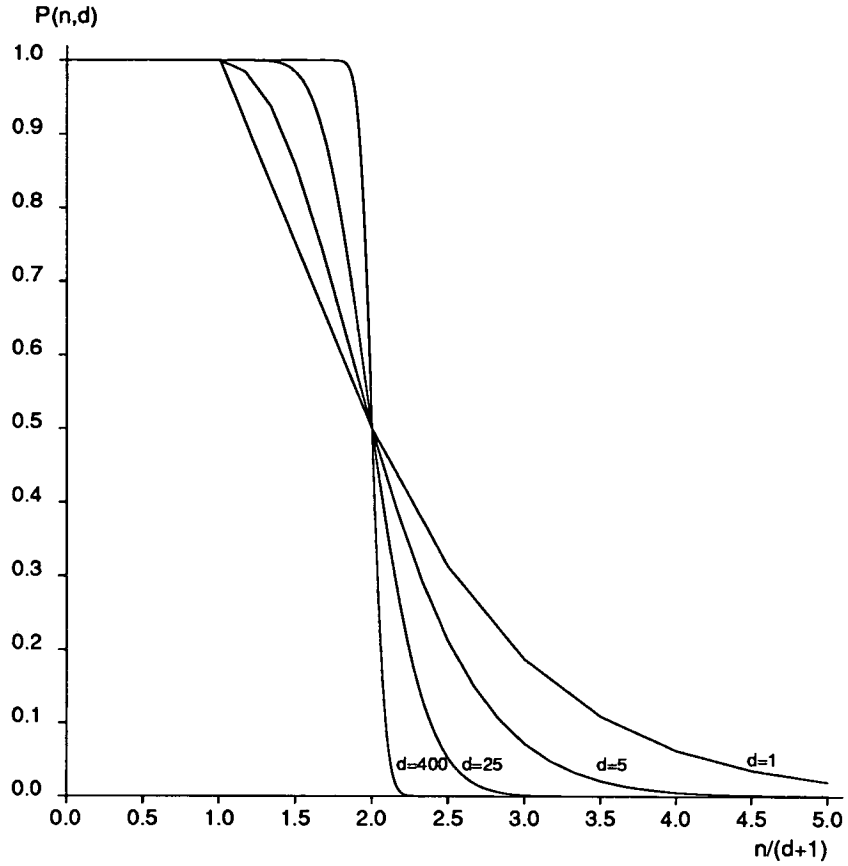


Figure 2–4: Graph of $P(n, d)$ vs. $\frac{n}{d+1}$ using (2.36). $P(n, d)$ is the probability that a random dichotomy of n d-dimensional points in general position is linearly separable. As $d \to \infty$, so $P(n, d) \to 1$ if $n < 2(d + 1)$, otherwise $P(n, d) \to 0$.

ear separability becomes more probable in a higher dimensional space[10].

The process of expanding a vector into a higher dimensional space usually supposes that the units in this space correspond to local volumes of the input space; this is the case for hyperelliptic RBF networks. Other systems using this idea of locality include the modified Kanerva model [155], networks of locally tuned processing units [133] and the SPAN network of Kawahara [91]. These methods may be interpreted as

10 Beware! Cover's work applies to random input samples and in pattern recognition the input data is generally correlated.

RBF methods, although these authors go to some pains to differentiate themselves from the method of RBFs.

The locally tuned processing units method of Moody and Darken [133] has a number of basis functions (Gaussians) placed in the input space; the centres are chosen using k-means clustering [121] and the widths (a single width for each Gaussian is assumed rather than a full covariance matrix) chosen using a nearest-neighbour heuristic. Since they were interested in time-series prediction and required on-line learning, the LMS algorithm [209] was used to assign the weights in the linear network. Moody and Darken compared this work to the method of RBFs (for strict interpolation). Their criticisms of RBFs were: (i) that RBFs are global since they do not approach 0 exponentially fast; (ii) that a separate basis function is required for each point; (iii) computation time scales as P^3 for a P point data set. These do not apply to the more general RBF formalism outlined here since: (i) Gaussians are frequently used as basis functions (rather than the multiquadratics or thin-plate splines that they evidently had in mind); (ii) the RBF formalism here goes beyond strict interpolation; (iii) computation time is dependent on the number of training examples but (see above) is distributed to allow quasi-on-line processing[11].

Kawahara and Irino [91] introduced the saturated projection algorithm (SPAN) as a general framework for regarding learning in neural networks as functional interpolation, with an additional emphasis regarding the incorporation of prior information. However in practise it is little different to RBF methods (as they acknowledge) with the idea of incorporating prior information about the problem being reduced to the choice of function centres.

The modified Kanerva model [155] took its inspiration from Kanerva's sparse distributed memory (see below), although the important features of the Kanerva model (relating to the properties of high (2^{1000})-

11 The computation time of the locally-tuned processing units method is also dependent on the number of data points, particularly as it is stated that the number of basis functions is $\frac{1}{10}$th the number of data points.

dimensional binary spaces) are lost in the feasible implementations of this scheme. In practise this is a radial basis functions model. Prager has examined the computational properties of several error metrics (d_1 (Manhattan), d_2 (Euclidean) and d_∞) in conjunction with hard threshold hyperspheres as basis functions. An important computational result was reported using the d_∞ metric [154]. Only a small fraction of basis functions are required to be active at any time. In the d_∞ metric a basis function is only active if all its coordinates lie within the corresponding activation radius. The combination of these two conditions results in a sparse weight matrix: each basis function only depends on the state of a very few input coordinates (assuming binary inputs) and these coordinates may be determined at compile time. This results in large savings in memory and computation (2 orders of magnitude in both areas for a real-time speech recognition experiment).

The method of random cells [61] involves placing random hyperplanes in the input space as a method of input dimensionality expansion, before solving with a linear network. Although not an RBF method, this method clearly bears a close relationship and similar motivation to such methods.

2.4.3 The Kanerva memory model

Kanerva's sparse distributed memory (SDM) [89] is an associative memory model, that is based upon the properties of high-dimensional (2^N, N is typically 1000) binary spaces. It is obviously impossible to model all 2^N possible memory locations of such a memory, so a small, random subset of locations (of the order of 2^{20}) is chosen, to give a sparse coverage of the overall space. This model relies upon the particular properties of high-dimensional binary spaces: for instance in a 2^{1000} space with 2^{20} actual locations an average of 98% of the locations will fall within a distance of 411-430 bits of a random location, with only an expected 0.01% of the points being a Hamming distance of less than 400 bits away. Kanerva's

storage scheme defines a hypersphere around each location: an input datum will cause activity in all locations in whose hypersphere it falls. Storing a pattern typically corresponds to incrementing and decrementing a set of counters, each corresponding to an input coordinate. For each coordinate of the datum that is "on", the corresponding counter is incremented, and a counter is decremented if its corresponding coordinate is "off". Reading from the memory is a similar process with the output being a reconstruction of the input obtained by applying a majority rule to each coordinate, using the locations in whose hyperspheres the input falls. This process may be iterated until a fixed point is reached, representing the reconstructed memory.

Keeler [92] showed how this may be implemented as a 3 layer network trained using the Hebb rule. The sparsely distributed location units may be regarded as the RBFs, defining a hypersphere using a Hamming distance metric. The Kanerva model, then, has a similar form of RBF to vector quantisation. However, it differs from vector quantisation in that recall involves an interpolation (described above) rather than a simple winner-take-all. Keeler demonstrated that the hidden-to-output weights are in fact determined via a Hebb rule:

$$w_{ij} = \mathbf{y}_i^T \mathbf{y}_j^H. \tag{2.37}$$

2.4.4 Restricted Coulomb energies

The restricted Coulomb energy (RCE) model [159] is another input dimensionality expansion method. Essentially, this is a method of modelling classes by mixture densities; the network architecture is similar to that of figure 2–3, except that the functions making up the mixtures are not constrained to be radial. Most applications of the RCE model have used square well (2.38) or triangular (2.39) potential functions:

$$\begin{aligned} \Phi(r) &= 0 \quad \text{if } r \leq \theta \\ \Phi(r) &= -b \quad \text{if } r > \theta \end{aligned} \tag{2.38}$$

$$\begin{aligned} \Phi(r) &= 0 & \text{if } r &\leq \theta \\ \Phi(r) &= r - b & \text{if } r &> \theta \end{aligned} \tag{2.39}$$

The potential well is restricted by the bias of the potential function (RBF), θ. Additionally the mixture parameters are set to unity, as are the priors. (Thus, neither of these sets of parameters meets the "sum to one" constraint; they should be considered as scaled uniform probabilities.)

This method differs from the method of potential functions in that a density function is constructed for each class; the hidden-to-output weight matrix does not specify a discriminative classifier. However, the training procedure is a discriminative, supervised learning method, with the position of the RBF centres being modified in the case of (i) no output class being activated; or (ii) a confusion between classes. In case (i) a new RBF is recruited to the set of RBFs corresponding to the class in question with it's centre being a scaled-up version of the input vector (i.e. $\mathbf{c}_i = \lambda \mathbf{x}$). In case (ii) the RBFs corresponding to the incorrectly activated classes are scaled down (i.e. λ is reduced) until they are at the threshold and the incorrect class is no longer active. Hence this heuristic procedure adjusts the RBFs of all classes involved in a confusion; this discriminative training procedure differs from the maximum likelihood approach (EM algorithm) used to compute the weights for mixture densities.

2.4.5 Other related techniques

The method of n-tuple recognition (or Boolean networks) [4] is another input expansion method. Each class has associated with it a set of n-bit random access memories (RAMs), each with 2^n locations. Each RAM is associated with an n-tuple drawn from the data; the set of n-tuples is usually selected randomly. The location in the appropriate RAM pointed to by the n-tuple is set to one. Training consists of simply repeating this procedure for all examples of all classes. Reading from the n-tuple recogniser is equally simple. The input datum, of unknown class, has

its set of n-tuples extracted and a logical AND is performed with the contents of the addressed location in that RAM of each class. The results of the AND are summed for each class, and classification is performed by a maximisation operator. This is not a discriminative pattern recognition system, as there is no interaction between classes. There is no obvious likelihood function (or similar) that is being maximised.

The self-organising feature maps of Kohonen [95] are dimensionality reduction methods: a high-dimensional input vector is mapped onto a low (usually two) dimensional space. This algorithm, based on work by Willshaw and von der Malsburg [211], uses a notion of conservation of topology: that is, input vectors that are somehow close in the input space will be close in the 2-space that they are mapped onto.

This method is somewhat similar to RBF methods: both rely on hidden units that correspond to a point in input space. The Kohonen method uses hyperspherical hidden units, coupled together to form a "winner-take-all" network, i.e. only a single unit is activated for each input point. However, the topology conserving aspects of this map complicate the issue as the units are regarded as being laterally connected, via an on-centre off-surround function. Hence the activation for each unit consists of the activation received from the input space, plus activation received from other units in the feature map[12]. In the case of no lateral connections, the Kohonen algorithm reduces to k-means clustering. As training progresses, the distance that locally excitatory connections extend is reduced. This leads to the initial discovery of coarse topological features, with the map being successively refined.

The memory-based reasoning (MBR) system [191] is essentially an RBF method. This is basically a very large table lookup system (with an entry for each data point), using various weighted error metrics for

12 In practice the lateral connections are not explicitly computed but approximated within the training algorithm by computing the winner from the input data alone (no lateral feedback) and updating only a set of feature map units within a certain radius of the winning unit. This radius is decreased as the training algorithm progresses.

operation on symbolic learning problems. The relationship to RBF methods with a basis function at each point is clear. This method also has similarities to the k-nearest neighbour method.

Omohundro [141] has suggested various computationally efficient algorithms, based on hierarchical data structures used in computational geometry, that are considerably more efficient than most neural network algorithms such as back-propagation. Omohundro's central data structure is the k-d (k-dimensional) tree. The k-d tree, is a method of structuring multi-dimensional data; at each branching of the tree, the data is partitioned at a certain point along a single co-ordinate. Hence, the leaves of the tree correspond to regions of the input space that have been partitioned according to the input data. This is implementationally similar to a radial basis functions network with non-overlapping hard-limiting hyperspherical RBFs, but computationally more efficient. The use of k-d trees to approximate hyperspheres has also been used in a computationally efficient way to compute the dimension of dynamical systems using ball-scaling methods [68] (see also chapter 6) (Omohundro, personal communication).

2.5 Summary

The RBF formalism has been shown to be a general way of describing several statistical and neural network pattern classification methods. The basic principal of these classifiers has been to transform the input data into a new (usually higher dimensional) space defined by a set of radially symmetric functions centred at points in the input space. The classifiers discussed in this chapter are summarised in table 2–1. Various ways have been used to choose the centres of the basis functions for these classifiers, both adaptive and non-adaptive. These are summarised in table 2–2. A principal division between the classification methods discussed has been whether they are designed to construct the most accurate model of each class or to minimise the misclassification between classes. The former

RBF classifiers	
Classifier	$\Phi(r)$
Approximated RBFs Locally Tuned Processing Units SPAN	$\exp\left(\frac{-r^2}{2\sigma}\right)$
Thin Plate Splines	$r^2 \ln r$
Multiquadratics	$(r^2 + c^2)^{1/2}$
Gaussian Classifier	$p(x\|c_i) = \frac{1}{(2\pi)^{d/2}\|\Sigma_i\|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\right)$
Parzen Windows	$\exp\left(\frac{-r^2}{2h^2}\right)$
Potential Functions	$\frac{\sigma^2}{\sigma^2+r^2}$
Kanerva Model	1 if $r \leq d$ 0 otherwise
Vector Quantisation	Hard-limited polygon
Restricted Coulomb Energies	0 if $r \leq d$ $x - b$ otherwise

Table 2–1: Table summarising various RBF classifiers discussed in this chapter; the form of Φ given is a typical form in the case of Parzen windows, potential functions and restricted Coulomb energies.

Centre positioning in RBFs		
Classifier	Adaptation	Choice of Centres
Approximated RBFs SPAN Potential Functions Parzen Windows	No	(Random) Data Points
Locally Tuned Processing Units Vector Quantisation	Yes	K-Means Clustering
Gaussian Classifier	No	Sample Means
Restricted Coulomb Energies	Yes	Supervised Discriminative Training
Mixture of Gaussians	Yes	EM algorithm

Table 2–2: Adaptive and non-adaptive means of choosing centres for RBFs.

Hidden-output weight training methods	
Classifier	Training Method
Approximated RBFs	Pseudoinverse $w^* = Y^T y^{H+}$
Locally Tuned Processing Units SPAN Potential Functions	LMS $\Delta w_{ij} = \eta(Y_i^T - y_i^T)y_j^H$
Gaussian Classifier Restricted Coulomb Energies	Prior Probabilities
Mixture of Gaussians	Priors and EM algorithm
Parzen Windows	Smoothing Parameter h
Vector Quantisation	Maximum $w_i = 1$ if $y_i^H = \max_j y_j^H$ $w_i = 0$ otherwise
Kanerva Model	Hebb Rule $w_{ij} = y_i^T y_j^H$

Table 2–3: A summary of the different training methods used to choose the hidden-output weights for various RBF classifiers.

criterion is instantiated in maximum likelihood training processes, the latter in discriminative training processes. Table 2–3 compares the training methods used for setting the weights of the linear RBF-output weight matrix. The contrast between maximum likelihood and discriminative training will remain an important theme throughout this thesis.

3

SPEECH RECOGNITION USING NEURAL NETWORKS

3.1 Continuous speech recognition

Whilst unconstrained speaker-independent continuous speech recognition is a task that people perform with apparent ease, it has proven to be difficult to program a computer to perform similarly. State of the art systems in continuous speech recognition [105, 34, 11] only perform well in artificially constrained tasks. Currently the best automatic speech recognisers are stochastic in nature, based upon hidden Markov models (HMMs). The best speaker independent system is probably Carnegie Mellon University's Sphinx [105] which has produced recognition errors of less than 4% on a difficult task using a 998 word lexicon and a constrained grammar (perplexity[1] 20). The IBM continuous speech recognition system [11], although not speaker independent (it used a set of 10 male speakers for both training and testing), featured a much less constrained statistical language model (training and testing sentences chosen from office correspondence) and a 5000 word vocabulary, producing average word recognition errors of 11%. BYBLOS, the BBN continuous speech recognition system [34] produced word errors of 1.5% in speaker dependent mode after training with 15 minutes of speech, using a 350 word lexicon

1 The perplexity of a grammar is the average number of words allowed after any other word.

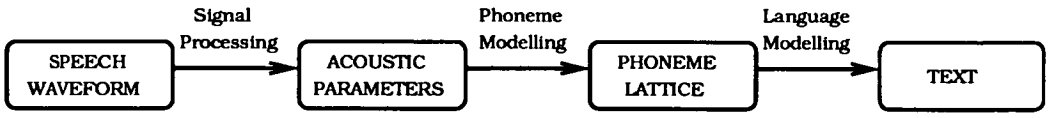


Figure 3–1: Simplified outline of a continuous speech recognition system. Three principal stages are identified: however, in many systems phoneme modelling and language modelling are coalesced into one stage, often via embedded hidden Markov models. In this case, a phoneme lattice is not explicitly computed; however, the relevant portion of the lattice must always be available to the language model.

and grammars ranging in perplexity from 30 to 60.

Figure 3–1 shows a basic outline of a continuous speech recognition system. Although most neural network approaches to speech recognition to date have concentrated on the area of phoneme modelling (and also isolated word recognition), it is by no means clear that this is the area in which neural networks have more to offer than other techniques. The article by Lippmann [114] reviews work in this field (performed up to and including 1988) comprehensively and Bourlard and Wellekens [23] provide a good overview of some initial work performed in relating time-delayed and recurrent neural networks to hidden Markov modelling.

3.1.1 Signal processing

The signal processing module of most speech recognition systems is based around a linear process such as linear predictive analysis (LPA) or the discrete Fourier transform (DFT). Problems with these processes include an information loss caused by the linear modelling[2] and also the need for a fixed frame rate, which does not take into account the different time scales of the speech signal: for example the closure, release and aspiration of a stop consonant occurs on a faster timescale than the onset, centre and offset of a vowel.

Most neural network methods for signal processing are based around

2 Although the existence of expert spectrogram readers indicates that a DFT analysis contains enough information to construct the spoken phoneme sequence with high accuracy.

models of the peripheral auditory system, derived from psychophysical experiments [182, 195, 147] or from what is known about the physiology of the inner ear [64, 120]. Comparisons between auditory model front ends and LPA or DFT front ends have given few indications that auditory modelling is superior to linear signal processing in noise-free conditions [161, 64, 19]; however there is strong evidence that auditory modelling is superior in noisy conditions [64].

An alternative approach to nonlinear signal processing is based upon nonlinear dynamics. Here, the speech signal is regarded as a time series produced by a nonlinear dynamical system. Consider the speech signal in terms of fluid dynamics, speech production being the modulation of airflow through the vocal tract. It is not unreasonable to assume that the output of this complex system will be of a high-dimensional character. However, although the vocal tract does not experience a laminar fluid flow, it is far from being turbulent. Recently, seemingly high-dimensional fluid systems close to turbulence have been studied using methods of nonlinear dynamics (e.g. [66, 113]). This research has indicated that such systems may in fact be accurately modelled using low-dimensional equations of motion with quasi-periodic or chaotic behaviour. Thus, we may hypothesise that the speech signal (or portions of it) may be modelled by some low-dimensional nonlinear system.

Certainly, the traditional linear signal processing view of the speech signal as a linear system plus stochastic noise is inadequate. For example, such a system is unable to generate the quasiperiodic waveforms that often make up vowels and other voiced parts of speech, whereas nonlinear systems naturally model such motion [116]. There is a growing and fruitful interaction between research in neural networks and nonlinear dynamics; it seems natural to use neural network architectures and algorithms in a dynamical systems approach to speech processing. Early attempts along these lines have recently been reported by Lowe and Webb [116] and Tishby [196].

3.1.2 Language modelling

Applying neural networks to language modelling in speech recognition is still at an early stage. Although some work has been performed in which neural networks were trained to learn elementary natural language tasks, e.g. simple grammars [49] and pronoun reference [5], it is not yet clear if these approaches will be useful for speech recognition. Recently, Jain and Waibel [86] presented a modular recurrent network architecture to perform on-line parsing of multiple clause sentences. This system however, does not deal with multiple word hypotheses and corrupted input, major issues in speech recognition. Lucke [119] has presented a scheme, based on a MLP, to perform lexical access, i.e. to learn a mapping from phonemes to words. Results on a small database were encouraging, however the system did not seem to scale well [51].

3.2 Phoneme modelling with neural networks

In this thesis we shall be principally concerned with phoneme modelling. The major problem in this stage of a continuous speech recognition system is the means by which the time-variability of the speech signal is represented. Neural network methods of phoneme modelling have been developed by increasing the sophistication of their temporal modelling of the speech signal.

The earliest crude methods neglected the temporal component of the speech signal entirely, with the problem being transformed into a static pattern classification problem. In these systems, any time-dependence was expressed spatially, e.g. by performing recognition using complete spectrograms as input. Consequently, these methods were only able to recognise small segments of an utterance (e.g. pre-segmented phonemes) or isolated words. However these systems were easy to engineer, as they could be mapped directly onto a static network which could be trained easily. Most of these approaches used feed-forward networks, generally multi-layer perceptrons. These systems are reviewed in section 3.4.

Static speech pattern classification systems avoid the issue of temporal processing, since the availability of segmented data assumes that there has been some earlier time-dependent processing. The addition of delay lines to these static systems imparted a limited amount of context-dependence into them. The resultant networks were also static, feed-forward networks (trained using identical algorithms) but the input layer was extended to cover several time frames of speech input. This allowed past (and future) context to influence the operation of the network. The use of delay lines was extended to hidden and output layers in the time-delay neural network (TDNN). The TDNN is a static feed-forward network; however, delay lines are added to each hidden and output unit, thus providing a memory of previous processing for the network. This has proven to be a successful architecture for speech recognition, allowing the network to learn time invariant features. Speech recognition systems implemented using delay-lined networks are discussed in section 3.5.

A more natural and powerful way of incorporating memory into neural networks is by the addition of recurrent connections: however, most recurrent networks are dynamic and learning algorithms for static networks are not applicable to them without some modification. A simple form of recurrent network is based upon a feed-forward architecture, with the input line extended to receive copies of the hidden or output units of the previous timestep. To enable training by back-propagation, the recursion is cut off after a single timestep in the backward error propagation network. Dynamic neural networks have also begun to be used for speech recognition by some researchers, using variants of the back-propagation algorithm. It seems likely that if speech recognition is to be tackled by the nonlinear modelling of speech dynamics, then the use of recurrent networks might be a powerful approach. However, current research is still at an early stage; phoneme modelling experiments using recurrent networks are reviewed in section 3.6.

Although dynamic networks offer a powerful way of modelling speech dynamics, their properties and training are still very much an open re-

search topic. The only attempts to use dynamic networks for large scale speech recognition have involved the extensive use of large supercomputers (e.g. [51]). Hence there is a motivation to employ static networks more effectively. Additionally, neural networks are usually discriminatively trained, which is not the case for many statistical models used for speech recognition. The combination of these two factors has led to the development of systems integrating neural network and statistical models — principally multi-layer perceptrons integrated with dynamic programming and hidden Markov modelling (outlined in section 3.3).

Static networks, even with delays, have been able to perform temporal modelling of the speech signal on a short timescale only — typically tens of milliseconds. It has proven difficult to engineer static networks which could model longer timescale aspects of speech, such as transforming continuous speech into a sequence of phonemes or words. To achieve this modelling, dynamic programming (or Viterbi) algorithms have been employed to post-process probabilistically interpreted network outputs (see section 3.7).

Rather than using static feed-forward networks as classifiers, they may also be used to model speech dynamics as prediction networks[3]. In this case the network is trained to predict the next frame of a time sequence. Section 3.8 reviews speech recognisers using such networks to model all or part of a speech unit and linked together using a Viterbi alignment procedure.

Such an approach starts to have strong links with statistical modelling techniques, particularly as prediction networks may be concatenated within a Markov chain. Recently, several workers have described how hidden Markov models (HMMs) may be implemented as recurrent neural networks and have described the relationships between the neural network cost functions and those used in HMMs (section 3.9).

Perhaps the most successful neural network speech recognition system

[3] A feed-forward prediction network may be related to a recurrent network with the same weight matrix, plus an extra connection from the predicted output to the next timestep's input. This recurrent network is a *generator*, rather than a predictor.

is the speaker-adaptive isolated word recogniser developed at Helsinki University of Technology [96]. This system performs phoneme modelling as each isolated word is modelled as a sequence of phonemes, rather than as a whole. The phoneme modelling portion of the system uses a self-organising feature map architecture (section 2.4.5) with units on the 2-dimensional map corresponding to particular classes. phoneme hypotheses are obtained via the active regions of the map at a given time. Lexical access is performed using a statistical grammar, trained from examples, that maps lattices of phoneme hypotheses to words. This PC-based system performs in real-time with errors of 2–4% when using a 1000 word Finnish or Japanese vocabulary.

3.3 Digression

Most state of the art continuous speech recognition systems are based around the methods of dynamic programming and hidden Markov modelling. Increasingly, these methods are being applied to produce better neural network speech recognisers. In this section the fundamentals of these techniques are briefly outlined.

3.3.1 Dynamic programming

The dynamic programming (DP) algorithm [20] is used to match an unknown input with a set of templates. If the time frames of the input are denoted by i and those of template k by j, then a local distance measure may be defined, $d_{k(i)}(i, j(i))$. The matching problem is finding a path through a sequence of grid points $(i, j(i))$ and templates $k(i)$ that minimises the total distance. That is, the best match is given by the sequence that minimises:

$$\sum_i (d_{k(i)}(i, j(i)) + T(j(i), j(i-1)))$$

where $T(j(i), j(i-1))$ is a time distortion penalty. When the DP algorithm deals with probabilities rather than distance scores, it is often referred

to as the Viterbi algorithm.

3.3.2 Hidden Markov modelling

Hidden Markov modelling is a stochastic approach to speech recognition that attempts to model speech units (e.g. phonemes or words) by a Markov chain [110, 152]. Such a model consists of a set of N states, with state transition probabilities a_{ij}. Associated with each state transition is an emission probability distribution[4], $b_{ij}(x_t)$, where x_t is an observation vector at time t[5]. A third set of parameters associated with a HMM are the initial state probabilities, c_i. In the case of a discrete HMM, the emission probability is a histogram over symbols corresponding to quantised vectors. In continuous parameter HMMs the emission probability is defined by a distribution over parameter space (typically a Gaussian or mixture of Gaussians). The forward probabilities give the joint probabilities for partial sequences in a HMM, w:

$$\alpha_{s_t}(t) = P(s_t, x_1^t | w), \tag{3.1}$$

where x_{t1}^{t2} notates the sequence $x_{t1}, \ldots, x_{t2}$ and s_t is the state of the model at time t. These may be efficiently computed using the recursion:

$$\begin{aligned} \alpha_i(0) &= c_i \\ \alpha_i(t) &= \sum_j \alpha_j(t-1) a_{ji} b_{ji}(x_t). \end{aligned} \tag{3.2}$$

The backward probabilities $\beta_i(t)$ give the probability of the rest of the sequence starting from state i at time t:

$$\beta_i(t) = P(x_{t+1}^T | i, w). \tag{3.3}$$

4 In some models the emission probabilities are associated with states rather than transitions. This is equivalent to tying all the emission probabilities corresponding to transitions from a particular state.

5 These probabilities should be properly represented as e.g. $b_{ij}(X = x_t)$, as x is an instantiation of random variable X. However the random variables have been omitted for clarity.

These may also be computed recursively, this time backwards in time:

$$\begin{aligned} \beta_i(t) &= \sum_j a_{ij} b_{ij}(x_{t+1}) \beta_j(t+1) \\ \beta_i(T) &= 1. \end{aligned} \tag{3.4}$$

And the probability of taking a transition from state i to state j at time t in a model w, given that the model generated the whole observation sequence is:

$$\gamma_{ij}(t) = \alpha_i(t-1) a_{ij} b_{ij}(x_t) \beta_j(t). \tag{3.5}$$

These probabilities are used to re-estimate the transition and emission probabilities in maximum likelihood training, using a particular formulation of the EM algorithm.

These equations for computing the forward and backward probabilities may be conveniently mapped onto a trellis, with each row of the trellis corresponding to a time frame. If we define:

$$w_{ij}(t) = a_{ij} b_{ij}(x_t), \tag{3.6}$$

then the trellis computation of the forward probabilities may be mapped onto the forward propagation of a feed-forward linear network, with each layer corresponding to a time frame. This network differs from a back-propagation through time network, since there is no tying of weights between layers. Likewise the computation of the backward probabilities also bears a resemblance to back-propagating down a feed-forward network, except the weight changes are multiplicative rather than additive.

Hidden Markov modelling has proven to be a powerful technique for speech recognition. Although it based on a false assumption (that speech is a first order Markov process) it is a tractable and theoretically solid probabilistic formalism. In speech recognition, embedded models are often used: for example, phoneme models may be concatenated to produce word models, which may be concatenated within a language model. An advantage of this technique is that, after initial bootstrapping to train the phoneme models, no transcribed (phonetically labelled and time-aligned)

data is required for further training, as alignments are automatically generated by the HMM training algorithm; all that is required is the word sequence. This is known as embedded training.

3.4 Static speech pattern classification

There have been many static speech pattern classification experiments reported in the literature. In this section the discussion will be limited to experiments performed with phonetic segments, rather than with isolated words[6]. In these experiments, the neural networks were static: that is, there were no delays or recurrent connections to enable time-dependent processing. The time-dependent nature of the speech patterns was extended spatially, so, in the case of the DFT front end that many of these studies used, the input to the network was a spectrogram quantised into time and frequency bins. These experiments required an initial segmentation of both training and testing data. This segmentation was generally carried out by a human phonetician. Many of these experiments used MLPs; other neural network architectures used include Kohonen's feature map, learning vector quantisation (LVQ) [95] and RBF networks.

Elman and Zipser [50] trained a MLP with a single hidden layer (of 2–6 nodes) to discriminate between 9 easily confused consonant-vowel (CV) pairs made up of the stops /b,d,g/ and the vowels /i,a,u/. The input spectrogram was quantised into 16 frequency bins (0–3.5 kHz) and 20 time slices (totaling 67 ms of speech). This speaker dependent recogniser produced errors of 1.5% for vowels and 7.9% for consonants using unseen test data.

Huang et al. [81] studied vowel classification using vowel formant data, consisting of the formant frequencies (F1 and F2)[7] of the midpoints of 10 vowels spoken in similar contexts (e.g. /hod/, /hud/) by men, women

6 There have been many isolated word recognition experiments, particularly digit recognition, e.g. [160, 149, 65].

7 The two formant frequencies F1 and F2 are the two lowest resonances of the vocal tract (above the fundamental frequency).

and children. The set of training data had no overlapping speakers with the test data. These experiments were performed using a MLP with 100 hidden units (19.8% error) and a feature map classifier (22.8% error), as well as various statistical classifiers, of which the k-nearest neighbour had the lowest error rate (18.0%). This work has been extended by Lee and Lippmann [106] and Ng and Lippmann (personal communication) using various other neural network classifiers (LVQ, RBFs and modified MLP training algorithms). This work has indicated LVQ to be the best performing neural network classifier, with an error rate slightly above that of the k-nearest neighbour classifier. The recent work by Lippmann and coworkers is also notable owing to its detailed comparisons between classifiers using such criteria as training time, memory usage and program complexity.

Leung and Zue [107, 108] and Cosi et al. [37] have used MLPs to perform speaker-independent vowel recognition; both sets of experimenters used an auditory model front end, based on that proposed by Seneff [182]. Cosi et al. used vowels (spoken by 13 speakers) that were excised from monosyllables. The input to the network consisted of 10 frames collected every 5 ms, with 40 coefficients per frame. A single hidden layer of 20 nodes was used. After testing on 7 new speakers, error rates of 5% were reported over 350 test cases. Leung and Zue [107] tackled a more difficult problem, classifying some 2,000 vowel tokens excised from 225 sentences of continuous speech spoken by 45 speakers (after training on 8,000 vowels spoken by 155 speakers). Using an input coding similar to that in section 4.1, the input consisted of 99 units, 33 coefficients representing auditory model output for the onset, centre and offset of each vowel. Error rates of 46% were reported in this 16 vowel classification problem; however when contextual information was added (in the form of extra units representing the phoneme label to the left and right) the error rate was reduced to 33%. This work has been extended [108] to cover all phonemes (a set of 39 were used) with error rates of 30%. More elaborate training methodologies and input representations were employed in

this work, and the vowel error rates were correspondingly reduced.

Niranjan and Fallside [138] compared RBF networks to a MLP, the modified Kanerva model and a k-nearest neighbour classification problem on speaker-independent vowel classification problem using the TIMIT database. 250 occurrences of each of 8 vowels were extracted from this continuous speech database, with 100 of each class used for training and 150 used for testing. The RBF network gave an error rate of 29.8%, lower than a MLP (35.4%) and the modified Kanerva model (33.9%), but higher than a nearest neighbour classifier (27.7%).

3.5 Introducing time delays

A static modelling of the speech signal is not a feasible way to attempt continuous speech recognition. The above techniques all assume a pre-segmentation and may be regarded as labelling a predetermined portion of a spectrogram. For continuous speech recognition a framewise processing of the speech signal is necessary. Such a modelling of speech dynamics requires some form of memory to be incorporated into the neural network model.

The simplest way of extending a feed-forward architecture to do this is via a time-delayed input buffer. This scheme, first introduced in the grapheme-to-phoneme system, NETtalk [181], is shown in figure 3–2. Here some context is provided to the input vector to be classified by extending the input to consist of several groups of units, each group representing the acoustic parameter vector for a particular time frame. This architecture has been used for labelling frames of continuous speech by several workers [24, 101, 203, 57, 127, 155, 51]; however all these workers used other techniques to increase the power of the temporal modelling.

Lang et al. [101, 100] extended the idea of delay-lines beyond the input layer. The time-delay neural network (TDNN) is a MLP incorporating delay-lined hidden and output layers. A schematic of such a network is shown in figure 3–3. A set of H hidden units time-delayed over T

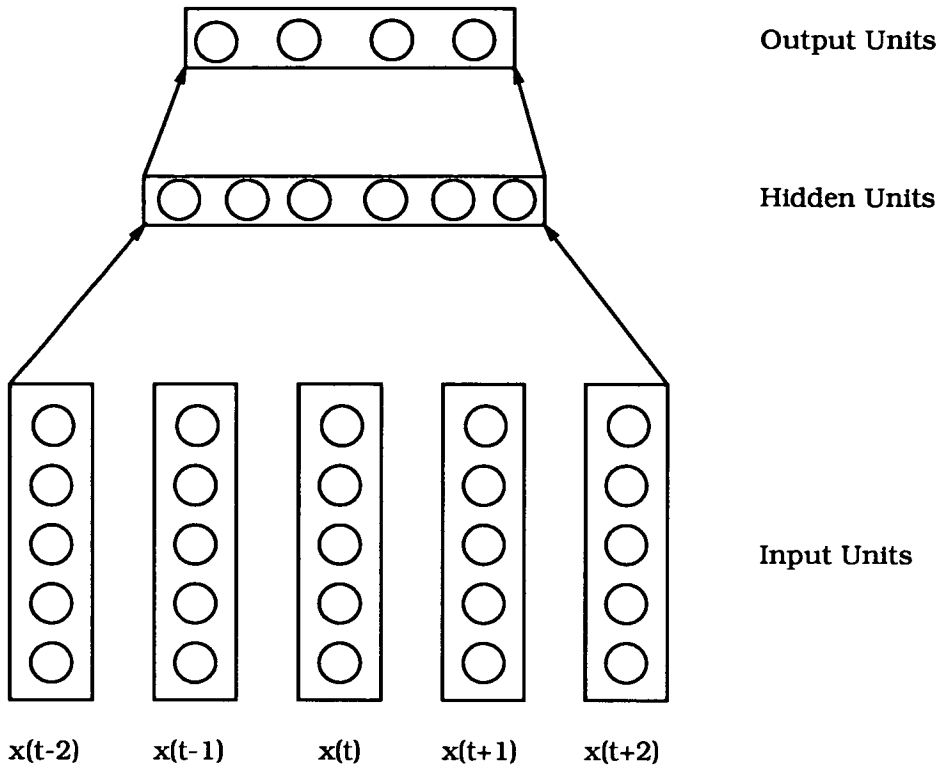


Figure 3–2: Sketch of the NETtalk architecture for a feed-forward network. The input layer is extended to include several frames of data, thus providing some contextual information. In this case the central frame x(t) has 2 frames of context to both the left and right.

timesteps, may be spatially unfolded, to give an equivalent network with HT hidden units, but with each set of T units having tied (constrained to be equal) weights. Alternatively, the network may be viewed in terms of delayed weights, in which each input unit is connected to each hidden unit with weights delayed by $1, 2, \ldots, T$ timesteps, with each set of time-delayed weights into a hidden unit being tied together. Hence, this network not only succeeds in modelling time in a more sophisticated way, it also lowers the number of free parameters in the network by weight tying. This is extremely important when there is limited training data, which is nearly always the case in speech recognition experiments.

This weight-tying leads to a translation invariance in the TDNN. A particular hidden unit has T sets of tied weights. This weight set may be regarded as being sensitive to a particular set of features (as determined

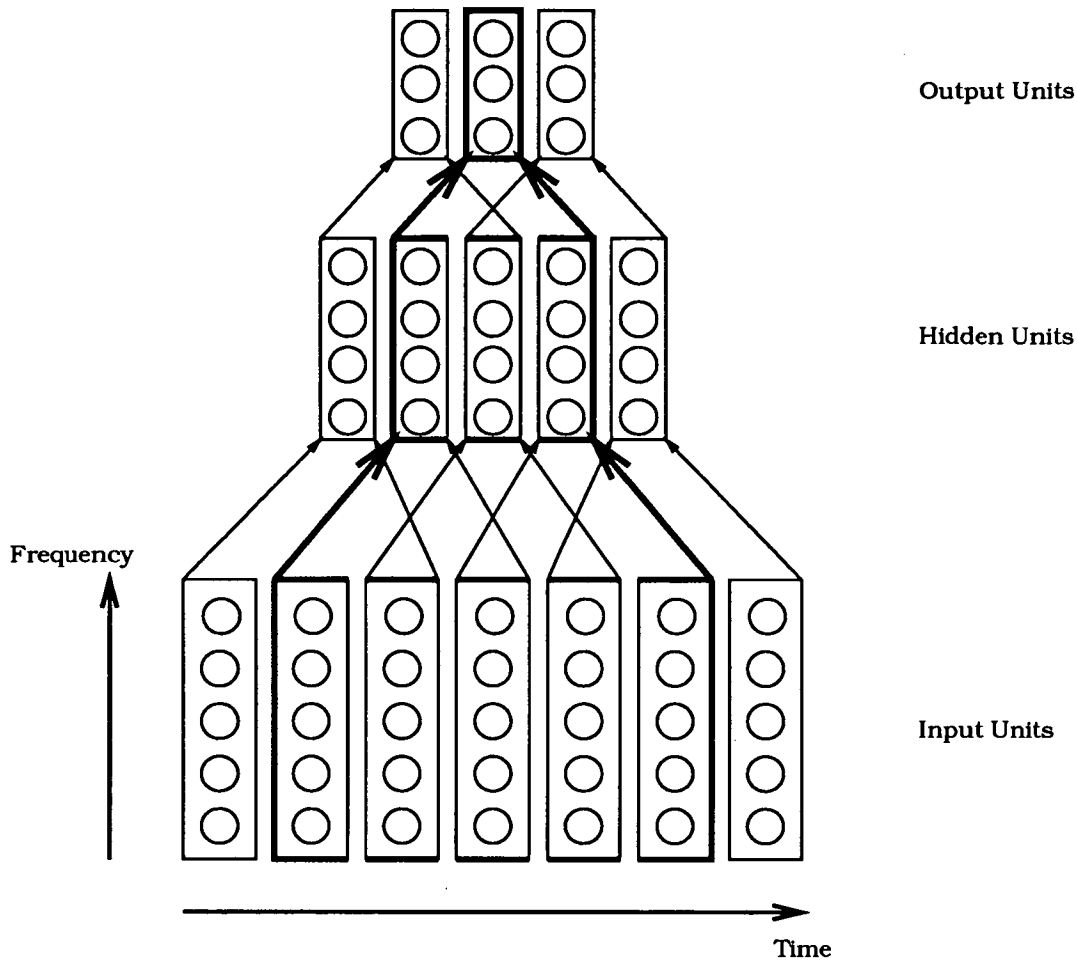


Figure 3–3: Schematic of a time-delay neural network (TDNN). In this feed-forward architecture the idea of using time-delays is extended from the input layer to the hidden and output layers. (More than one layer of time-delayed hidden units may be used.) In this diagram, the TDNN is unfolded in time. There are 60 free weights between the input and hidden layer (3×5 inputs into 4 hidden units) but 300 connections (the hidden layer is delayed over 4 additional timesteps). The final output is often retrieved by integrating over the output units; alternatively each set of delayed output units may be regarded as the output for a time t.

by the training). Since it is tied across several timesteps, the hidden unit will be sensitive to the same features over several timesteps. Thus the network may be trained to recognise a particular event, irrespective of the exact time it occurs in the input.

Lang [100] investigated the "E-set" problem, originally used by Brown [28], involving the discrimination between four letters of the alphabet "bee", "dee", "ee" and "vee". Each input token consisted of a 216ms piece of speech, containing the CV transition. This transition was not time aligned between tokens, so the translation invariance property of the network was crucial in the recognition. A 9.1% error rate was recorded on this task using a TDNN, which compared well with the discriminative continuous parameter HMM studied by Brown (11% error) and TANGORA, the IBM isolated word recognition system (maximum likelihood trained discrete HMM) which produced a 20% error rate on this task.

The TDNN has been extensively investigated by Waibel and coworkers [203, 204, 74] for various phoneme recognition tasks. These multispeaker experiments have used a Japanese speech database containing 6 speakers. Early experiments [203] reported error rates of 1.5% on a discrimination task between the phonemes /b,d,g/. This improved on the 6.3% error achieved by a discrete HMM on the same task. The TDNN architecture did not scale well to larger recognition problems with more output classes. The problem was tackled by training several subnets to perform discrimination between similar consonants (e.g. voiced stops, nasals) and other subnets to perform broad class recognition. These subnets were then "glued" together [204] to give a single network capable of discriminating between all phoneme classes occurring in the subnets (after further training). A network trained to recognise Japanese consonants in this way achieved an error rate of 4.1%, superior to the 7.3% error achieved using a HMM-based system. Hampshire [74] has extended the TDNN model to perform speaker adaptation. The basis of this extension is the "meta-pi" network which is a gated superstructure that is used to arbi-

trate between several TDNNs trained on the same discrimination task using different speakers. This has proved successful in decreasing the six speaker recognition error on the /b,d,g/ classification problem from 4.1% to 1.6%, which is the same performance as the speaker dependent task.

Another delay-lined network architecture was used by Franzini et al. [57] in a continuous speech recognition experiment using strings of digits from the TI/NBS database. Here 7 frames of input context were used and the hidden unit outputs were passed down a tapped delay line covering 10 timesteps. A second, undelayed, layer of hidden units was used and the 46 output units represented both phonemes and the current word. This large network architecture was trained incrementally, in order to make the training problem more tractable. Further temporal modelling was achieved by using the outputs of the network as the input to a dynamic programming system to transform framewise hypotheses into word and sentence hypotheses (see section 3.7).

McDermott and Katagiri [127] used the ideas of the TDNN in designing a time-delayed version of Kohonen's LVQ network: the input, hidden (VQ codebook) and output layers were all time-delayed in a similar fashion to the above work. Using a mel-scaled filter bank input with a test problem involving the discrimination between all 23 Japanese consonants resulted in an error rate of 2.3% — an improvement on the TDNN results mentioned above.

Prager has used a delay-lined input to the modified Kanerva model (section 2.4.2) in various continuous speech recognition tasks [155, 51]. This network had a very long input delay line covering 1s of speech (50 frames of 20 coefficients with a frame shift of 20ms). The output representation in this work was morph (small word) based rather than phoneme-based, with the input being mapped to the 1-from-n output every 20ms. On the small Cambridge "Hotel" database (with a lexicon of 133 words) errors of around 5% were reported on a multispeaker task using a network with 9600 location units (RBFs) [154]. This representation scaled up poorly to the larger TIMIT task, with a lexicon of around

1800 words. On a speaker independent task using 85 training sentences and 45 test sentences containing 118 different morphs a word recognition error of 33.8% was reported [51].

Prior to the work on TDNNs, Hopfield and Tank [193] presented an analog neural network architecture of similar structure. This time concentration network uses tapped delay lines and tuned filters to combine information over several time scales. Although there have been no substantial speech recognition results reported in the literature using this architecture, this is an interesting system since it has been implemented in analog electronic hardware using operational amplifiers, rather than simply being simulated on a digital computer.

3.6 Recurrent networks

Most recurrent network approaches to phoneme recognition have used modified MLPs; additionally most of these architectures featured some kind of delay line.

Rumelhart et al. [175] demonstrated that an arbitrary recurrent network may be unfolded in time, to produce a multi-layered feed-forward network, where each layer corresponds to a single timestep. In training, errors must be propagated back through time, with the usual back-propagation algorithm, the changes for each weight being averaged over time. This seems an infeasible way to train long time series (since the magnitudes of the gradients decrease as more layers are back-propagated through, thus making learning long distance temporal dependencies unlikely), but Robinson and Fallside [165] have used this network in a single-speaker continuous speech recognition problem, using a perceptually-scaled filter bank as a front end and having recurrent output-to-input connections. After training on 2 utterances of each of 31 sentences by a single speaker, test results on 2 further utterances of the sentences by the same speaker resulted in a framewise error rate of 21.9%, after smoothing using dynamic programming, The error rate using a k-nearest

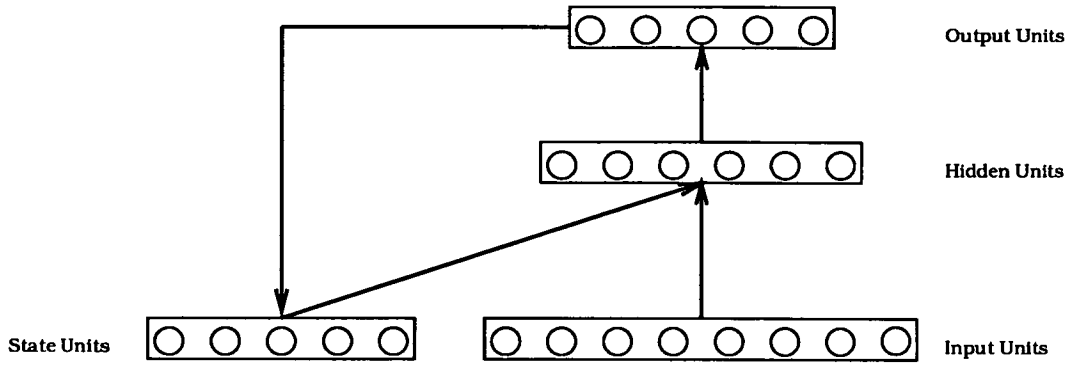


Figure 3–4: Schematic diagram of the recurrent network proposed by Jordan. The output units are fed back to an extended input layer (state units).

neighbour classifier was 51.9% on this task. On a similar problem, but training and testing on 6 male speakers, the error was 29.2%. These experiments used a subset of 27 phonemes. More recently, Robinson and Fallside [167] have applied this network to the TIMIT database; this was a speaker-independent continuous speech recognition task. Using a similar front-end and dynamic programming back end to perform segmentation, segmental error rates (over 61 phonemes) of 25.1% substitution, 6.3% deletion and 6.1% insertion were reported. This compares well to the best HMM results on the TIMIT database using phoneme modelling [105]. However, much superior HMM results (at a word and sentence level) have been achieved using techniques such as generalised triphone modelling and embedded training methods.

Simple recurrent MLPs were introduced by Jordan [88] and Elman [49]. Jordan introduced simple recurrent connections from the output units to a set of additional units ("state units") (figure 3–4). This recurrence enables a discrimination to be made at a given timestep using information about discriminations at previous timesteps, something that is not possible with non-recurrent networks. Elman proposed a similar scheme, with the hidden units recurrently connected to the state units (figure 3–5). Both these networks may be trained by back-propagation, with the assumption that the recursion in the backward network, is ended after a single iteration. The derivatives ($\frac{\partial E}{\partial w_{ij}}$) computed by the backward

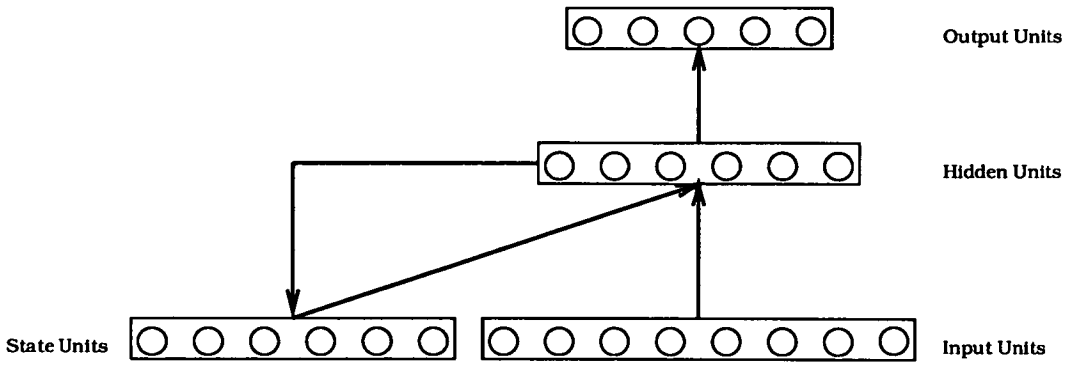


Figure 3–5: Schematic diagram of the recurrent network proposed by Elman. In this network, the hidden units are fed back to an extended input layer (state units).

network will thus be only an approximation to the true derivatives.

Anderson et al. [6] applied Jordan's network architecture to a speaker independent problem, in which the stop consonants /b,d,g,p,t,k/ where discriminated in a similar vowel environment. The inputs were perceptually scaled power spectra with each CV syllable being represented by 25 frames, using a 5 ms frame shift. Several network architectures, with 1 or 2 hidden layers and varying numbers of units in each layer, were experimented with and the best results produced an error of 13% after training on 10 speakers and testing on another 10 speakers.

Franzini et al. [56] extended the delayed model of the previous section to incorporate the hidden-to-input recursion suggested by Elman. In this network the input layer (a sliding window of 7 frames of acoustic parameters) was extended to include the hidden unit outputs for the previous 10 timesteps. As in Elman's formulation the hidden-to-input recursion is cut off after a single timestep to allow training by back-propagation. Here the training of the MLP was embedded in a Viterbi algorithm and the mapping was from input acoustic data to HMM emission probabilities (see section 3.7).

Watrous [206] has performed many experiments using a recurrent architecture he terms the "temporal flow model". In this network, each hidden unit has a self-recurrent connection, thus imparting some mem-

ory of previous events into the network. Since the only recursion in the network is from each hidden unit to itself, there are no multi-unit loops in the network; it may be shown that the usual back-propagation algorithm may be used with only a trivial modification. Watrous's experiments incorporated a good deal of speech-specific knowledge, and concentrated on learning phonetic discriminants such as place, manner and voicing. These single speaker experiments produced very low error rates, of the order of 0.5%. It is not clear how Watrous's techniques will scale to larger speech recognition problems.

The temporal flow model was used by Aktas et al. [3] in a speaker-dependent continuous speech recognition experiment in which each frame was classified into one of 7 broad classes. The German speech database SPICOS was analysed, producing 16 cepstral coefficients per 20ms frame, using a frame shift of 10ms. The temporal flow model produced a frame-wise error rate of 14.8%, somewhat better than a maximum likelihood histogram classifier (25.0% error) and context-independent discrete HMM (17.0% error), but poorer than a context dependent discrete HMM (11.0% error).

Robinson [164] noted that back-propagation of error through time was a linear process and could be compressed into a single operation. The relevant gradient may be written as:

$$\frac{\partial y_i(t+1)}{\partial w_{kl}} = f'(\sum_j w_{ij} y_j) \left(\sum_j w_{ij} \frac{\partial y_i(t)}{\partial w_{ij}} + \delta_{ki} y_l(t) \right). \tag{3.7}$$

However this algorithm is extremely expensive (memory usage is $\mathcal{O}(W^2)$, where there are W independent weights in the network), hence it was not used in speech recognition experiments. This algorithm was independently proposed by Kuhn et al. [97][8] who have applied it to the same "E-set" problem as attempted by Lang using TDNNs (section 3.5). Their initial results are somewhat poorer than those achieved by Lang (and also

8 This algorithm has been rediscovered independently many times recently; other authors who have independently proposed the algorithm include Williams and Zipser [210], Gherrity [63] and Gori et al. [67].

Brown's HMM results) with an error of 15.4%.

3.7 Modelling time by dynamic programming

Training recurrent networks is still an open research problem, and it seems unlikely that delayed networks alone will be powerful enough for continuous speech recognition, so several workers have studied systems in which discriminative neural networks are integrated with a dynamic programming algorithm [24, 176, 57, 56, 21]. Additionally, Lippmann and Gold have shown [65] that the Viterbi decoder used in hidden Markov modelling may be implemented as a neural network.

Early work by Bourlard and Wellekens [24] used a MLP with a delay line input of 7 frames to produce framewise phoneme scores. A sparse input was used, the result of a vector quantisation procedure, with each frame being represented by 132 binary units, only one of which was active. In this connected word recognition problem, each item of the vocabulary was described by a concatenation of phoneme models. Word recognition was performed using the DP algorithm to compute a matching score with each word model. Using the single speaker German database SPICOS, a word error rate of 37% on a 918 word lexicon was achieved. This compares well with the results achieved using a (somewhat unsophisticated) HMM, which produced a 48% error rate on the same task. More recently Bourlard and Wellekens [22] have demonstrated that the outputs of a MLP may be regarded as good estimates of the Bayesian posterior probabilities; these may be used to train a HMM discriminatively (see section 3.9). Bourlard and Morgan [21] extended this DP approach by modelling each phoneme as a PDF repeated D/2 times, where D was a prior estimate of the average duration for that phoneme. This was a left to right Markov chain with only sequential or self-recurrent transitions allowed. The output probabilities of the MLP (when scaled by estimates of the phoneme prior probabilities) were used as emission probabilities in this discrete HMM. In this work the best results were achieved using

a 9 frame delayed input and no hidden units, giving a 34.4% error rate.

A similar approach was adopted by Franzini et al. [57, 56]. Using the partially recurrent Elman network, described in section 3.6, the outputs produced by the neural network were interpreted as probabilities. In this network architecture there were 46 outputs, 12 corresponding to words (11 digits + silence) and 34 corresponding to phonemes. The network outputs were used to determine the probabilities in a HMM consisting of word models made up of concatenated phone models and a sentence model made up of seven (the maximum number of words in a sentence) concatenated word models. The emission probabilities were obtained by linearly combining the relevant phoneme and word outputs produced by the network. Duration was modelled using transition probabilities. The models were left to right models, with no self-recurrences until the state previous to the final state. Thus all transition probabilities except those for the only self-recurrence and the transition to the final state set to unity. On the TI/NBS digit recognition task this system produced 3% word errors and 9% sentence errors.

This approach was extended by integrating the DP and neural network aspects of the system to be used within a HMM. In this approach, "connectionist Viterbi training" (CVT), phoneme models were initialised using maximum likelihood HMM training with vector quantised input [105]. The vector quantisation step was then discarded, to be replaced by the neural network. The emission probabilities for each transition, previously obtained using a histogram (with each bin corresponding to the probability of a VQ codeword being generated by that transition), were computed by a network with one output for each emission probability. Each output was regarded as the emission probability of a particular transition given the acoustic input. By doing this it was hoped to lower distortion errors caused by the vector quantisation; this is a somewhat similar approach to semi-continuous hidden Markov modelling [82] (see section 4.5.3.2). After training the network to map from acoustic input to emission probabilities, the training data was Viterbi aligned with the

new HMMs. The network was then retrained on the new acoustic parameter to output probability mappings, and the transition probabilities were re-estimated using the new alignment. This process was iterated. The stopping criterion depended on a third set of data (in addition to the training and testing sets) which was used as a cross-validation set; training was completed when there was no improvement in this set. On the TI/NBS digits task the word error rate was improved to 1.5% and the sentence error rate to 3.0% using this model. These results do not surpass the best HMM efforts on this problem, but this is clearly a competitive technique. It also seems likely that this basic technique should be extensible to larger vocabulary recognition.

Miyatake et al. [132] also applied dynamic programming to the output of a feed-forward network. The network used in this work was a large TDNN with 4 hidden layers and 24 outputs corresponding to the 24 Japanese phonemes. The first 2 hidden layers were effectively 9 separate modules, corresponding to 9 broad classes of phonemes. The output of this network was interpreted probabilistically and processed using a DP algorithm before passing the resultant phoneme lattice to a predictive LR parser[9] that was trained to perform lexical access. This system was tested using a lexicon of 5240 Japanese words, spoken individually, giving a speaker-dependent word error of 8%.

Sakoe et al. [176] introduced a "dynamic programming neural network" (DNN). In this system each word model is represented by an individual feed-forward network. Here the DP alignment is applied to the network input rather than the output; the maximisation problem that is solved by the DP alignment is then to maximise the output of the network, given a particular input. Two training methods were suggested. In the fixed alignment procedure the network is trained by back-propagation after the time-alignment has been performed. In adaptive time alignment, patterns that do not correspond to the word model (desired output of 0)

9 An LR parser is a deterministic, bottom-up parser used to parse context-free grammars.

are realigned each time they appear; since alignment is a maximisation procedure, this has the effect of transforming these patterns into their most "non-separable" alignment, thus increasing the efficiency of learning. This network was applied to a speaker-independent Japanese digit recognition problem giving error rates as low as 0.7%.

Lippmann and Gold [65] described how a Viterbi decoder may be implemented as a neural network. The Viterbi net is basically a "neural" implementation of a left-to-right HMM (with self-recurrences) with each state modelled by a single Gaussian. Each state of the HMM consists of a summing node that implements the quadratic decision surface and a delay node that also performs a thresholding operation. Additionally, there are small subnets that compute the maximum of the output of the previous state and the output of the current state. A sequence of vectors is presented to the net and activity proceeds from left to right across the "HMM state nodes". The final output is the probability of the most likely state sequence. This neural Viterbi decoder differs from the usual Viterbi algorithm; however experiments using a HMM isolated word recogniser have indicated that its performance is almost identical. This network is most interesting because it demonstrates a way in which this algorithm might be implemented in analog VLSI.

3.8 Prediction

In the work discussed so far in this chapter, a major motivation for using neural networks has been their discriminative power, which seems likely to produce better classifiers than the maximum likelihood methods often used to train hidden Markov models. Feed-forward networks may also be regarded as functional interpolators (see chapter 2) and have proven to be successful predictors of chaotic time series [30, 102, 133]; it seems reasonable to use them as predictors of a speech signal. This would be a potentially powerful approach to speech recognition, similar to hidden Markov modelling. For example, a set of prediction networks could be

trained for each word model, with a DP match being used to choose the word hypothesis. It is easy to see how such a system could be extended to continuous speech recognition using phoneme models and an embedded training method. The advantage of using feed-forward networks as predictors is that each network is engineered to learn the dynamics of a particular unit of speech. Additionally, predictive networks may be treated as HMMs or states of HMMs, thus allowing the well-understood methods used for HMM training and recognition to be applied.

Such systems have been recently proposed by Levin [109], Iso and Watanabe [85] and Tebelskis and Waibel [194]. The former two papers concentrate on isolated word recognition, but it is easy to extend them to perform continuous speech recognition.

Levin's hidden control neural architecture [109] consists of a predictive MLP, with the acoustic input $\mathbf{x}(t)$ augmented by a control signal $\mathbf{c}(t)$. The network is trained to predict $\mathbf{x}(t+1)$, with $\mathbf{c}(t)$ acting as a modulator. The control signal is hidden: it is produced by a second network, in this case a left to right finite state machine. To train the system, a joint minimisation of the prediction network weights and the hidden control state sequence is required. The former is performed using back-propagation, the latter using the Viterbi algorithm. This system has been applied to multi-speaker connected digit recognition (using a portion of the TIMIT digits database) giving errors of 0.7%, which is competitive with the best HMM results.

Iso and Watanabe have proposed a somewhat similar system [85]. Here, a word model is a left to right Markov chain, with self-recurrence, with each state corresponding to a MLP predictor. Training is somewhat similar to the connectionist Viterbi training of Franzini et al. [56] (see section 3.7). After initialisation of the MLP weights the prediction residual is computed using a DP match, and the weights for each MLP along the optimal trajectory are re-estimated using back-propagation. This procedure is iterated over all the training data. On the same speaker-independent digit recognition problem as used by Sakoe et al. [176] (see section 3.7),

the error rate was decreased from 0.7% to 0.2%.

Tebelskis and Waibel [194] independently developed an almost identical scheme to Iso's neural prediction model. This system was applied to phoneme modelling, rather than word modelling, with each phoneme represented by a left to right sequence of 3 predictive networks. Word models were constructed by concatenating phoneme models. On a simple speaker-dependent test using short words and a subset of 13 phonemes, a 10% error rate was reported on a 924 word vocabulary, with 6% on a 234 word vocabulary.

3.9 Connections with stochastic modelling

The above approaches make some links between hidden Markov models and neural networks, but they do not provide a clear insight into the exact relationship between HMMs and neural networks. Recently some researchers [84, 93, 22, 26, 25, 213, 137] have demonstrated how a HMM may be implemented as a recurrent neural network, trained using some variant of back-propagation.

Two early approaches demonstrate how a HMM trained using MLE may be implemented as a recurrent back-propagation network. Hwang et al. [84], who were primarily interested in systolic array implementations, observed that the trellis formulation for computing the forward and backward probabilities in a HMM may be simply mapped onto a recurrent network. Using the back-propagation through time training method [175] they demonstrated that the forward and backward passes correspond to the forward- and back-propagations in such a network (in which each layer corresponds to a single timestep). As pointed out in section 3.3.2 the backward propagation in the forward-backward algorithm for HMMs is similar to a multiplicative gradient descent. Additionally, it also meets the constraints that probabilities must be non-negative and sum to 1. Kehagias [93] also demonstrated how the Baum-Welch algorithm could be implemented by a feed-forward network, trained using back-propagation

through time. Kehagias employed the usual additive gradient descent in the back-propagation and implemented the constraints by a change of variables. Consider the constraint $\sum_i c_i = 1$; then auxiliary variables, s_j, were defined such that $c_i = \frac{s_i^2}{\sum_j s_j^2}$. Kehagias also offered a common theoretical basis for the Baum-Welch algorithm and back-propagation, by placing both in an optimal control framework.

A principal reason for the interest of speech recognition researchers in neural network models is that they are naturally discriminative. A feed-forward network (when used for classification) is trained to discriminate between several classes. This contrasts with maximum likelihood estimation (MLE) methods (often used to train HMMs) which are not discriminative; a model of a particular class is trained only on data known to be from that class and is kept isolated from data from other classes.

Consider training a HMM given a set of acoustic evidence x_1^T. In MLE, the probability, $P(x_1^T|w)$ that the model w has produced the acoustic data is maximised. This may be accomplished using Bayes' Rule:

$$p(w|x_1^T) = \frac{P(x_1^T|w)P(w)}{P(x_1^T)}. \tag{3.8}$$

$P(x_1^T)$ is constant for all models and $P(w)$ is given by the language model. Thus an estimate of $P(w|x_1^T)$ is is maximised.

MLE is the correct procedure to adopt if it is known that the correct model exists in the space of models under investigation. However, this is not the case for hidden Markov modelling of speech. In this case, the function that should be maximised is the mutual information between the acoustic evidence x_1^T and the word sequence w [12, 28]; maximising this function allows the system to obtain as much information as possible about the word sequence from the acoustic evidence. (Maximum mutual information estimation (MMIE) is equivalent to MLE when the space of models includes the correct model.) The mutual information is given by:

$$I(x_1^T, w) = P(x_1^T, w) \log \left(\frac{P(x_1^T, w)}{P(w)P(x_1^T)} \right). \tag{3.9}$$

Since $P(x_1^T, w)$ is not known, the function that is usually maximised is:

$$\begin{aligned} f(x_1^T, w) &= \log\left(\frac{P(x_1^T, w)}{P(w)P(x_1^T)}\right) \\ &= \log(P(x_1^T|w)) - \log(x_1^T) \\ &= \log(P(x_1^T|w)) - \log\sum_{w'} P(x_1^T|w')P(w'). \end{aligned} \tag{3.10}$$

The first term on the right of this expression is what is maximised when MLE is used. The second term may be regarded as the discriminative term.

MMIE is usually maximised by gradient ascent, since no provably convergent training method (such as the EM algorithm for MLE) is known. When the partial derivatives of (3.10) are taken with respect to the parameters of the models, the resultant expression has two terms [28]. The first term is in the direction of the derivative of the MLE cost function with respect to the parameters (notated $Q(w, \theta)$). The second term, summed over all models other than the correct one, acts to subtract a component in the direction of $Q(w', \theta)$ for all incorrect word sequences w'.

Bourlard and Wellekens [22] defined a particular discriminant HMM, using discriminant emission and transition probabilities trained using a Viterbi method. It was demonstrated that the local (combined transition and emission) probabilities in this model were equivalent to the probabilities computed by a MLP with output-to-input feedback (a network proposed by Jordan [88] and discussed above).

Tighter links between HMMs and neural networks have been made more recently by defining networks with cost functions whose minimisation is equivalent to MMI training of discriminant HMMs. As an alternative to using a least mean square cost function, a cost function based on the Kullback-Liebler divergence between two distributions P and Q may be used:

$$G(Q|P) = -\sum_{w} P(w|x)\log\frac{P(w|x)}{Q_w(x)}. \tag{3.11}$$

P is the true distribution and Q is the output of the network. An error metric based on this was proposed by Solla et al. [188] and termed the cross-entropy. Bridle [26] simplified this metric for the case of a 1-from-n output coding:

$$J = -\sum_t \log Q_{w_t}(x_t), \tag{3.12}$$

where w_t is the correct word class for input t. J is the negative log probability of the network choosing the correct output class. It was demonstrated that minimising this criterion is equivalent to maximising the mutual information. In a companion paper [25], Bridle defined a recurrent network, similar to those defined above, that implemented the propagation of forward probabilities in a HMM. The requisite partial derivatives required to train such an "alphanet" by back-propagation using the cost function defined above were derived[10]. In this method the parameters of the true model are always re-estimated in the direction of the Baum-Welch method; however parameters of the other models are also re-estimated in a different direction to their Baum-Welch re-estimates for that utterance.

Similar work has also been presented by Niles and Silverman [137] and Young [213]. The principal innovation offered by Niles and Silverman is the relaxing of the non-negativity constraint for the transition and emission probabilities. Whilst this weakens any probabilistic interpretation of the model, it does correspond to a neural network with inhibitory interactions[11]. Young studied a single composite HMM, capable of recognising all phonemes. That is, all the phoneme models were connected

10 A "softmax" transformation was used to define auxiliary variables which may be minimised in an unconstrained fashion:

$$c_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}.$$

This is similar to the transformation used by Kehagias (above).

11 Niles and Silverman draw analogies with quantum mechanics to support the use of negative (and complex-valued) probabilities. They argue that inhibition in neural networks is similar to interference in quantum mechanics

together to form a network of models. Since this was a composite model, maximising the probability of occupation of one state implied minimising the occupancy of all other states. Hence discriminative, or competitive, training is natural in this system. Here the forward computation was represented as a linear neural network going forward through time. The back-propagation training involved the maximisation of the posterior probability of state occupation, which is equal to the normalised forward probability. This network was tested on a speaker-dependent continuous speech experiment, using 23 combined 3 state phoneme models (plus an entry and exit state) resulting in a 71 state composite model. The error rate decreased from 47% when using a MLE criterion in training to 41% when using competitive training.

3.10 Summary

In this chapter we have reviewed the development of phoneme modelling neural networks from static networks classifying presegmented speech units, through delayed networks performing a limited amount of temporal processing to recognise individual speech units to techniques integrating neural networks and statistical methods. Additionally methods of more adequately modelling speech dynamics using recurrent networks and static prediction networks have been reviewed.

Whilst dynamic networks remain a promising way of better modelling speech dynamics, there have been few demonstrations of their efficacy in a speech recognition system, as yet. This is due to a twofold ignorance:

- an ignorance of the behaviour of even simple dynamic network models;
- an ignorance of powerful learning algorithms for dynamic networks.

Chapters 5 and 6 go some way toward extending our knowledge of the dynamical behaviour of a simple recurrent network model. An alternative way of modelling speech dynamics, motivated by the successes of using

feed-forward networks to predict nonlinear dynamical systems, was reviewed in section 3.8. Here neural networks were used to predict, rather than classify, a speech signal.

Additionally, it was shown how both prediction and classification networks can be profitably post-processed using a Viterbi algorithm and a first-order Markov assumption. The systems of linked prediction networks, in particular, demonstrated a close affinity to HMM systems, with a single prediction network being analogous to a state of a HMM. The connections between HMMs and neural network models were more strongly made in section 3.9. where work was reviewed which demonstrated that HMMs could be implemented as neural network models. This is more than trivial isomorphism: describing HMMs as neural networks may enable the development of better discriminative training algorithms as well as encouraging a more rigorous, probabilistic understanding of neural networks.

4

SPEECH RECOGNITION EXPERIMENTS USING RADIAL BASIS FUNCTIONS

In this chapter the results of applying RBF networks to various speech recognition problems are reported. Two classes of problem were investigated:

- labelling phoneme segments pre-segmented from continuous speech;
- constructing phoneme models for continuous speech using RBF networks.

All the work in this chapter used data recorded from a single male speaker.

The first problem was reduced to a static pattern classification problem, and so was far removed from the principal difficulties of automatic speech recognition. To perform continuous speech recognition, the temporality of the speech signal must be modelled. This is not naturally achieved by the simple RBF network. Some interim approaches to this problem were used here, involving modifications to the RBF model and a separate post-processing stage.

4.1 Classifying vowels

The task reported here was to label vowel tokens, hand-segmented from continuous speech. Since each vowel was represented as a "snapshot" of speech, an intrinsically dynamic problem was transformed into a static pattern classification problem.

4.1.1 Speech data and signal processing

The vowel tokens were segmented by a trained phonetician from a phonemically dense speech database of 98 sentences [83][1]. This set of sentences was uttered twice by a male RP[2] speaker, once for training and once for testing. Each set consisted of approximately 750 vowel tokens. The vowels were divided into 20 classes — 12 monophthongs and 8 diphthongs. After digitisation at 16kHz and pre-emphasis using a filter[3] with a transform function of $1 - 0.98z^{-1}$, one of two analyses was carried out on the speech data (in each case the analysis was performed using a 20 ms Hanning window[4] with a frame shift of 5 ms):

- a 20th order linear predictive analysis [126], from which 20 cepstral coefficients describing the linear predictive filter were produced[5];
- a discrete Fourier transform producing a power spectrum that was formant tracked (using the method of global optimisation of generalised spectral centroids [39]), producing frequency, bandwidth and amplitude information for each of the first three formants.

1 Examples of the sentences in this database include "Our lawyer will allow your rule" and "Patty picked up a potato cake.".

2 RP stands for Received Pronunciation, which a standard British English dialect ("BBC English").

3 This is a high pass filter to model radiation of non-nasalised sounds at the lip boundary[126].

4 The Hanning window is obtained as a weighted sum of the rectangular window and shifted versions of the rectangular window (see [163], chapter 6).

5 The cepstral coefficients are the Fourier coefficients of the logarithm of the transfer function of the filter. They may be computed by a convenient recursion from the predictor coefficients [126].

4.1.2 Data representation

Each vowel token was regarded as a static pattern. In an attempt to impart some of the dynamical properties of a vowel into this representation a convenient approximation was made following Dalby et al. [42][6]. A vowel was considered as being split into three parts of equal duration, corresponding to the onset, centre (steady state) and offset. In this representation, the centre was regarded as containing the typical formant pattern for a vowel, whilst the onset and offset both provided a limited amount of contextual information. Whilst this was an acceptable approximation for monophthongs (which often achieve a steady state), it was inadequate for diphthongs which are characterised as a trajectory between two articulatory positions[7].

For the cepstral analysis a feature vector was constructed consisting of the 20 cepstral coefficients averaged over each third of the token, plus the duration, which was coarse coded over 12 real numbers valued from 0–1, giving a 72 element feature vector. When the formant analysis was employed a 15 element feature vector was constructed consisting of 9 median formant frequencies (the median frequency of each of the first 3 formants in each third of the utterance), the median bandwidths of each of the 3 formants in the centre portion of the utterance, 2 amplitude ratios from the central third (median(A2)/median(A1) and median(A3)/median(A2)) and the overall duration of the utterance. The formant frequencies were expressed using both a linear frequency scale and the nonlinear Bark frequency scale[8].

6 This representation was also employed by Leung and Zue [107]

7 Vowels may be classed as monophthongs or diphthongs. A monophthong is a vowel for which there is no appreciable change in quality during the segment, e.g. /e/ in "bed". A diphthong is a vowel in which there is a change in quality (i.e. the tongue moves) during a segment, e.g. /ai/ in "fly".

8 The Bark scaling of frequencies is a nonlinear transformation designed to model the response of the ear to incoming sounds [214]. Derived from experimental data, the

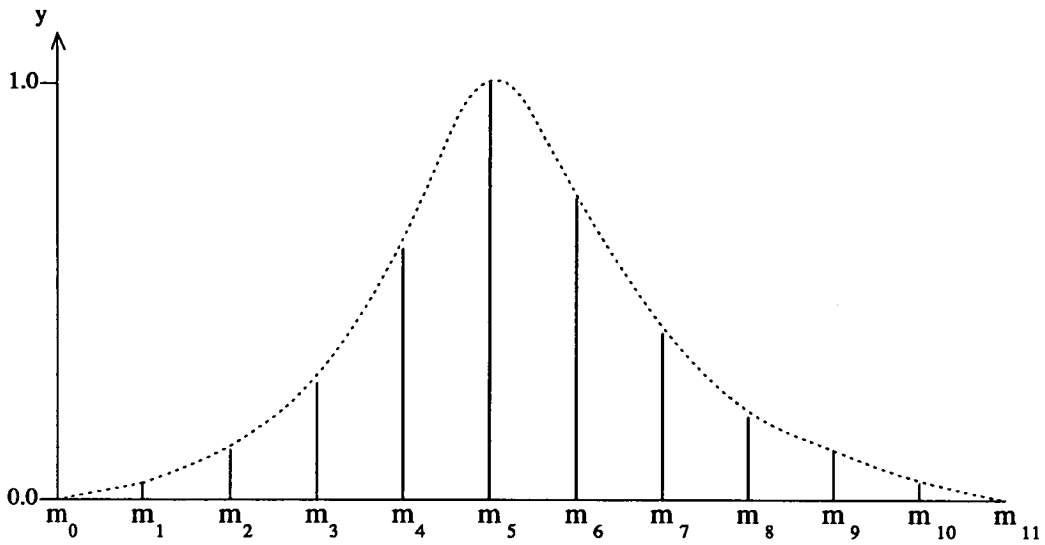


Figure 4–1: An example of coarse coding a real number onto 12 units (centred at real numbers m_0 to m_{11}) with output values, y, ranging from 0–1. In this case the number being coarse coded lies between m_5 and m_6.

4.1.2.1 Coarse coding

Feed-forward networks are sensitive to inputs with a large dynamic range: in the case of MLPs, inputs much beyond the bounds of the transfer function can cause the development of very large (or very small) weights leading to instability in learning. Additionally feed-forward networks are known to be poor at performing analog-to-digital conversion [177]. A solution to this would be a method in which an arbitrary real number could be coded over a group of units taking scalar values bounded by the bounds of the transfer function (in this case 0 and 1). In this work a Gaussian coarse coding method was used (figure 4–1). To determine the clamped input values y_i^I when coarse coding $x \in \mathbf{R}$ onto N input units

following curve gives a good fit to the experimental curve [182]:

$$B(f) = \begin{cases} 0.01f & 0 \leq f < 500 \\ 0.007f + 1.5 & 500 \leq f < 1220 \\ 6\ln f - 32.6 & 1220 \leq f \end{cases}$$

where f is the frequency in Hz and B is the frequency in Bark.

the distribution

$$y_i^I = \exp\left(\frac{-(x - m_i)^2}{2\sigma^2}\right) \qquad i = 0, 1, \cdots, N-1 \tag{4.1}$$

was used, where m_i was the predetermined mean value corresponding to input unit i and σ^2 was the variance which was constant over all N input units. In the work here, m_i increased in a linear, monotonic fashion with i, between the parameter-dependent upper and lower bounds.

This coarse coding technique was used to code the duration over 12 units in the RBF network with cepstral input data. When training multi-layer perceptrons using the formant data (which has a large dynamic range) the 15 element feature vector was coarse coded onto 218 units, with a heuristic choice of centres. Each bark-scaled frequency was encoded onto 16 units ($m_i - m_{i-1} = 1$ Bark), each bandwidth was encoded onto 10 units ($m_i - m_{i-1} = 50$ Hz), each amplitude ratio was encoded onto 13 units ($m_i - m_{i-1} = 75$) and the duration was encoded onto 18 units ($m_i - m_{i-1} = 20$ ms).

4.1.3 Network architecture

The RBF networks used for these vowel classification experiments consisted of an input layer of 15 or 72 input units, a hidden layer with a varying number of RBF units and an output layer of 20 units, allowing a 1-from-n type coding for each of the 20 vowel classes. The radial basis functions, Φ, were chosen to be Gaussians:

$$\Phi(\|x - c_i\|) = \exp\left(-\sum_j \left(\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right)\right) \tag{4.2}$$

where the means c_i were points in the input space and σ_{ij} was the covariance matrix, which was assumed to be diagonal. Degenerate (or nearly degenerate) function centres were avoided as this would cause the correlation matrix M (equation 2.15) to become (nearly) singular.

In the case of the cepstral coefficient input data a Euclidean distance metric was used. This was a reasonable metric to use: Gray and

Markel [70] have demonstrated that the Euclidean distance metric in N-dimensional cepstral coefficient space following an Mth order LP analysis has a correlation of greater than 0.98 with rms log spectral distance if $N = M$. (Additionally, Gray and Markel showed that as $N \rightarrow \infty$ so the Euclidean distance converges to the rms log spectral distance.) This metric is complicated somewhat for the cepstral data by the duration; this is coarse coded onto a set of units valued between 0 and 1, so the approximation being made is that the appending of durational data (which is of similar magnitude to the cepstral coefficients) does not entirely corrupt the usefulness of the Euclidean distance as a meaningful speech processing distance metric[9]. When using this metric, the widths are assumed to be fixed for each RBF and equal in all directions.

The situation was more complicated for the formant data: here there was a feature vector with a much greater dynamic range. Frequency values varied from 150–3500Hz (or 1–16 Bark), amplitude ratios varied from 50–1000, etc. Clearly a Euclidean distance metric was not suitable; the obvious metric to use was the Mahalonobis distance, which is a covariance weighted Euclidean distance.

The data used in these experiments was very unbalanced: some classes had fewer than 5 examples, while there were nearly 200 examples of the schwa (/@/) class in the training set. Unbalanced data can bias the minimum of the error function to be a point in weight space that classifies better populated classes well, whilst neglecting more sparsely populated classes that contribute little to the overall error function. An attempt to ameliorate this effect was made by modifying the error function to incorporate a factor inversely proportional to the number of training examples in a class. That is, (2.12) was replaced by:

$$E = \frac{1}{2}\sum_{ip}(y_{ip}^{T} - Y_{ip}^{T})^2 m_p \tag{4.3}$$

9 In some vector-quantised hidden Markov modelling systems such as Sphinx [105] or BYBLOS [34] the problem of defining a distance metric over a mixed input space is partially addressed by using multiple codebooks.

$$m_p = \frac{\min_i P_i}{P_c} \qquad p \in c$$

where there are P_c examples from desired class c in the training set. In practice, m_p was given a lower bound of 0.2.

4.1.3.1 Choice of centres and widths

The means by which the centres and widths of the RBFs are chosen is an important algorithm design decision. A fundamental choice is whether these parameters are to be chosen using a supervised or unsupervised method. If the RBFs are to undergo supervised training then the back-propagation algorithm may be employed using the gradients given in (2.34) and (2.35). In this case we have a training situation analogous to the multi-layer perceptron. The situation is a little different as the input-to-hidden weights may be regarded as varying on a slower timescale compared with the hidden-to-output weights [207]. In such a case, the linear hidden-to-output network can be exactly optimised by a pseudoinverse-type method each iteration of the training algorithm.

Unsupervised RBFs may be adaptive or non-adaptive. In either case the choice of RBF centres and widths may be regarded as being performed by a pre-processor, prior to the optimisation of the linear network. Adaptation may be performed by algorithms such as k-means clustering [133] or the EM algorithm [158, 140]. Alternatively the centres may just be a subset of (non-degenerate) data points[10]. A principled way of choosing the covariances might also be via the EM algorithm.

In this work the centres were randomly chosen from the training data. The widths were chosen heuristically. The width of an RBF was defined to be the distance to the nearest neighbouring RBF. This heuristic results in more remote RBFs having larger widths and helps to encourage the RBFs to provide a good covering of the regions of input space where the data lies. In the case of the Mahalonobis distance this width value acts

10 Chen et al. [31] have recently used the method of orthogonal least squares to choose a subset of data points as RBF centres.

Formant Input	
Method	Recognition (% error)
RBF	34.3
MLP	31.7
Gaussian	33.7
HMM	28.2

Table 4–1: Recognition results after training on formant-based feature vectors (+ duration). The classifiers used perceptually scaled formant frequencies (see text).

Cepstral Input	
Method	Recognition (% error)
RBF	26.7
MLP	27.0
HMM	30.6

Table 4–2: Recognition results after training on LPC cepstral coefficients (+ duration)

as multiplier to the diagonal covariance matrix computed from the data.

4.1.4 Results

Results were collected using the data described above, with three different front ends: cepstral, formant (frequencies in Hz) and formant (frequencies in Bark). Classification was performed using RBF networks, multi-layer perceptrons (MLP) (both with varying numbers of hidden units), a multivariate Gaussian classifier [42] and a discrete HMM [8]. Tables 4–1 and 4–2 compare the results obtained using the optimal setup of the different classifiers on this task.

Extensive simulations were carried out using MLPs for this task. Fully-connected models were used with a single hidden layer containing 2–52 hidden units. Results in detail are given in figure 4–2 when cepstral input was used. Five simulations were performed for each size of network

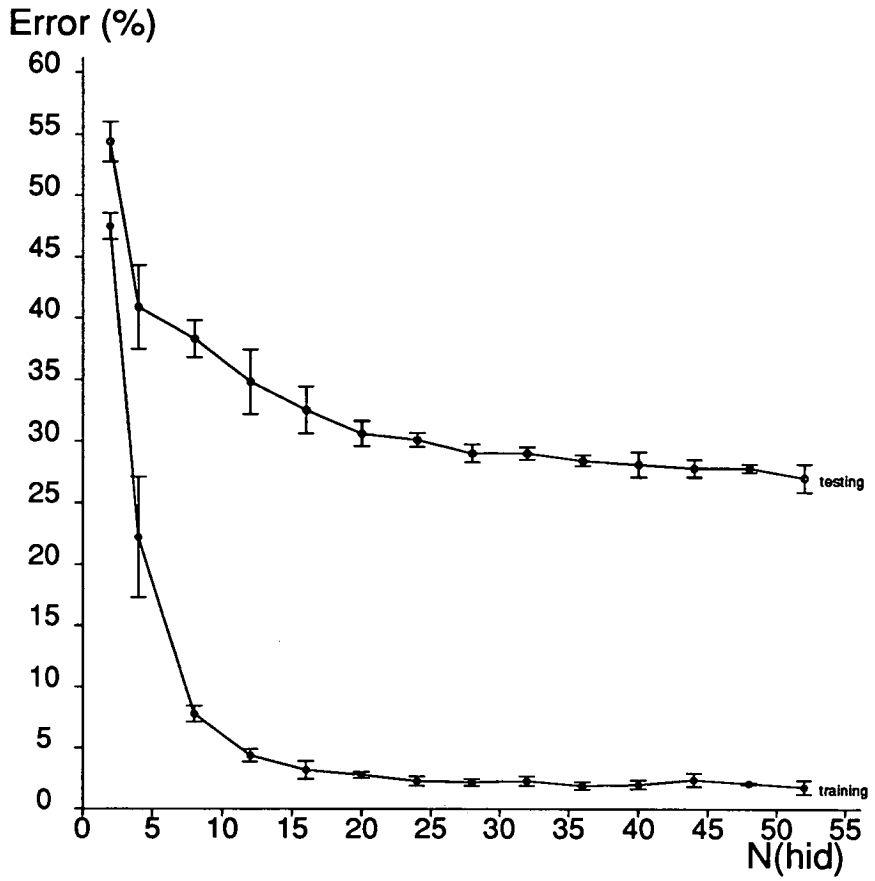


Figure 4–2: Classification results for MLPs when cepstral input was used. The error is plotted against the number of hidden units. The graph points correspond to means, the error bars to standard deviations computed from 5 initial random weight matrices for each network size with *N(hid)* hidden units.

(with different random initial weight matrices) and means and standard deviations of the classification performance on the training and test sets were computed. In the case of the formant data a 28 hidden node network architecture was used and statistics were computed over 10 simulations with different random initial weight matrices. The results in tables 4–1 and 4–2 are mean errors not minimal errors.

A discrete HMM was employed in the 1988 CSTR configuration [8], with each phoneme modelled by a three-state, left-to-right model, with explicit duration modelling at each state — a hidden semi-Markov model.

The VQ codebook was of size 256, with the codewords found by the k-means clustering algorithm (for the formant data, the VQ process used a diagonal covariance Mahalonobis distance metric). This model was applied to exactly the same problem, with the same training data (speech parameters + duration). Of course, the HMM is able to model time more explicitly, so the three-part vowel assumption was not required.

The Gaussian classifier modelled each vowel class as a multivariate Gaussian distribution[11]. Since there were not enough tokens to estimate separate covariance matrices for each class, a single covariance matrix was estimated from the entire data and was used for each distribution. This gave poor results; better results (cited here) were obtained by computing two covariance matrices: one for the ill-defined /@/ class and a second for the remaining classes [42]. Prior probabilities were estimated using the training data. This classifier was only applied to the formant data.

When classifying the formant data, the MLPs, RBF networks and the Gaussian classifier used bark-scaled frequency values and the hidden Markov model used mel-scaled[12] frequency values. The MLP used a coarse coded input with a 218 element feature vector, whereas the RBF networks used a real-valued, unbounded 15 element feature vector and a diagonal covariance Mahalonobis distance metric. The results for the MLP and RBF networks are mean results (over different random initial conditions) for the best size of network, found empirically — 28 hidden units for the MLP[13] or 256 RBFs.

The results for formant input data (table 4–1) indicate that the HMM

11 This classifier may be viewed as a RBF network. See section 2.3.1.

12 The mel-scale is another nonlinear frequency scale derived from auditory psychophysics. It is approximated by:

$$M(f) = \frac{1000}{\ln 2} \ln\left(1 + \frac{f}{1000}\right)$$

where f is frequency in Hz and M is frequency in Mel. It results in a similar transformation to the Bark scale (logarithmic compression of high frequencies).

13 28 hidden units was the smallest number above which the MLP showed no significant improvement on the test set.

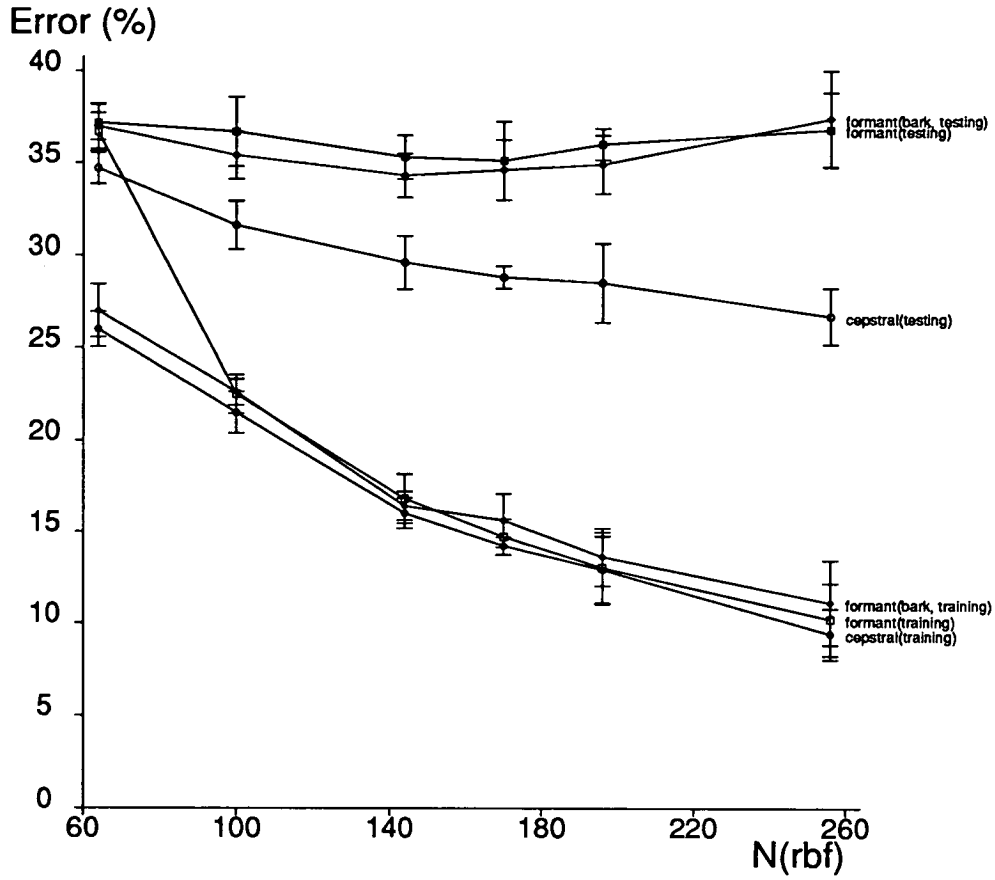


Figure 4–3: Classification results for the radial basis functions network, trained on 759 tokens from 20 vowel classes (12 monophthongs and 8 diphthongs) represented using cepstral data, formant data and bark-scaled formant data. The error is plotted against the number of RBFs. Graph points correspond to means, error bars to standard deviations computed using 12 sets of randomly chosen centres.

considerably out-performs the static classifiers. The RBF network performs worst of all. An advantage of the MLP over the RBF network in this situation was that the MLP was operating on a more redundant coarse coded input set; indeed, this may be regarded as being pre-processed by 218 one-dimensional radial basis functions. Additionally the MLP did not need the explicitly defined distance metric required by the RBF network (this was dealt with by the dimension-specific coarse coding). The Gaussian classifier did require a distance metric: however (as discussed

above), the Mahalonobis distance calculation used 2 full covariance matrices in the Gaussian classifier, whereas a single diagonal covariance matrix was used in the RBF network.

The results were different for the cepstral input data: the RBF network and the MLP both produced higher recognition scores, whereas the HMM produced poorer scores compared with the formant data[14]. In the case of the RBF network the improvement was 7.5%: this may be rationalised by considering the higher-dimensional (and hence more redundant) input coding used (72 inputs opposed to 15) and the fact that the Euclidean distance metric used for the cepstral data has a more natural interpretation in terms of speech processing (see section 4.1.3 above).

Figure 4–3 demonstrates the effect of varying the number of RBFs for both cepstral and formant data. This graph was compiled by training each network size 12 times on each of the three input representations, with a different choice of random centres. Although the error curves for training data were similar for all 3 front ends, the test error curve for the cepstral input was significantly lower than for the two formant-based front ends. There is a trend for the standard deviation of the classification performance to increase with the number of RBFs. A possible explanation for this is that the correlation matrix M (equation 2.15) was more likely to become nearly degenerate as the number of RBFs increases and hence more likely to become ill-conditioned with respect to matrix inversion. In large networks ($>$ 150 RBFs) the matrix would sometimes become extremely ill-conditioned (although not singular), resulting in the "optimal" weight values being extremely large. This effect was reduced by replacing the matrix inversion method (LU decomposition was used here) by a pseudoinverse method, such as singular value decomposition (which requires more computation[15]).

14 The HMM error rate for (linearly scaled) cepstral data is lower than that obtained using linearly scaled formant data (31.1%). (The earlier results were with perceptually scaled formant data.)

15 Simulations (using Numerical Recipes code [156]) indicate that singular value decomposition increases compute time by a factor of 3 compared with LU decomposition.

Error per class (%)									
ii (68)	i (96)	e (38)	a (44)	aa (15)	oo (32)	o (21)	u (10)	uu (47)	uh (33)
13.2	40.6	86.8	40.9	40.0	18.7	33.3	90.0	17.0	45.5
@ (191)	@@ (5)	ei (43)	au (8)	ai (56)	oi (2)	ou (38)	i@ (6)	e@ (3)	u@ (3)
9.4	80.0	4.7	75.0	5.4	0.0	39.5	26.7	66.7	33.3

Table 4–3: Detailed results of vowel recognition for a network with 196 radial basis functions trained using LPC cepstral data. The overall error rate was 36.6% over a total of 759 tokens. () indicates number of tokens of that class. Phonemes are represented in MRPA notation [77].

More detailed results for a network with 196 RBFs trained on cepstral data are given in tables 4–3 and 4–4. It is clear that the RBF network had a low error rate in classifying /@/[16] — over 90% of /@/ tokens were correctly labelled. *A priori* this might not be expected, since this token is not a distinct acoustic-phonetic category and is not clustered in a single region of parameter space. Indeed this class was problematic for the HMM (24.6% error for /@/) and the Gaussian classifier (34.6% error for /@/). The MLP and RBF networks performed better on this large, diffuse class since they were trained using a discriminative method rather than a maximum likelihood method (used for the Gaussian classifier and the HMM). Discriminatively trained networks model well-populated classes better than sparsely populated ones, since class boundary positioning is a data-driven process. The number of free parameters used to model each class in these discriminatively trained networks is dependent on the class population. This is most explicit for RBF networks, since the centres were randomly chosen data points. In maximum likelihood training, each class is modelled individually, with a fixed number of free parameters per model.

The confusion matrix (table 4–4) indicates the most common classification errors made by the RBF network. The two major types of confusion concerned diphthongs and short vowels. Diphthongs were generally classified poorly, being confused with each other and /@/. This was expected

16 The MLP also has low a error rate for /@/.

Confusion Matrix

	ii	i	e	a	aa	oo	o	u	uu	uh	@	@@	ei	au	ai	oi	ou	i@	e@	u@
ii	59	6	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
i	14	57	0	0	0	0	0	0	0	0	23	0	2	0	0	0	0	0	0	0
e	0	8	5	0	0	0	0	0	0	1	15	0	9	0	0	0	0	0	0	0
a	0	1	2	26	1	0	0	0	0	1	2	0	4	0	3	0	4	0	0	0
aa	0	0	0	2	9	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0
oo	0	0	0	0	0	26	2	0	4	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	4	14	0	0	1	2	0	0	0	0	0	0	0	0	0
u	0	0	0	0	0	0	0	1	7	0	2	0	0	0	0	0	0	0	0	0
uu	0	1	0	0	0	1	0	0	39	0	5	0	0	0	0	0	0	1	0	0
uh	0	0	2	1	0	0	0	0	1	18	7	0	0	0	1	0	3	0	0	0
@	0	9	0	0	0	2	1	0	2	0	173	0	1	0	0	0	2	1	0	0
@@	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	2	0	0	0
ei	1	1	0	0	0	0	0	0	0	0	0	0	41	0	0	0	0	0	0	0
au	0	0	0	0	2	0	0	0	0	3	0	0	0	2	1	0	0	0	0	0
ai	0	0	0	1	1	0	0	0	0	0	0	0	0	1	53	0	0	0	0	0
oi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
ou	0	0	0	0	0	0	0	0	2	0	11	0	0	0	0	0	23	1	0	1
i@	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
e@	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
u@	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2

Table 4–4: Confusion matrix corresponding to table 4–3. Note that columns correspond to classes output by the network, rows correspond to desired classes.

since the "onset-centre-offset" assumption made in the data representation clearly breaks down for diphthongs, which are usually more dynamic in feature space then monophthongs. Short vowels (such as /i/, /e/ and /a/) were classified poorly since their acoustic properties are extremely context dependent. There is no time for them to achieve a steady state; the assumption that vowels may be modelled using the onset-centre-offset representation was again shown to be naïve. Unsurprisingly, /@/ (the largest class and most diffuse in feature space) was the most common confuser. A particularly common misclassification was classifying /i/ as /@/ or as the longer vowel /ii/.

A major advantage of using radial basis functions networks is the speed of training — the experiments reported here required a little over 3 minutes training time for a network with 196 RBFs running on a Sun 4; this corresponds to around 3 hours to train a 28 hidden node MLP[17] on the same task. Although the hidden Markov model can be trained in a similar time, recognition is slower than for a neural network as a Viterbi algorithm must be applied at run-time to choose between the models.

4.2 Frame labelling continuous speech

Following the promising results reported in the previous section, a second set of experiments was performed to judge the feasibility of using an RBF network for phoneme modelling in continuous speech. This is a much more difficult problem, as the temporal nature of the speech signal may not be avoided. Instead of assigning labels to presegmented portions of continuous speech, the task here was to assign labels on a frame-by-frame basis.

17 It is quite possible that MLP training could be speeded by using one of the various minimisation algorithms proposed for faster training (e.g. as used in [56]), but it is unlikely that a speed up of over 2 orders of magnitude could be obtained.

4.2.1 Speech database and signal processing

A different speech database was used for these experiments. This consisted of 200 phonetically rich sentences spoken by a male RP British English speaker. The set of sentences was designed to cover every triphone in British English. Half of this set (the even numbered sentences) was designated as training data, the remainder was used as a test set.

The speech was digitised at 20kHz and later downsampled to 10kHz and pre-emphasised with a filter of $1 - 0.98z^{-1}$. A 14th order linear predictive analysis was then performed, using a Hanning window with a width of 20ms and frame shift of 5ms. 14 cepstral coefficients plus the spectral energy were extracted for each frame, giving a 15 element acoustic feature vector to be used as input to the network. The energy value had a much larger dynamic range than the cepstral coefficients. To compensate for this the mean value of the spectral energy was computed over the training set. Incoming patterns had this mean subtracted from their spectral energy, which was then passed through a squashing function bounded by ± 1; $\tanh(\cdot)$ in this case.

4.2.2 Network architecture and training

The network was trained to perform a mapping from an input acoustic parameter vector to a phonetic label. The training data consisted of the acoustic information together with a time-aligned phonetic transcription prepared by an expert phonetician. A set of 44 phonemes was used together with an additional symbol representing silence.

A delay-lined input layer was used, in an attempt to provide some contextual information to the RBF interpolator. Thus the input layer contained 15D units, where D was the width of the delay line. In practice delay-lines of widths ranging from 1 to 15 timesteps were used. The label to be associated with the input vector corresponded to the time of the central set of input parameters in the input. That is, the input layer may be regarded as giving equal right and left context information to

a particular frame. End effects were handled by duplicating the first or last frame where appropriate.

A 1-from-N output coding was used so there were 45 output units representing the set of phonemes. Target values of 1 and 0 were used for these linear output units. The usual least squares error function was used, together with the training method described in section 2.2.2. Matrix inversion was performed using an LU decomposition.

Varying numbers of RBFs (hidden units) were used (50–384). The transfer function of these RBFs was again a Gaussian. The centres were randomly chosen input vectors and the widths were computed by the nearest neighbour heuristic used previously.

4.2.3 Results

The results of the frame-labelling experiments are summarised in figure 4–4. Two clear trends are apparent: the recognition rate increased with N(rbf) and with D. These increases in network size had the effect of increasing the number of free parameters — meaning that more training data was necessary to achieve a valid generalisation — and of increasing the training time. A principal reason for not investigating larger values of D or N(rbf) was the large amount of computer time that these experiments would have demanded. The tradeoff between the number of free parameters and amount of training data may be seen by examining training and test errors as N(rbf) is increased. There was a definite trend for the training error to be lowered at a greater rate than the test score; this suggests that as the number of free parameters increased so the probability of learning an optimal input-output mapping (i.e. one that generalises well) decreased, given a constant amount of training data. A similar trend was not observed as D was increased, suggesting that the increase in free parameters was offset by the additional contextual information offered by a longer delay line.

Table 4–5 gives detailed test results for a N(rbf) = 256 network, with

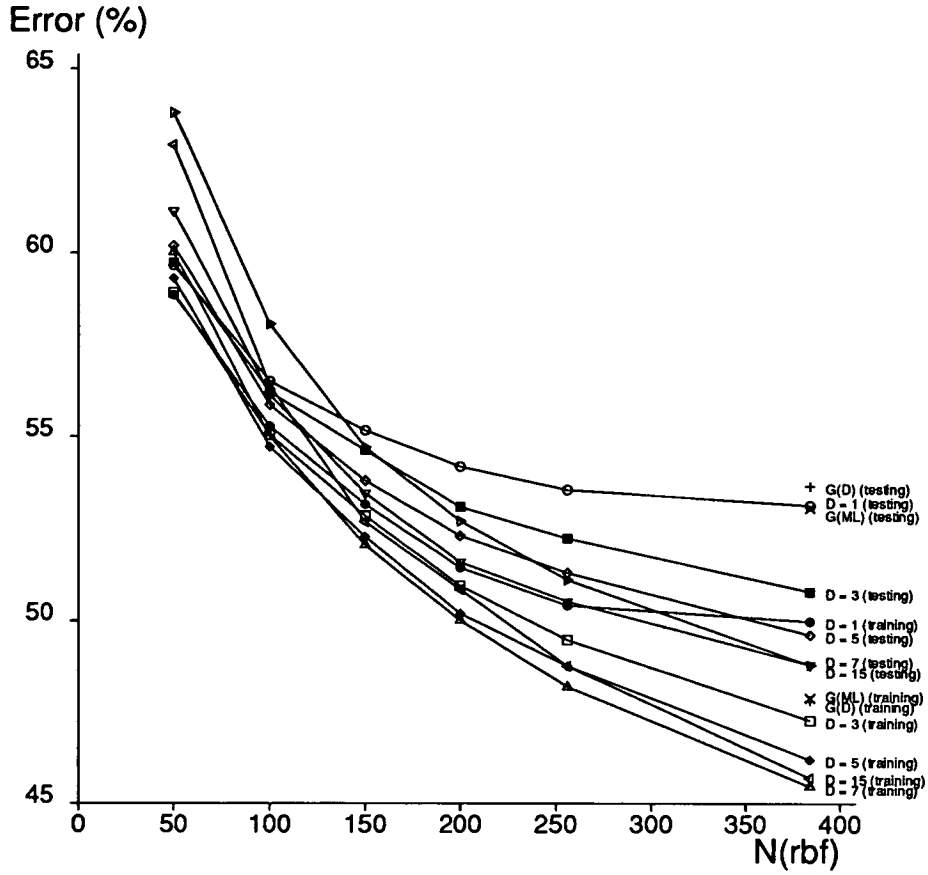


Figure 4–4: Framewise errors from frame labelling 100 sentences of continuous speech spoken by a single male speaker after training on a separate 100 sentences. RBF networks with N(rbf) hidden units, a Gaussian classifier (maximum likelihood training) and a "discriminative Gaussian" classifier (see section 4.3) were used, with a sliding window covering D frames. There were 68,029 frames in the training set and 71,251 frames in the test set. There were 45 output classes corresponding to 44 phonemes + silence.

input sliding windows of size D = 7 and D = 1. In this case the overall framewise error rate is reduced by 3% on using a delay-lined input. As might be expected, the classes that mainly contribute to this decrease are those phonemes that may not be characterised by a single frame: diphthongs (/ai, ei, @@/) and stop consonants (/p, t, b, k/). These phonemes do not reach a steady state, but are characterised by a sequence of events. Diphthongs consist of a trajectory from a starting vowel location to a final vowel location (in addition to the onset and offset); stops consist of

Frame labelling — scores by phoneme					
		D = 7		D = 1	
	Total	Correct	Error (%)	Correct	Error (%)
ə	4460	2643	40.74	2630	41.03
i	3038	1039	65.80	1059	65.14
n	3216	2513	21.86	2456	23.63
l	2097	1159	44.73	1360	35.15
ii	2550	1724	32.39	1668	34.59
t	4034	2277	43.55	1555	61.45
m	1412	561	60.27	642	54.53
r	1539	1273	17.28	1215	21.05
d	1384	271	80.42	277	79.99
ai	2166	720	66.76	629	70.96
w	1223	787	35.65	669	45.30
y	287	12	95.82	6	97.91
p	2136	259	87.87	85	96.02
a	1307	796	39.10	866	33.74
ei	1593	702	55.93	571	64.16
uu	1032	625	39.44	584	43.41
s	1929	936	51.48	906	53.03
b	1786	738	58.68	535	70.04
ou	1637	467	71.47	494	69.82
e	1816	443	75.61	630	65.31
k	2513	1215	51.65	829	67.02
uh	1175	1	99.91	12	98.98
s	4684	3720	20.58	3463	26.07
oo	1665	1012	39.22	966	41.98
ng	476	78	83.61	74	84.45
ch	872	112	87.16	104	88.07
jh	750	115	84.67	218	70.93
g	663	77	88.39	121	81.75
dh	1117	180	83.89	70	93.73
v	735	122	83.40	148	79.86
o	1041	216	79.25	226	78.29
f	1682	1208	28.18	1160	31.03
th	707	0	100.00	1	99.86
sh	782	492	37.08	553	29.28
h	489	65	86.71	89	81.80
zh	224	0	100.00	9	95.98
aa	1779	628	64.70	553	68.92
u	256	0	100.00	0	100.00
au	1097	145	86.78	4	99.64
iə	378	34	91.01	11	97.09
əə	798	304	61.90	203	74.56
uə	70	0	100.00	0	100.00
oi	247	0	100.00	0	100.00
eə	599	29	95.16	2	99.67
##	5810	5510	5.16	5441	6.35
TOTAL	71251	35208	50.59	33094	53.55

Table 4–5: Frame labelling errors by phoneme for a network with 256 RBFs and sliding windows of size 1 (no context) and 7 (3 frames of both left and right context). Phonemes here and elsewhere are labelled using MRPA notation [77].

a closure, a release and (not always) an aspiration. These 7 phonemes accounted for 85% of the decrease in error from D = 1 to D = 7.

The overall difference would be above average except certain classes (notably /l, m, e, jh, sh/) showed an increased error on employing the delay-lined input. These increased test set errors were also reflected in the training set. However the discrepancy between training and testing errors was larger for D = 7 compared with D = 1 and was larger for these particular classes. Thus it may be hypothesised that the degradation in performance for these classes using a D = 7 network is due to the greater number of free parameters causing poorer generalisation. So there is a balance between poorer generalisation and increased context on increasing D; the dominant effect was dependent on the class.

An overall trend, irrespective of D, is that the better populated classes returned lower error rates. For example, /@/ and /s/, the most common phonemes, had low recognition errors. However, /@/ is an ill-defined vowel and is generally regarded as being difficult to recognise. Since there were many examples of /@/, the error function was biased towards a lower class error for /@/. Conversely, several of the less common phonemes, e.g. /oi, u@, u, th/, had a 100% error rate; this was an artifact of the least squares minimisation. The minimum of the cost function was achieved by minimising the error rate on the well populated classes at the expense of the less populated ones.

The results here are given by frame and consider only the top placed phoneme per frame. Bourlard and Wellekens [22] have shown that the output of feed-forward networks trained using a least squares criterion may be interpreted probabilistically. Hence a great deal of information, that would assist in the segmentation, is lost if top scores only are considered each frame. This problem is addressed in section 4.4.

4.3 Discrimination and maximum likelihood

As discussed in section 2.3.1, a Gaussian classifier may be implemented as a RBF network. Such a classifier was applied to the same frame-labelling problem as above. In this case the RBF network contained 45 hidden nodes (RBFs) and 45 output nodes, one for each class. Each RBF had a greater number of free parameters than in the previous section, as a full covariance matrix was specified, rather than a scalar width. This covariance matrix was estimated from the data for each class. The input dimensionality was 15 and there was a mean of 1580 frames per class (with the smallest class, /u@/, having 70 example frames) so reasonable approximations to the true covariances were made for each class[18]. The hidden-to-output weight matrix is diagonal in a maximum likelihood trained Gaussian classifier, compared to the fully-specified matrix found in the usual RBF network. In a Gaussian classifier the diagonal elements correspond to the prior probabilities, here estimated using the training data.

A delay lined input was not used for a Gaussian classifier; this was because using delay lines causes degeneracies between successive training patterns. These degeneracies cause the covariance matrix to be singular (or nearly so), and thus incompatible with a Gaussian classifier.

The unimodal Gaussian classifier is trained using a Bayesian maximum likelihood process. This is optimal if each class is known to be a Gaussian and the training sample is sufficiently large. However, it is unlikely that each class is described by a Gaussian distribution, so a Gaussian classifier is not optimal. Using an RBF representation it is trivial to impart discriminative power into what was a (non-discriminative) Gaussian classifier. This may be achieved by replacing the diagonal hidden-to-output weight matrix (containing the prior probabilities) by a full matrix, that can be trained using the usual pseudoinverse approach. A Gaussian classifier in which the prior probability diagonal matrix has been substituted

18 Of course, this does not mean that sample covariance is a good statistic for the problem.

Frame labelling — scores by phoneme					
		maximum likelihood		discriminative	
	Total	Correct	Error (%)	Correct	Error (%)
ə	4460	1857	58.36	1835	58.86
i	3038	863	71.59	824	72.88
n	3216	2374	26.18	2299	28.51
l	2097	1351	35.57	1328	36.67
ii	2550	1655	35.10	1566	38.59
t	4034	1162	71.19	1235	69.39
m	1412	774	45.18	789	44.12
r	1539	1215	21.05	1207	21.57
d	1384	285	79.41	324	76.59
ai	2166	376	82.64	561	74.10
w	1223	770	37.04	761	37.78
y	52	287	81.88	60	79.09
p	2136	331	84.50	207	90.31
a	1307	766	41.39	730	44.15
ei	1593	564	64.60	571	64.16
uu	1032	587	43.12	582	43.60
s	1929	946	50.96	931	51.74
b	1786	689	61.42	651	63.55
ou	1637	350	78.62	353	78.44
e	1816	415	77.15	439	75.83
k	2513	852	66.10	886	64.74
uh	1175	95	91.91	120	89.79
s	4684	3520	24.85	3369	28.07
oo	1665	1084	34.89	1030	38.14
ng	476	192	59.66	213	55.25
ch	872	158	81.88	186	78.67
jh	750	254	66.13	250	66.67
g	663	189	71.49	204	69.23
dh	1117	206	81.56	185	83.44
v	735	183	75.10	239	67.48
o	1041	350	66.38	407	60.90
f	1682	1194	29.01	1207	28.24
th	707	35	95.05	45	93.64
sh	782	586	25.06	586	25.06
h	489	233	52.35	241	50.72
zh	224	44	80.36	66	70.54
aa	1779	957	46.21	742	58.29
u	256	34	86.72	11	95.70
au	1097	115	89.52	163	85.14
iə	378	108	71.43	104	72.49
əə	798	426	46.62	381	52.26
uə	70	0	100.00	2	97.14
oi	247	24	90.28	35	85.83
eə	599	91	84.81	100	83.31
##	5810	5135	11.62	4997	13.99
TOTAL	71251	33447	53.06	33022	53.65

Table 4–6: Frame labelling errors by phoneme for a Gaussian classifier trained using maximum likelihood (ML) and using a discriminative least squares method. The overall training scores for these classifiers were 47.87% for the ML classifier and 47.80% for the discriminatively trained classifier.

by a fully-specified least-squares optimal matrix shall be termed a "discriminative Gaussian classifier".

Both forms of Gaussian classifier were applied to the same speech problem as above. The overall and per phoneme errors are given in table 4–6. An examination of table 4–6 indicates that there were no major differences between the first choice frame classifications of the maximum likelihood and discriminative classifiers. However there were significant differences between the within class scores of the Gaussian classifiers and the RBF network (with D = 1 (table 4–5) which received similar inputs). The principal difference is that the Gaussian classifiers were not as sensitive to the class sample size, compared with the RBF network. The prime examples of this may be seen with the less frequent vowels such as /e@, @@, i@/. These all produced 100% error rates for the RBF network, whereas the Gaussian classifier produced error rates of between 50% and 85% for these classes. Conversely, some of the more frequent classes such as /@, a, i/, had much lower error rates with the RBF network.

These differences may be explained by the way the hidden layer of RBFs was constructed. In the standard RBF network, the RBF centres were drawn randomly from the input data; hence the representation in RBF space was strongly dependent on the input data. This was not so for the Gaussian classifiers; there was one centre per class, corresponding to the class mean. A larger number of training examples for a particular class simply improved the estimate for mean and covariance.

4.4 Modelling time

It is unlikely that modelling time using delay lines alone is adequate for continuous speech recognition. Rather than an extended use of delay lines (a time-delayed RBF network) or the addition of recurrences, the temporal modelling used here is a post-processing operation: Viterbi smoothing.

4.4.1 Viterbi smoothing

Lowe and Webb [118] have proved that the outputs of a network whose final layer weights are normalised using a pseudoinverse process (i.e. linear output units) satisfy a particular linear constraint dependent on the target vectors used for training. In particular, a 1-from-n output representation constrains the outputs to sum to 1. These outputs are not probabilities, however, not least because they are not constrained to be non-negative. In this work, it is hypothesised that these outputs may approximate probabilities, after some transformation. Here negative outputs were set to zero and this new set of outputs was normalised to sum to unity. It was then assumed that these transformed outputs were good approximations to the posterior probabilities.

Thus for each frame i there is a set of probabilities p_{ij} denoting the probability of symbol (label) j in that frame. A sequence of symbols is required, where s_i is the symbol corresponding to the ith frame. This can be trivially achieved by choosing the maximal p_{ij} for each frame, as this maximises

$$\prod_{i=0}^{N-1} p_{is_i}, \tag{4.4}$$

the probability of a sequence of symbols. This is not acceptable as the resultant sequence will change classification with great frequency. To obtain a suitable segmentation, a notion of segment duration is required, to penalise frequent changes in symbols.

A simple way of achieving this is by regarding the symbol sequence s_i as the output of a first order Markov process that is parameterised by an initial vector of probabilities b_i and a transition matrix a_{ij} (both of which may be estimated from the training data). By assuming that speech is a first order Markov process, information about frame labelling from the training data (the fact that frame labels clump together in segments, average segment duration, etc.) may be used to make better estimates of the prior probabilities.

In (4.4), the prior probability of a sequence $s_0 s_1 s_2 \ldots s_{N-1}$ was uni-

form and equal to C^{-N}, if there were C classes. Using the estimates of priors produced using the Markov model, the prior probability of that sequence is given by $b_{s_0} a_{s_0 s_1} a_{s_1 s_2} \ldots a_{s_{N-2} s_{N-1}}$. Denoting the sequence $s_0 s_1 \ldots s_{N-1}$ by s_0^{N-1}, we have the prior probability

$$p(s_0^{N-1}) = b_0 \prod_{i=1}^{N-1} a_{s_{i-1} s_i}. \tag{4.5}$$

Hence the new expression for the posterior probability that should be maximised to give the optimal sequence is:

$$p_{0 s_0} b_{s_0} \prod_{i=1}^{N-1} p_{i s_i} a_{s_{i-1} s_i}. \tag{4.6}$$

The optimal sequence may be computed using the Viterbi algorithm. The array q_{ij} is defined as the likelihood of the most likely sequence s_0^i ending at frame i with class j:

$$\begin{aligned} q_{ij} &= \max_k q_{i-1,k} a_{ki} p_{ij} \qquad 1 \leq j < N \\ q_{0j} &= b_j p_{0j} \end{aligned} \tag{4.7}$$

To derive the optimal class sequence, a note must be kept of the k used in the maximisation of (4.7) at each frame. If t_{ij} is the value of k used to compute q_{ij} then the optimal label sequence may be computed using the backwards recursion:

$$\begin{aligned} s_{N-1} &= \operatorname*{argmax}_i q_{N-1,i} \\ s_i &= t_{i+1,s_{i+1}}. \end{aligned} \tag{4.8}$$

4.4.2 Phoneme lattices

The Viterbi smoothing may be regarded as a segmentation process, in which each segment is associated with its most likely label. However, a set of phoneme hypotheses, with associated probabilities, is required for each segment to avoid discarding most of the information output by the RBF network. If a segment lasts from frame m to frame n, then:

$$s_m = s_{m+1} = \ldots = s_{n-1} = s_n. \tag{4.9}$$

The Viterbi algorithm produces the probability of s_m^n consisting of the most likely class. One way of computing the probability of segment s_m^n being labelled by symbol j is:

$$p(s_m^n = j^{n-m}) = \sum_{i=0}^{C-1} p_{m-1,i} a_{ij} \prod_{k=m}^{n} p_{kj} a_{jj}. \tag{4.10}$$

However a problem with this expression is that the computed probabilities for the previous segment are being propagated forward; this is desirable if these probabilities are known to be correct (or nearly so), otherwise this would just have the effect of propagating errors. So the expression may be amended to treat each segment in isolation:

$$p(s_m^n = j^{n-m}) = \prod_{k=m}^{n} p_{kj} a_{jj}. \tag{4.11}$$

The above expressions have the effect of penalising long segments, since the overall probability of a segment being labelled by a particular symbol is the product of framewise probabilities. This is undesirable as it may result in the lexical access portion of a system deleting long segments in preference to short segments. To counter this, a geometric mean was taken of the probabilities in each segment, as a form of time normalisation:

$$p(s_m^n = j^{n-m}) = (\prod_{k=m}^{n} p_{kj})^{\frac{1}{n-m}} a_{jj} \tag{4.12}$$

This equation was used to extend a string of phoneme labels into a lattice of time-aligned phoneme probabilities.

4.4.3 Evaluation of phoneme lattices

Evaluation of a complete speech recognition system is comparatively straightforward: the quantity of interest is usually the utterance recognition error. However, it is often desirable to evaluate a phoneme lattice produced by a front end without using language modelling — i.e. the evaluation of a portion of a speech recognition system. The most usual means of doing this is by comparing the optimal string of phonemes produced by the front end and computing the percentage of correct segment

labels together with the percentage of errors, after comparison with the correct phonetic transcription (which need not be time-aligned). Errors may be divided into 3 categories: substitutions, insertions and deletions. If a single figure is required then the overall error, E, may be computed:

$$E = 100 - (s + i + d), \tag{4.13}$$

where s, i and d are the percentages of substitutions, insertions and deletions.

This evaluation metric is only partially satisfactory, as it takes no account of multiple label hypotheses for each segment, which would be an integral part of any higher level language modelling processes. McInnes et al. [128] have suggested using an entropy measure to evaluate phoneme lattices. The basis of this measure involves aligning a phoneme lattice with the correct transcription and finding the path through the lattice with the maximal probability for correct recognition. Clearly this involves assigning insertion and deletion probabilities as well using the label probabilities from the lattice. The entropy measure quantifies the uncertainty (per phoneme) of a phoneme lattice and it may be estimated using

$$H = \frac{1}{N}\langle \log_2 \frac{\sum_{\text{correct paths}} P(\text{path})}{\sum_{\text{all paths}} P(\text{path})} \rangle \tag{4.14}$$

where there are N phonemes in the utterance. The insertion and deletion probabilities needed to estimate P(path) may be computed iteratively using a forced Viterbi alignment. A forced Viterbi alignment is an iterative procedure that searches for the most likely path through a phoneme lattice that matches a given string of phonemes. To enable the match to be made, phoneme segments may be deleted or inserted. An initial set of deletion and insertion probabilities is specified; these are re-estimated after each iteration, using frequency information.

4.4.4 Language modelling

The language modelling was performed by a lexical access routine, which transformed an input phoneme lattice into a list of scored sentence hy-

potheses. The CSTR lexical access program, LexAx, was used, in the July 1989 configuration [129]. LexAx uses a Viterbi algorithm (with beam searching), implemented using finite state transducers within a chart parsing framework. The phoneme-phoneme confusion costs, needed for the Viterbi alignment were generated by designating half the phoneme lattices as a training set and performing a forced Viterbi alignment to the correct transcriptions; the costs were computed by counting confusions. This process may be iterated, but in practise one iteration was sufficient: further iterations led to "overtraining" due to limited training data. LexAx used a tree structured finite-state lexicon to map phoneme strings to word hypotheses and higher level language modelling was provided by a deterministic finite state grammar compiled from a phrase structure grammar.

4.5 Continuous speech recognition experiments

The techniques described in the previous section were used in a set of single speaker continuous speech recognition experiments, in which an input speech time waveform was processed to produce an output sentence.

4.5.1 Procedure

Previously, a 200 sentence phonemically rich database was split into training and test sets of 100 sentences each when performing frame labelling experiments. This was necessary as time-aligned transcriptions were required to evaluate the frame labelling. Time-aligned transcriptions are not required to evaluate a continuous speech recognition experiment, as the evaluation is carried out on a phoneme segment, word or sentence level. In this work, the transcribed database of 200 sentences (11-12 minutes of speech) was used as training data and the test set was a database of 170 sentences (9-10 minutes of speech), from a particular task-specific domain — cytopathology reports. Although the test set was spoken by the same speaker as the training set, there was some variation as the

test set was recorded over a year later. There was only a small overlap between the training lexicon and the test lexicon — the training data was produced with the aim of covering the acoustic space, whereas the test data corresponded to a particular language model.

Time-aligned transcriptions were not available for the test set. However, non-time-aligned transcriptions (i.e. a string of phonemes for each sentence) were available. These were used for evaluation of the phoneme lattices and in estimating insertion, deletion and substitution probabilities for phoneme lattice evaluation and for LexAx. In this case the test set was split in two, half of which was used for probability estimation using the transcriptions, the other half used for testing.

The same set of networks (with the same random number seeds where applicable[19]) were used in the frame-labelling experiments. After having trained and run over the test set, a set of approximated phoneme probabilities for each frame of each utterance was generated. These were input to the Viterbi smoothing routine described above. The phoneme-phoneme transition probabilities (a_{ij}) for the Viterbi smoothing were computed from the test data using frequency information, on a frame-wise basis (a small offset was added to prevent zero probabilities arising as an artifact of limited sample size). These probabilities also acted as a duration model for each phoneme as they included self-self probabilities. The initial state in the Viterbi smoothing was constrained to be a silence: every utterance in the database was known to start (and finish) with a silence.

The output of the Viterbi smoothing was a phoneme lattice for each utterance: a typical phoneme lattice is shown in figure 4–5. This was produced as described above, using a single segmentation with a list of 45 phoneme probabilities for each segment. The lattices were evaluated by computing the per phoneme entropy and by word and sentence recog-

19 The same random number seed was used for different networks, so the first 100 RBF centres of a N(rbf) = 256, D = 1 network would be the centre frames of the RBF centres in a N(rbf) = 100, D = 7 network.

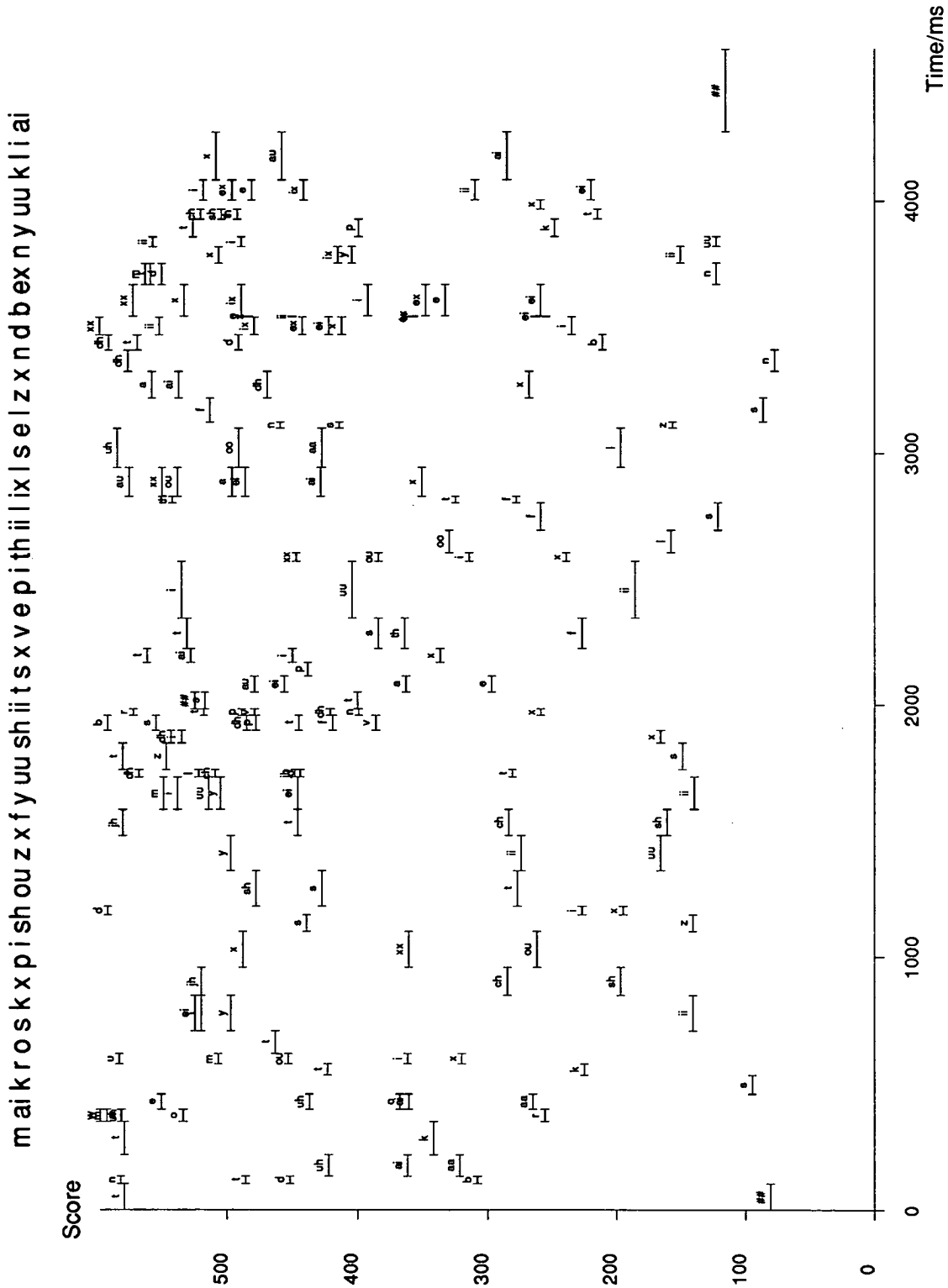


Figure 4–5: A phoneme lattice produced using an RBF network with D = 1 and N = 256, followed by a Viterbi smoothing. Note that the /@/ symbol has been substituted by /x/ in this figure. The y axis is scaled in negative log probabilities (in fact, $-128 \log p$). The true phonetic transcription is shown above the lattice. The text of this sentence was "Microscopy shows a few sheets of epithelial cells and bare nuclei.".

			Lexical access — % errors					
	Free		Top 1		Top 2		Top 10	
	Parameters	Entropy	Word	Sentence	Word	Sentence	Word	Sentence
150/1	9195	3.59	17	25	15	19	14	16
150/7	22695	3.52	14	24	12	18	12	14
200/1	12245	3.61	16	22	15	16	14	14
200/7	30245	3.46	12	20	9	12	9	10
256/1	15661	3.57	6	14	4	6	4	4
256/7	38701	3.41	13	25	10	13	10	8
G(ML)	10845	3.72	9	15	8	11	8	8
G(D)	12870	3.54	14	21	11	12	10	9
HMM	34830	–	4	8	–	–	3	4

Table 4–7: Word and sentence errors after lexical access on phoneme lattices using a restricted (perplexity = 6) grammar with 571 word lexicon. "256/7" refers to an RBF network with N(rbf) = 256 and D = 7; G(ML) is a maximum likelihood trained Gaussian classifier, G(D) is a discriminative Gaussian classifier. The errors produced using the CSTR HMM front end in its best configuration for this problem are also given.

nition scores after passing through LexAx (using either a constrained low perplexity grammar or no grammar whatsoever).

Half of the test utterances were reserved for computing entropy and lexical access scores; the other half were used for a forced Viterbi alignment needed for both the entropy calculation and the lexical access. In the entropy calculation, a training set was required to iteratively compute the phoneme insertion and deletion probabilities; in the lexical access phoneme-phoneme transition costs were computed as well as insertion and deletion probabilities. Hence evaluation was performed on 85 utterances (4-5 minutes of speech).

4.5.2 Results

The results of this continuous speech recognition experiment are summarised in tables 4–7 and 4–8. The best performing network using the

	Lexical access — % errors							
	Free		Top 1		Top 2		Top 10	
	Parameters	Entropy	Word	Sentence	Word	Sentence	Word	Sentence
150/1	9195	3.59	58	95	55	92	50	87
150/7	22695	3.52	61	96	57	93	54	92
200/1	12245	3.61	57	93	54	91	50	89
200/7	30245	3.46	58	91	54	89	51	86
256/1	15661	3.57	53	92	50	92	46	87
256/7	38701	3.41	57	93	52	91	48	87
G(ML)	10845	3.72	42	91	40	89	38	88
G(D)	12870	3.54	52	93	50	92	47	85
HMM	34830	–	53	95	–	–	–	–

Table 4–8: Lexical access errors (%) using no grammar (perplexity = lexicon size) and 571 word lexicon. Notation as in table 4–7.

perplexity 6 grammar contained 256 RBFs with a single frame input (notated a 256/1 network): 6% of the first choice words were erroneous. This compares well with the best configuration of the CSTR hidden Markov model front end[20] which produced a 4% word error using the same data. The maximum likelihood Gaussian classifier performed better than any of the other RBF networks, producing a 9% first choice word error. Results without using a grammar were much inferior (as would be expected with the perplexity increasing from 6 to 571). However, the maximum likelihood Gaussian classifier used here was markedly superior, with a 10% lower word error than any of the other configurations.

If scores on the top two and top ten choices are considered in table 4–7, then the 256/1 network may be seen to be significantly superior to the Gaussian classifier, especially in terms of sentence accuracy. For example, the 256/1 network achieved a sentence error rate of 6% when considering the top two hypotheses, compared to 11% for the Gaussian

20 This was a discrete hidden Markov model with a size 256 codebook, and 45 phoneme models. Each phoneme model was a 3 state left-to-right model with self-recurrences and explicit state duration modelling.

classifier. There is a large increase in sentence error for the 256/1 network when considering the top hypothesis compared with considering the top two, owing to confusions between small function words. The most common confusions were "but" being confused with "with" and "and" being confused with "or". Examination of the hypotheses output by LexAx indicated that there was only a small difference in sentence probability between these confusible pairs. The problem of confusing these function words is well-known; in the Sphinx system, for example, it was tackled by the construction of specific function word models, rather than constructing them out of phoneme models [105].

The results of the maximum likelihood Gaussian classifier (Gaussian (ML)) on the grammar-free task are surprisingly good; there is no ready explanation to account for the fact that this network configuration has a 10% lower error than any of the other networks or the HMM front end. Since this task has fewer constraints, the accuracy of front-end phoneme modelling is likely to be more important than in the restricted grammar case. As the discriminatively trained Gaussian classifier (Gaussian (D)) also outperformed the other RBF networks, it may be hypothesised that the full covariance modelling used in these classifiers (and none of the others) was important.

There is no significant correlation between the per phoneme entropy and the word (or sentence) errors. In the case of the restricted grammar, the entropy scores and word recognition errors over the top 10 choices are almost entirely uncorrelated with a correlation coefficient of -0.01. For example, the Gaussian (ML) classifier produced low error rates on both lexical access tasks, yet had the highest per phoneme entropy of any of the classifiers investigated; conversely the 256/7 RBF network had the lowest per phoneme entropy, yet produced mediocre word and sentence recognition results. An explanation for the particularly high per phoneme entropy returned by the Gaussian (ML) classifier, may lie in the maximum likelihood training procedure. Since this is non-discriminative, there was no term in the cost function that was optimised to reduce the

probabilities for competing classes: the only impetus was to maximise the probability of the correct class. In recognition this led to a "clumping" of probabilities for each segment, whereby the chosen class did not have a probability vastly greater than the competing classes. This tends not to be the case for discriminative classifiers. This clumping was evidenced by the time taken to train LexAx using the output of the Gaussian (ML) configuration. This time was approximately 10× that of the other networks, indicating that the beam search employed by LexAx was not effective: the probabilities were sufficiently tightly clumped that the beam included most hypotheses, thus barely reducing the search space.

This lack of correlation between the entropy measure of front-end performance and final utterance recognition need not be seen as a reflection on the validity of entropy as a measure of front-end performance. Indeed, the entropy measure is an information theoretically correct measure to apply. However, the computed entropy is only the correct entropy given that the correct prior probabilities are used. These prior probabilities are largely determined by the grammar and lexicon used for the recognition. However, the per phoneme entropy is a purely front end measure of accuracy of phoneme modelling and is not influenced by the language model. These results simply indicate that the priors estimated in the entropy calculation are considerably different to the true priors taking into account the language model.

Finally it must be emphasised that the final lexical access test set consisted of 85 sentences, containing a total of 525 words, corresponding to less than 5 minutes of speech. This set is too small to make a good evaluation of various speech recognition configurations. A more thorough examination would require a larger database for both training and testing.

4.5.3 Discussion

4.5.3.1 Viterbi training

A frequently cited advantage of HMMs over neural network systems (e.g. [105]) is that time-aligned phonetic transcriptions are necessary for neural network training in contrast to training hidden Markov models (once the phoneme models have been bootstrapped). This is not so: neural networks require neither more or less transcribed training data compared with HMMs. If a Viterbi training procedure is used, then a neural network model — be it a TDNN, a set of predictor networks or the type of network used in the previous section — may be trained using a non-time-aligned sequence of phonemes. Such a string of phonemes may be obtained from a sequence of words if there is a phonetically structured lexicon, which is usually the case. The Viterbi training method can also train networks in the usual situation of alternative pronunciations, when there may be many phoneme strings corresponding to a single word.

Consider the network of the previous section, with one output for each phoneme. Initially the network must be bootstrapped using a time-aligned phonetic transcription — this is easily accomplished using a readily available speech databases, such as the TIMIT database. This database may also used to initialise the transition probabilities for the Viterbi algorithm. Training may now proceed using speech data consisting of a set of acoustic vectors for each utterance, together with a string of phonemes corresponding to that utterance. The training schedule is as follows:

1. Run the network using the acoustic input data, to produce a sequence of frame labels.

2. Perform a forced Viterbi alignment, matching the output frame labels with the target sequence of frame labels. Re-estimate the substitution, insertion and deletion probabilities.

3. Re-estimate the phoneme-phoneme transition probabilities by counting

transitions in the new alignment.

4. The output of the forced Viterbi alignment is now used as time-aligned target data for the feed-forward network. Retrain the network to map the input data to this target data.

This sequence may be iterated until some convergence criterion is met.

4.5.3.2 Semi-continuous hidden Markov models

The above method of training is similar in spirit to the Connectionist Viterbi Training (CVT) introduced by Franzini et al [56] and described in section 3.7. In CVT a feed-forward network was used to map input acoustic data to outputs corresponding to HMM emission probabilities. Consider using a RBF network for this task. The network performs a mapping from the input acoustic data $\mathbf{x}$ to the output emission probabilities of the HMM $b_{s_t s_{t+1}}(\mathbf{x})$. (For clarity $b_{s_t s_{t+1}}(\mathbf{x})$ is sometimes written as $b_i(\mathbf{x})$, with the transition from s_t to s_{t+1} being notated as transition i.) The equation of this network is:

$$b_{s_t s_{t+1}}(\mathbf{x}) = b_i(\mathbf{x}) = \sum_{j=1}^{N} w_{ij}\Phi_j(\mathbf{x}), \tag{4.15}$$

where s_t is the state of the HMM at time t, Φ_j is the jth RBF (all RBFs are Gaussians in this example) and w_{ij} are the weights from the jth RBF to the ith emission probability.

The structure of this network is isomorphic to that of the semi-continuous hidden Markov model (SCHMM) introduced by Huang [82]. This model may be considered to be a form of continuous hidden Markov model with all emission probabilities sharing a Gaussian mixture probability density function. Alternatively, it may be intuitively regarded as a discrete hidden Markov model with an interpolating, overlapping codebook. In the SCHMM, the expression for the emission probability ($b_{s_t s_{t+1}}(\mathbf{x})$) may be written in terms of the discrete emission probabilities using a VQ code-

book:

$$
\begin{aligned}
b_{s_t s_{t+1}}(x) &= p(x|s_t, s_{t+1}) \\
&= \sum_{j=1}^{N} p(x|O_j, s_t, s_{t+1}) P(O_j|s_t, s_{t+1}) \\
&= \sum_{j=1}^{N} p(x|O_j) b_{s_t s_{t+1}}(O_j) \\
b_i(x) &= \sum_{j=1}^{N} b_i(O_j) p(x|O_j)
\end{aligned}
\tag{4.16}
$$

Here $b_{s_t s_{t+1}}(O_j)$ is the discrete emission probability: that is the probability of symbol (or VQ codeword) O_j being emitted when there is a transition from state s_t to s_{t+1}. The PDF $p(x|O_j)$ specifies the probability the acoustic data will be x when VQ codeword j is activated. If the RBFs Φ_j are identified with the VQ PDFs $p(x|O_j)$, and the weights w_{ij} with the discrete emission probabilities $b_i(O_j)$, then we see that SCHMM equation (4.16) is implemented by RBF network (4.15).

A SCHMM is trained by a maximum likelihood procedure, using the EM algorithm. In this procedure the means (μ_j) and covariances (Σ_j) defining the Gaussian PDFs $p(x|O_j)$ are also re-estimated, in addition to the HMM parameters a_{ij}, $b_i(O_j)$ and c_i. The reestimation equations for the μ_j and Σ_j (given in [82]) are simply an extended case of the usual EM algorithm for Gaussian mixture densities [158]. However, they include feedback from the HMM, and are thus dependent on the forward and backward probabilities computed in the Baum-Welch algorithm. Thus the VQ codebook is optimised by a HMM maximum likelihood process, rather than the usual unsupervised distortion minimisation process, such as the k-means clustering algorithm. Using the framework discussed in chapter 3, SCHMM training involves a multiplicative back-propagation through time in the HMM, summed over time and all state transitions.

4.6 Summary

The results of a progression of single speaker speech recognition experiments using radial basis function networks have been reported in this

chapter. The first experiment was a static vowel classification; the RBF network demonstrated that it was a powerful classifier for this task, producing results not worse than a MLP, a discrete HMM and a Gaussian classifier. However this experiment was limited by the relatively small amount of (unbalanced) training data that was available when it was performed.

The following experiments all attempted to model, in some way, continuous speech, with the aim of producing a continuous speech recogniser. The first attempt was the production of a frame labelling system in which a sliding window of acoustic input vectors were mapped onto their correct frame labels. Various RBF networks were applied to this task, including Gaussian classifiers trained both discriminatively and using a maximum likelihood method. However, these frame labellers had no longer timescale model of speech dynamics, and were thus unable to produce a suitable segmentation. This problem was solved by assuming the sequence of phonemes was produced by a first order Markov model: the transition probabilities for this model were estimated using the training data. This crude model was then applied as a post-processor to the output of the RBF networks, using a Viterbi smoothing process. The output of this process was a segmentation and corresponding phoneme lattice for each utterance. Evaluation of these phoneme lattices was performed by estimating the per phoneme entropy. Finally, these phoneme lattices were passed to the CSTR language model, LexAx, which performed a lexical access process using the Viterbi algorithm (with a beam search for efficiency), with a tree-structured phonetic lexicon and a low-perplexity phrase structure grammar. Using this constricting grammar, recognition errors of 6% (word) and 14% (sentence) were obtained, which compared well with the best CSTR HMM errors of 4% (word) and 8% (sentence).

Additionally Viterbi training methods for neural networks were discussed and it was demonstrated that an RBF network trained using Connectionist Viterbi Training (CVT) and a semi-continuous hidden Markov

model were isomorphic models using the same form of PDF to estimate the HMM emission probabilities.

There is much scope for future work. The RBF networks may be extended by using covariance or diagonal covariance matrices instead of the scalar widths used in this work. To avoid a massive increase in the number of free parameters, these covariances could be tied between several RBFs. Presently, the means for the RBF networks are chosen randomly from the training data; this might be improved by using some form of competitive training to choose them. Indeed, the EM algorithm could be employed to estimate both means and covariances for the RBFs. There have been conflicting results on the usefulness of adaptive RBFs [140, 117].

Perhaps the most promising area for immediate future work is in the links between HMMs, feed-forward networks and Viterbi training. Speech recognition experiments using RBF networks in a Viterbi training framework would be most interesting, particularly when compared with the results obtained using the similar semi-continuous hidden Markov models.

5

NETWORKS WITH FEEDBACK

5.1 Dynamical neural network models

5.1.1 Motivation

The principal problem addressed in this thesis has been speech recognition. Although this is a dynamic process, the models employed were static, layered models (albeit, with delay lines and time-dependent post-processing); hence fixed point dynamics were guaranteed (this is, of course, a trivial statement for a layered network). This approach has several drawbacks: the network has no memory of previous states (as there are no feedback connections) so all context has to be specified by a sliding window or delay line[1].

For a more complex signal the approach of fixed time delays is problematic. Varying parts of the speech signal (and indeed higher level processes such as syntax) require varying amounts of context for optimal prediction; a fixed length context is wasteful and probably inadequate because some parts of the signal will require a greater degree of context than others. So the signal will be imperfectly modelled and the number of degrees of freedom used in the interpolation will be greater than necessary. A method to address this problem in a statistical framework has been offered by Rohwer [170], who described an algorithm for the

1 Dynamical systems may be predicted by learning functions that map from n previous outputs of the system to the current output [192, 143, 54]. An example of such functions are feed-forward networks with a delay-lined input.

construction of a variable-order Markov model.

The approach then, is to construct a dynamical system to emulate the system under investigation. In hidden Markov modelling a piecewise stationary, probabilistic dynamics is used to model each speech unit. This has been a successful approach, due mainly to the power and generality of the EM algorithm. However, this is a poor model of speech dynamics. A more natural method might be to use a recurrent neural network for dynamical modelling. Recurrent networks allow computation with non-fixed point attractors, with which it seems natural to model complex time-dependent processes such as speech production and recognition, motor control and higher level cognitive processes. However, work on developing learning algorithms for recurrent networks is still at an early stage. Here, I shall be concerned with characterising the dynamical behaviour of a particular network model.

Complementing the impetus from engineering described above, neurobiologists have produced experimental results and physiological models indicating the importance of complex dynamical behaviour in the brain. Over the past 20 years, Freeman and co-workers [58, 186, 13] have stressed this and have offered experimental evidence for chaotic behaviour in the olfactory bulb [186]. Models taking into account this (and other) physiological evidence have been proposed by Baird [13] and Li and Hopfield [112]. Both these models regard stored memories as oscillations rather than fixed points. Guevara et al. [71] have discussed experimental results and mathematical models of individual neural oscillators and networks modelled by time-delay equations which suggest that chaotic dynamics is an important feature of neurobiology. Babloyantz and Destexhe [10] have also produced evidence for chaotic activity in the brain, using well-known methods from dynamical systems to analyse time series obtained from EEGs. They have produced values for the fractal dimension of chaotic attractors resulting from normal brain activity and epileptic seizure; they suggested that epileptic seizure may correspond to bifurcation to a low-

dimensional chaotic attractor[2]. Recent results in visual cortex, obtained independently by two groups [69, 47], have offered further evidence for the importance of dynamical behaviour in understanding brain function. These results are especially interesting as they have provided evidence of spatial phase locking, which may provide a mechanism for establishing relations between different areas of the visual field.

5.1.2 Dynamical studies of neural network models

Recently, several researchers have reported complex dynamics[3] in neural network models, arising from both theoretical analysis and simulation. Early work in neural network modelling laid stress on the notion of oscillations as memory states: simulations at Lincoln Laboratory [52] and IBM [168] produced sustained complex dynamical behaviour in random networks of binary threshold neurons ("diffuse reverberations" was the term used at IBM). Such behaviour was later theoretically investigated (in terms of graph theory) by Sloane [187] who showed that certain network architectures executed limit cycles whose periods increase exponentially with network size.

However, it was not until the mid-1980s that investigations into network dynamics using methods from dynamical systems theory began to be published. Choi and Huberman [33] studied the Little network model [115] (a network of binary units with synchronous update and no enforced symmetry, a precursor to the Hopfield model) in terms of a master equation giving the time-dependent probabilities of the network being in a given state. They demonstrated that a third order approximation to this system could display deterministic chaotic behaviour. Riedel et al. [162] investigated the sequential memory suggested by Sompolinsky and Kanter [190]. This is essentially a Hopfield model with two sets of weights:

2 It is by no means certain that the time series analysed result from attractor dynamics, rather than transient behaviour.

3 For a good introduction to nonlinear dynamics and deterministic chaos see the book by Schuster [179] and the review article by Eckmann and Ruelle [48]

the usual symmetric set and an asymmetric set to induce transitions between patterns. The latter is stabilised by a delay term. They discovered that in the case of exponential delay, an increase in the transition amplitude resulted in chaotic motion. However, this transition was not characterised in terms of any of the well-known routes to chaos. Sompolinsky et al. [189] studied a continuous time network of N nodes with a random weight matrix. Using mean field theory they predicted that there would be a transition from a fixed point (at the origin) to chaotic motion as the gain of the network was increased. This theory is exact as $N \to \infty$. In simulations of several hundred nodes they observed chaotic motion. However, there was no direct transition from the zero fixed point to chaos; there was an intermediate phase where the system displayed non-zero fixed points and limit cycle oscillations. In accordance with the theory, this intermediate region shrunk as N was increased. Additionally an analytic expression for the maximal Liapunov exponent was derived.

Several groups have also characterised deterministic chaotic motion through simulation of neural network models. This has been found in computer and electronic simulation of the network dynamics [98, 15, 9, 124], the memory retrieval dynamics [198, 60, 134] and the learning dynamics of the back-propagation and Hebbian algorithms applied to layered networks[4] [200].

5.2 A survey of network dynamics

5.2.1 The model

Following previous workers (e.g. [7, 202, 180, 35, 80, 150]) an N node network was studied, governed by the dynamical law:

$$\frac{dy_i}{dt} = -y_i + f\left(r\sum_{j=1}^{N} w_{ij}y_j + I_i\right) \tag{5.1}$$

4 In this work the control parameter was the step-size in the gradient descent and the observable was the sum of the absolute values of the weights. The investigators observed period doubling to chaos.

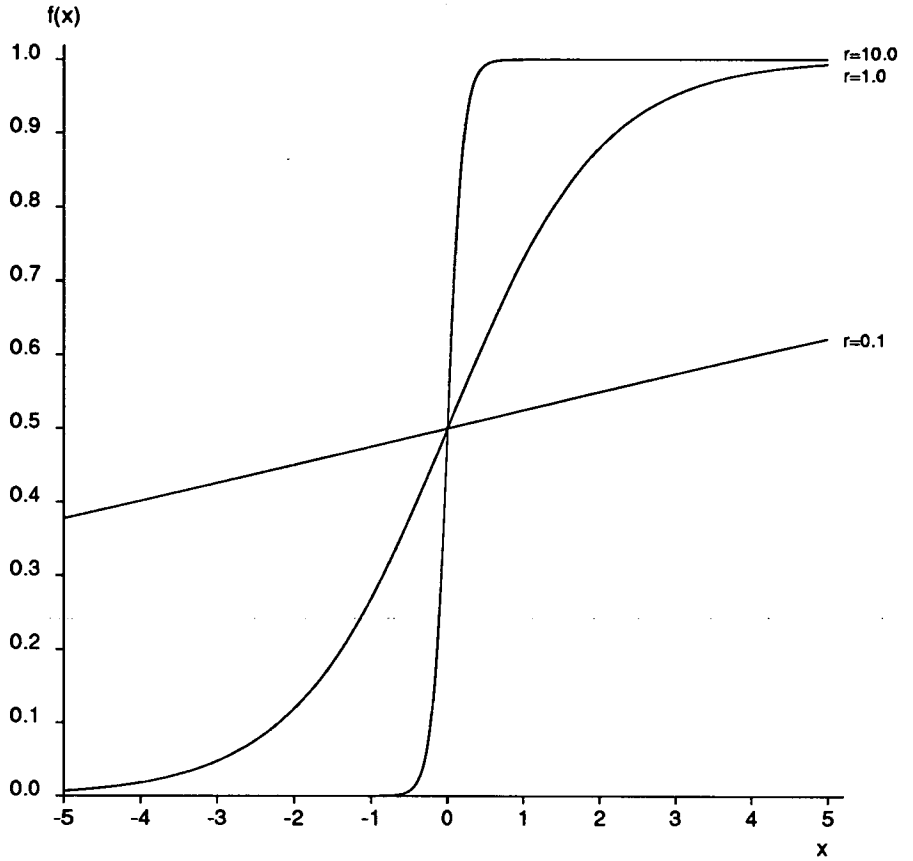


Figure 5–1: Graph of the sigmoid nonlinearity, $f(x) = \frac{1}{1+\exp(-rx)}$, for different values of r.

$$f(x) = \frac{1}{1+\exp(-x)} \tag{5.2}$$

where w_{ij} is an $N \times N$ weight matrix, r is the magnitude of the weight matrix, corresponding to the slope (gain) of the sigmoid nonlinearity f (figure 5–1), y_j is the output of node j and I_i is the external bias. In computer simulation a discretised version of this equation was iterated:

$$y_i(t+\Delta t) = (1-\Delta t)y_i(t) + \Delta t f\left(r\sum_{j=1}^{N} w_{ij}y_j(t) + I_i\right). \tag{5.3}$$

When $\Delta t = 1$ and there is no external input, this is a discrete-time neural network model:

$$y_i(t+1) = f\left(r\sum_{j=1}^{N} w_{ij}y_j(t)\right). \tag{5.4}$$

The studies reported in this chapter and the next used fixed random weight matrices with specified symmetries and magnitudes. An initial random weight matrix v_{ij} was randomly generated so as to produce an RMS activation of 1.0 for each node's input by initialising each weight uniformly in the range $\pm\frac{3}{\sqrt{N}}$. This matrix was then decomposed into symmetric S_{ij} and antisymmetric A_{ij} components:

$$\begin{aligned} S_{ij} &= v_{ij} + v_{ji} \\ A_{ij} &= v_{ij} - v_{ji} \end{aligned} \tag{5.5}$$

and w_{ij} was generated using a symmetry parameter ϵ:

$$\begin{aligned} w_{ij} &= \sigma S_{ij} + \alpha A_{ij} \\ \sigma + \alpha &= 1 \\ \epsilon &= \frac{\alpha}{\sigma}. \end{aligned} \tag{5.6}$$

Four control parameters were used: N (the number of nodes in the network), Δt used to discretely simulate the equations of motion (equation 5.3), τ (the gain of the sigmoid nonlinearity) and a, the weight matrix symmetry parameter, which may be obtained from ϵ:

$$a = \frac{1 - \epsilon^2}{1 + \epsilon^2}. \tag{5.7}$$

So $a = -1$ corresponds to an antisymmetric matrix, $a = 1$ a symmetric matrix and $a = 0$ a random matrix with uncorrelated elements.

Two investigations were performed in an attempt to elucidate the dynamics accessible to this model. In this chapter, a method of cataloging the richness of dynamics offered by this neural network model is described. Networks with from 2–25 nodes and random weight matrices and initial states were investigated via power spectra of their time-dependent output. In the next chapter the results of a detailed study and characterisation of the dynamics of a particular network are reported.

5.2.2 Network power spectra

To examine the dynamical behaviour of an N-node network a power spectrum was computed for the time-dependent output of each node in

the network. After 100 timesteps to run out transients[5] the network was run for a further 1024 timesteps, following which a Fourier transform was computed (using a 1024 point ($T = 1024$) FFT), giving a maximum detectable period of 512

$$P_i(k) = \left| \frac{1}{T} \sum_{t=0}^{T-1} y_i(t) \exp\left(\frac{-2\pi i k t}{T}\right) \right|^2 \qquad k = 0, 1, \ldots, \frac{T}{2}. \tag{5.8}$$

The peaks of the power spectrum were interpreted as specific periods $T(k)$ or frequencies $\omega(k)$

$$\begin{aligned} w(k) &= \frac{2\pi k}{T} & k &= 0, 1, \ldots, \frac{T}{2} \\ T(k) &= \frac{2\pi}{\omega(k)} = \frac{T}{k} & k &= 0, 1, \ldots, \frac{T}{2}. \end{aligned} \tag{5.9}$$

Each power spectrum was summarised into 2 statistics:

- μ_i mean (over frequency) power

$$\mu_i = \frac{2}{T} \sum_{k=1}^{T/2} P_i(k) \tag{5.10}$$

- S_i "spectral entropy", computed from the normalised power spectrum $P_i'(k)$

$$S_i = -\sum_{k=1}^{T/2} P_i'(k) \ln P_i'(k) \tag{5.11}$$

Note that these statistics discard the DC component of the spectrum. The spectral entropy will be maximal for a uniform power spectrum and will be 0 for a spectrum with a single peak. In practice a noise threshold is defined in computing S_i, and the entropy of a fixed point system (with all the power in the DC component) is defined to be 0.

These statistics pertain to nodes; to summarise the activity of the network as a whole they were averaged over all nodes in the network

5 Transients lasting considerably more than 100 iterations may be observed; see later.

to give two mean values $\overline{\mu}$ and $\overline{S}$. The mean power statistic gives an indication of the overall amplitude of the oscillations in the network. This will be large when the network is oscillating between saturated regions of the transfer function — e.g. at large values of r when the nonlinearity approximates a step function. The spectral entropy, which increases with the broadness of the power spectrum, gives an indication of the temporal complexity of the network. Larger values of $\overline{S}$ indicate peaks at several frequencies, suggesting complex periodic motion (with two or more incommensurate frequencies and the consequent harmonics and linear combinations) or chaotic motion (which is characterised by a broadband power spectrum).

In the experiments reported below, one hundred simulations were performed for each network size and for each value of the (r, a) parameter pair; this corresponded to 10 random weight matrices (v_{ij}) and 10 random initial states for each network. For each set of simulations at a particular (r, a) value, means and standard deviations (over simulations) of $\overline{\mu}$ and $\overline{S}$ were computed giving 4 numbers characterising a particular (r, a) parameter region. Initially a value of $\Delta t = 1$ (discrete time system) was investigated; the dependence on varying Δt was investigated with respect to r and a, but N was held constant ($N = 8$, in all experiments when $\Delta t < 1$).

5.2.3 Dynamics of the discrete time system

In this section an investigation of the dynamics of networks specified by the discrete time system (5.4) is reported. Network sizes of 2–25 nodes were studied. The results are shown graphically for a network of size $N = 8$ (figure 5–2). Figure 5–3 shows the spectral entropy with respect to (r, a) for networks with 10 and 20 nodes.

A most striking feature of these results is the qualitative invariance with respect to N of the dynamics of this network model; for $N \geq 4$ the (r, a) phase space in power and entropy may be divided into characteristic

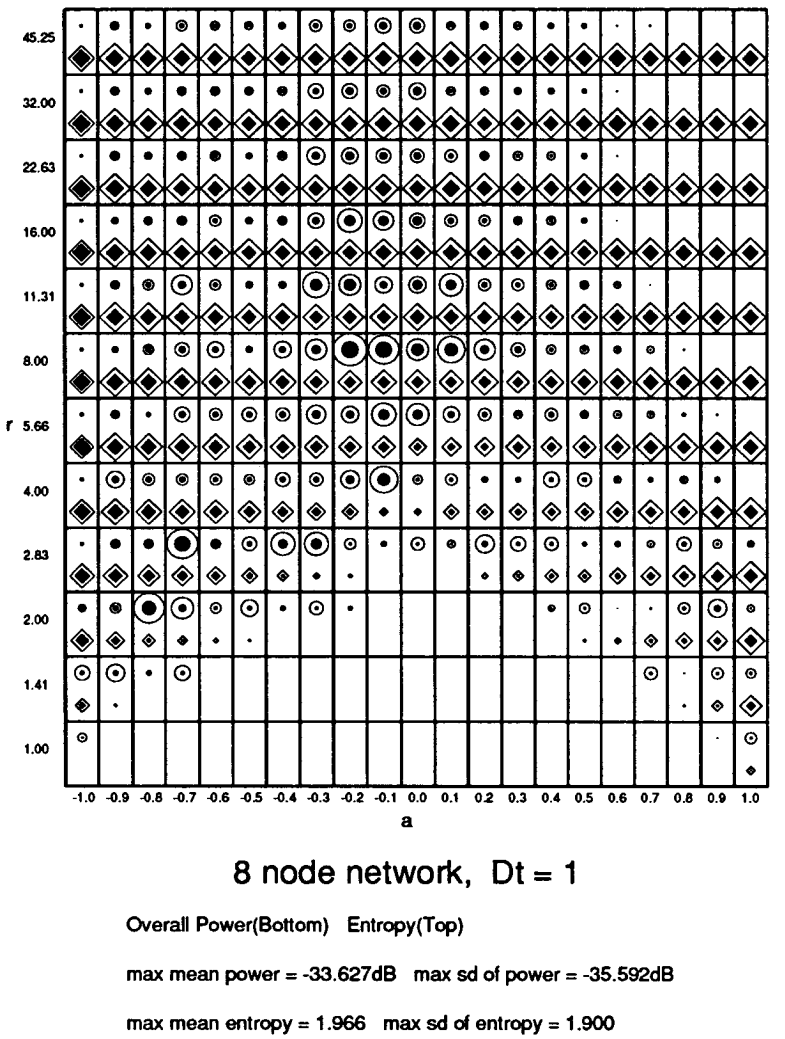
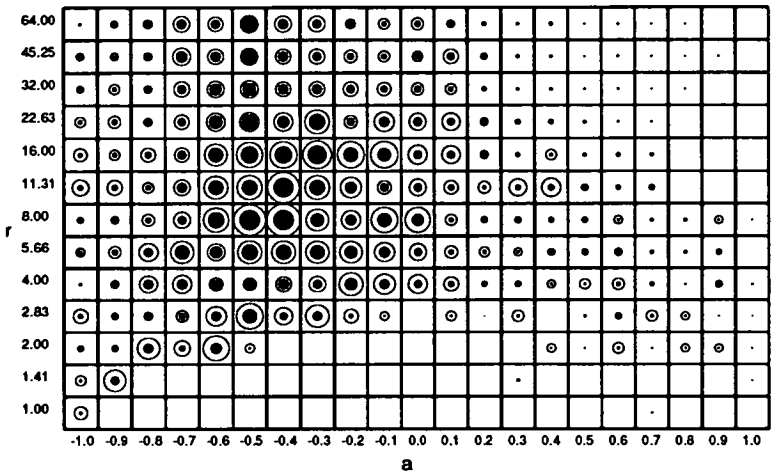


Figure 5–2: Summary statistics from power spectra taken for a network with $N = 8$. Each box represents 100 simulations at a particular (r, a) value. The upper circles correspond to the entropy $\overline{S}$ and the lower diamonds the overall power $\overline{\mu}$. The filled shapes represent means, the surrounding outlines standard deviations, with the scaling corresponding to radius. Statistics were taken over simulations.



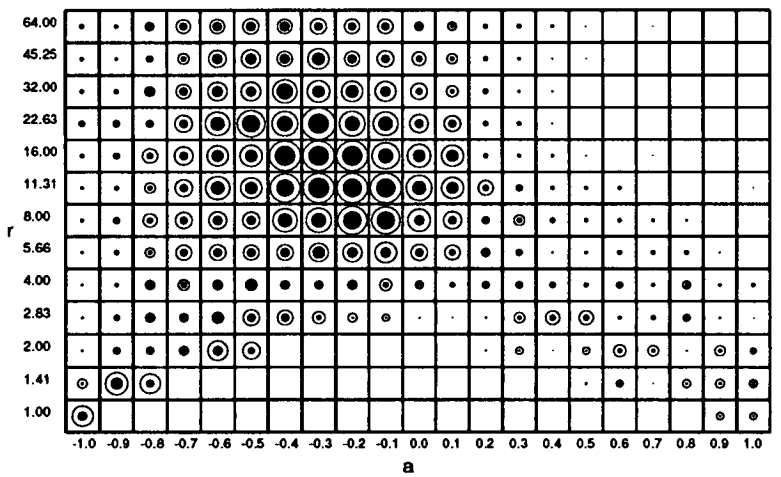
10 node network, Dt = 1

Mean and SD of Entropy

max mean power = -33.942dB max sd of power = -36.139dB

max mean entropy = 2.784 max sd of entropy = 2.000

(a)



20 node network, Dt = 1

Mean and SD of Entropy

max mean power = -33.887dB max sd of power = -36.318dB

max mean entropy = 3.774 max sd of entropy = 2.311

(b)

Figure 5–3: Means and standard deviations (over simulations) of average entropy $\overline{S}$ for networks with (a) N = 10 and (b) N = 20.

regions. At low values of τ the transfer function is virtually linear and the weight matrix has low magnitude; hence, the eigenvalues will usually fall inside the unit circle and the network will display fixed point behaviour. When τ is large, the transfer functions approximates a step function; for a symmetric network ($a = 1$) this is the symmetric Little model and it can be shown analytically that the system will display fixed point or limit cycle period 2 dynamics[6][59]. For an antisymmetric weight matrix ($a = -1$) it may be shown that fixed point or limit cycle period 4 dynamics will emerge [72].

Away from these extremes, more complex dynamics with longer temporal correlations may be observed. In figure 5–3 these areas are indicated by large values of the mean and standard deviation of the entropy. A high region of activity may be located at approximately $4 \leq \tau \leq 32$ (i.e. "intermediate" values of the transfer function, away from the linear and step function limiting behaviours) and at approximately $-0.8 \leq a \leq 0.0$ (i.e. uncorrelated networks or networks with a greater antisymmetric component).

When temporal activity first occurs in the network (as τ is increased) high spectral entropies are recorded, but with very low power values. It seems likely that this low power activity is not due to attractor motion, but to long-lived transients (transients were assumed to die out in 100 iterations, which in some cases appears to be an optimistic assumption). This is understandable as it may correspond to an initial bifurcation away from fixed-point behaviour. As the system approaches the bifurcation point, so the maximal Liapunov exponent will approach 0. Qualitatively this means that the rate of convergence to the final attractor becomes slower as the bifurcation point is approached, leading to long-lived transients. Numerical noise arising from the accuracy of the floating point processor[7] may be discounted, as the power levels are too

6 Marcus and Westervelt [125] have provided a stability criterion for the gain that will guarantee only fixed point attractors in this case

7 The simulations in this chapter were performed using 32 bit floating point arithmetic.

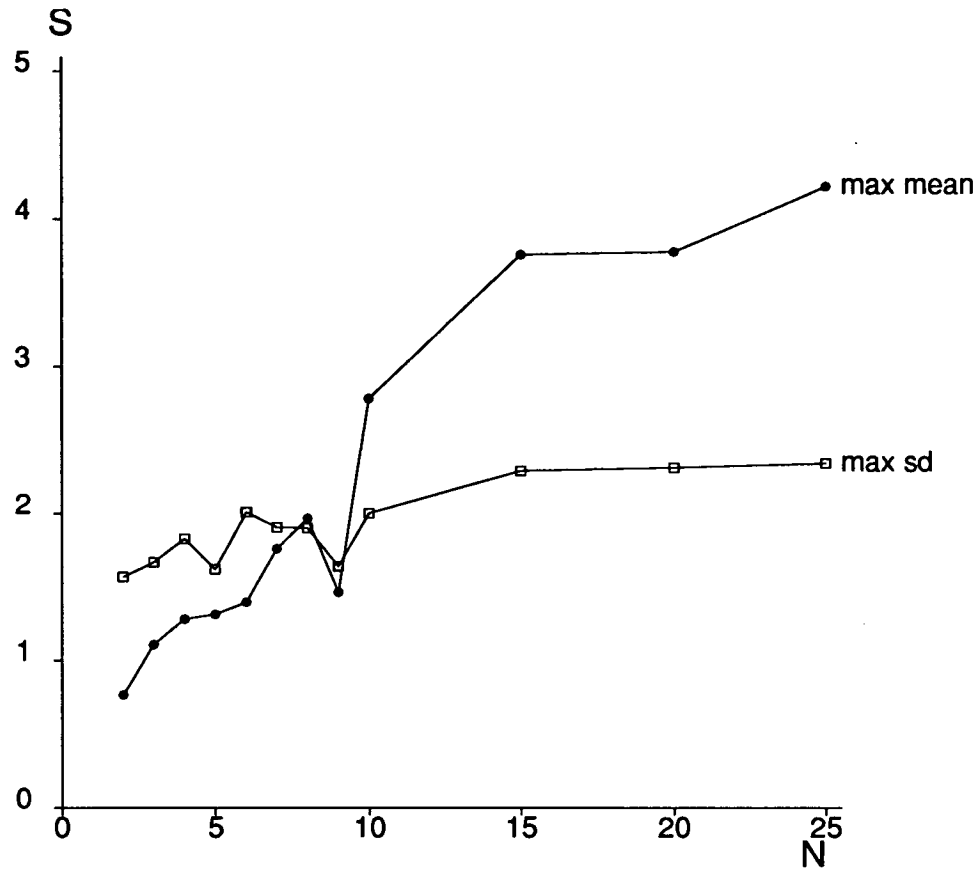
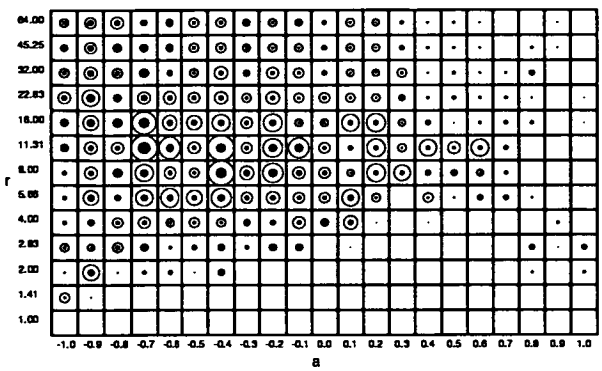


Figure 5–4: Graph of maximum (over (a, r)) mean and standard deviation of entropy ($\max\langle\overline{S}\rangle$ and $\max\sigma(\overline{S})$) against N. The maximum value of $<\overline{S}>$ is $\ln(\frac{T}{2}) = \log(512) = 6.23$.

high.

Although the qualitative features of the dynamics phase space appear to be independent of N (provided $N \geq 4$), there is a trend of increasing dynamical complexity as N increases. This is evidenced by plotting the maximum (over a and r) mean (over simulations) spectral entropy, $\max <\overline{S}>$, against N (figure 5–4).

In the next chapter 64 bit floating point arithmetic was used.



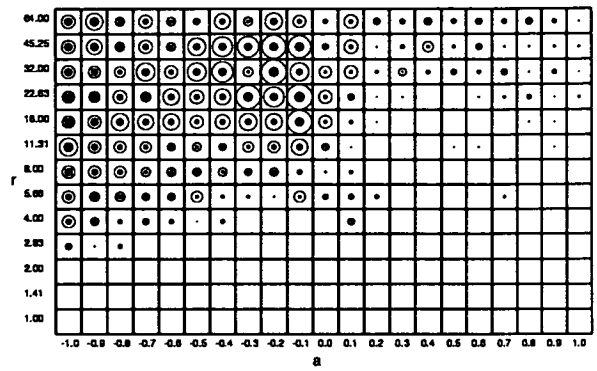
8 node network, Dt = 1

Mean and SD of Entropy

max mean power = -7.532dB max sd of power = -9.943dB

max mean entropy = 1.307 max sd of entropy = 1.875

(a)



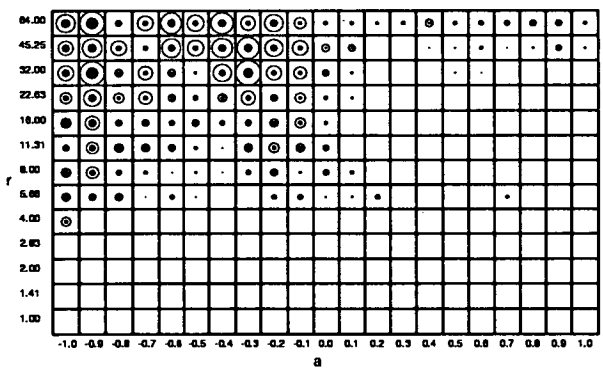
8 node network, Dt = 0.5

Mean and SD of Entropy

max mean power = -14.116dB max sd of power = -16.070dB

max mean entropy = 1.364 max sd of entropy = 2.070

(b)



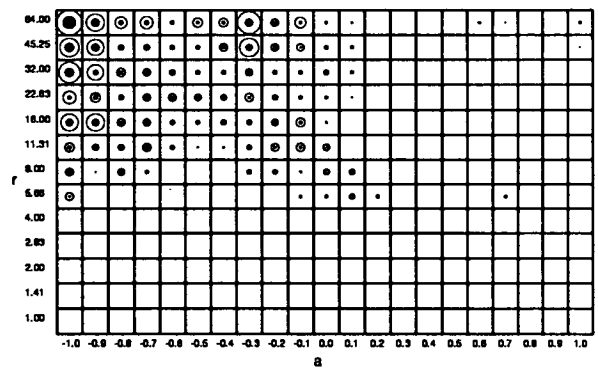
8 node network, Dt = 0.25

Mean and SD of Entropy

max mean power = -17.127dB max sd of power = -18.227dB

max mean entropy = 1.468 max sd of entropy = 2.007

(c)



8 node network, Dt = 0.125

Mean and SD of Entropy

max mean power = -21.213dB max sd of power = -19.724dB

max mean entropy = 1.621 max sd of entropy = 1.824

(d)

Figure 5–5: Summary statistics from power spectra taken from 8 node networks, with (a) $\Delta t = 1.0$, (b) $\Delta t = 0.5$, (c) $\Delta t = 0.25$ and (d) $\Delta t = 0.125$. The diagram uses the same representation as figure 5–3.

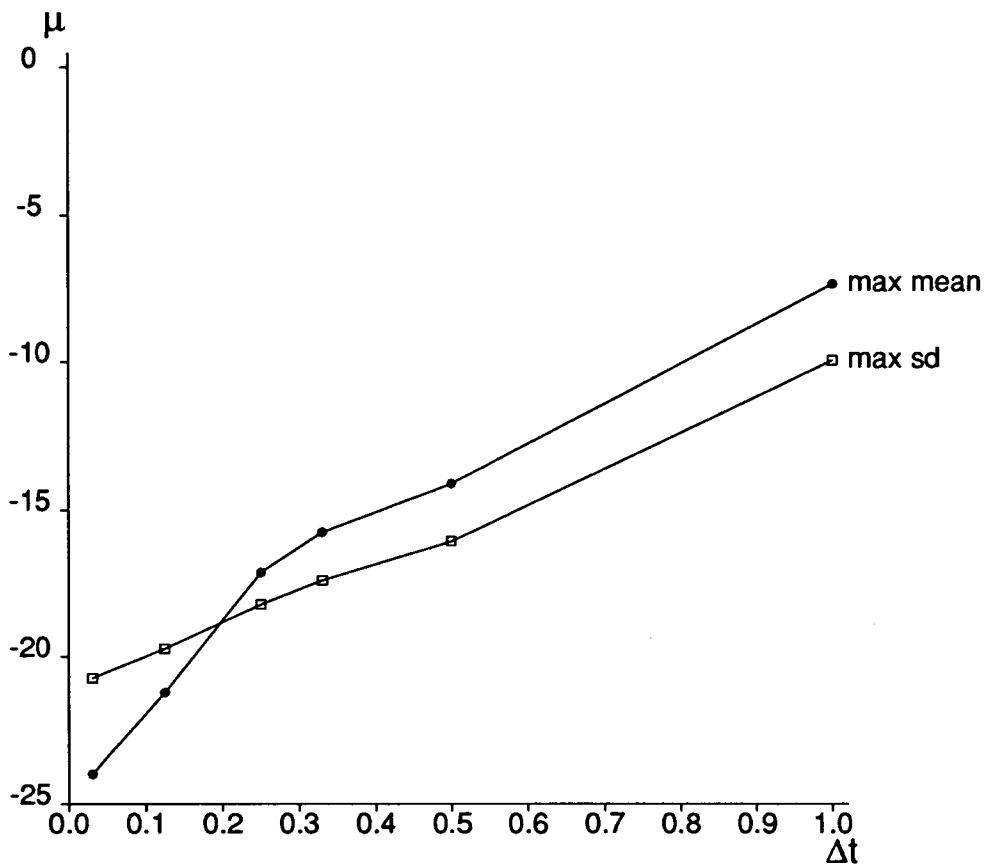


Figure 5–6: Plot of maximum (over (a, r)) mean and standard deviation of power ($\max\langle\overline{\mu}\rangle$ and $\max\sigma(\overline{\mu})$) against Δt.

5.2.4 Discrete and continuous modelling

The above section does not investigate the effect of the resolution of the discretisation (i.e. the value of Δt in equation 5.3) on the dynamics of the network. To study this effect, the dynamics of an 8 node network[8] were surveyed at various values of Δt (figure 5–5). As Δt was reduced (i.e. the system better approximated the continuous time limit) so the areas of fixed point dynamics in (r, a) parameter space increased. Specifically, when $\Delta t = 1$ and $r > 2.0$ most networks displayed non-fixed point behaviour for all values of the symmetry, a. (For positive a, limit cycles of

8 Owing to the extensive computer time required to investigate the dependence on Δt, this survey was only carried out for a single value of N.

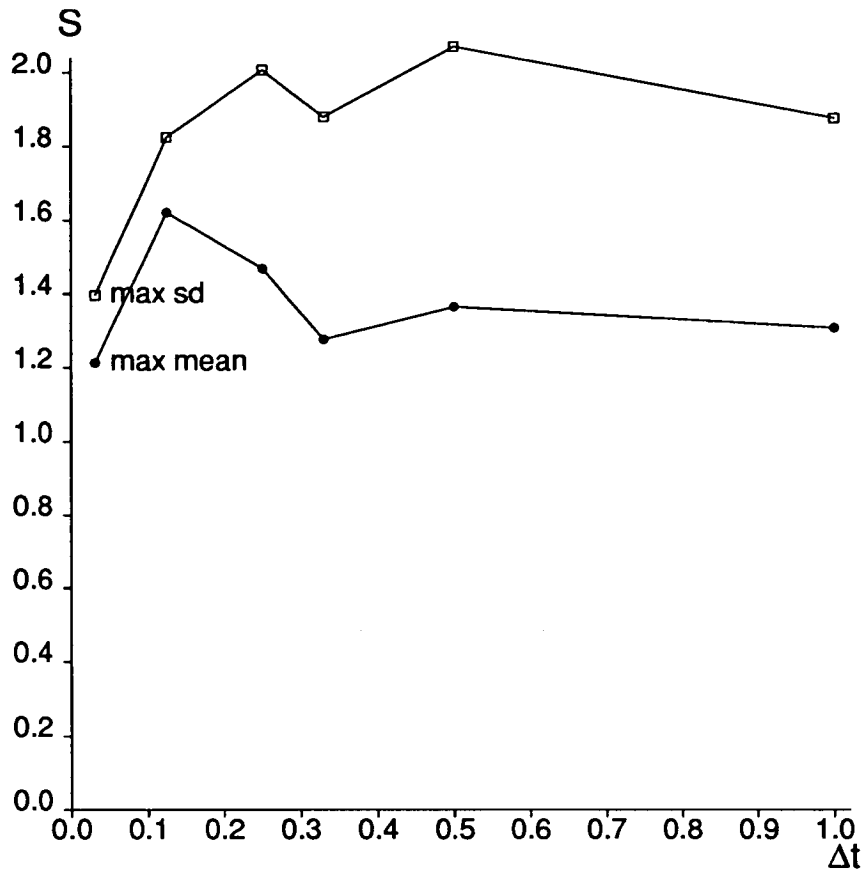


Figure 5–7: Plot of maximum (over (a, r)) mean and standard deviation of entropy ($\max\langle\overline{S}\rangle$ and $\max\sigma(\overline{S})$) against Δt.

period 2 were the most common attractor types.) However, as Δt was reduced, larger areas of (r, a) space displayed fixed point dynamics, particularly in regions where a was positive. This stabilisation effect is to be expected, as decreasing Δt will have a damping effect on the dynamics, arising from the first term on the right hand side of (5.1). Additionally, as Δt is decreased the power of any oscillatory motion is also decreased, as may be observed by plotting the maximum (over r and a) mean and standard deviations of power against Δt (figure 5–6). Although the power of the oscillations is decreased with Δt, the maximum spectral entropy does not; if anything it increases. Furthermore, the region of maximal entropy drifts to larger r and more negative a values as Δt is decreased

(figure 5–7).

The stabilising effect of reducing Δt is to be expected [32] and is in line with results reported by other researchers. Pineda [150] has suggested that values of Δt marginally below 1 impart stability into the network (he typically used $\Delta t = 0.99$ or $\Delta t = 0.90$ in simulations of (5.1) when fixed point behaviour was required). Simard et al. [185] reported that after numerical simulating (5.1) only 2 out of 500 100-node networks with weights randomly initialised uniformly in the range ± 1 (corresponding to $\tau = 3.33$) did not display fixed point behaviour[9]. Additionally they reported that all networks of 50 or fewer nodes simulated displayed fixed point behaviour (with a gain corresponding to $\tau = 2.36$). These results are in regions of low τ; however, complex dynamical activity has been reported by other workers investigating the behaviour of (5.1). Kürten and Clark [98] have reported chaotic behaviour in such a system with greater than 25 nodes and a limited fan-in of 8 connections inputting to each node. Pearlmutter [148] has trained small networks governed by dynamical law (5.1) to display complex limit cycles.

9 Simard et al. do not give the value of Δt they used for the discretisation of (5.1).

6

CHAOS IN NEURAL NETWORKS

The survey of network dynamics reported in the previous chapter gave some indication of the dynamics displayed by this neural network model. However, the survey threw little light on the detailed structure of these dynamics. In an attempt to rectify this situation, the dynamics of a particular network were investigated in some detail. The control parameters were again the gain (r), symmetry (a), network size (N) and discretisation (Δt). A random network was simulated, using $N = 8$ and $\Delta t = 1$; hence this was an 8 node discrete-time model given by (5.4)[1]. Thus, the two variable control parameters used were r and a.

6.1 Methodology

The first tools used to investigate the dynamics of a network were bifurcation diagrams plotted for each node. Each bifurcation diagram showed the output of that node with respect to a single control parameter (r or a). More quantitatively, the dependence of the dynamics on the control parameter was investigated using Liapunov diagrams, a plot of the Liapunov exponents of the attractor against the control parameter. Other methods used to characterise attractors included power spectra of the node outputs, computations of fractal dimension and 2-dimensional attractor sections obtained by plotting the output of one node against that

1 Jan Scheurich [178] has performed some similar computational studies of networks simulated with higher resolution discretisation, $\Delta t = 0.75$ and $\Delta t = 0.25$.

of another.

Characterising fixed points and periodic motion is trivially achieved using power spectra: the peaks in a power spectrum correspond to the frequency of motion (or its harmonics). A power spectrum is not enough to characterise chaotic motion: whilst deterministic chaotic motion is distinguishable from purely periodic motion by virtue of its broadband power spectrum, it is not distinguishable from noisy periodic or random motion. However, deterministic chaos is very different from random noise: chaos is characterised by a deterministic dynamics, in which initially adjacent points become exponentially separated as the dynamics evolve. In random motion, initially adjacent points are distributed with equal probability over the entire region of accessible space. This exponential divergence may be measured, on average, for an attractor by computation of the Liapunov exponents.

A second way to distinguish chaotic motion from random motion is by the fractal dimension. Whereas random motion in a d-dimensional space will tend to fill that space, chaotic motion will remain bounded in a certain region of space. The structure of this region to which the attractor is confined may be described by the fractal dimension.

6.1.1 Liapunov exponents

Consider the motion of 2 adjacent points under the influence of a 1-dimensional discrete dynamical system, $x_{n+1} = F(x_n)$. If the system is chaotic, then the 2 points will become exponentially separated (figure 6–1). This separation may be measured by the Liapunov exponent ($\lambda(x_0)$), which is defined as:

$$\begin{aligned} \lambda(x_0) &= \lim_{N\to\infty} \lim_{\epsilon\to 0} \frac{1}{N} \log \left| \frac{f^N(x_0+\epsilon) - f^N(x_0)}{\epsilon} \right| \\ &= \lim_{N\to\infty} \frac{1}{N} \log \left| \frac{df^N(x_0)}{dx_0} \right| . \end{aligned} \tag{6.1}$$

This may be generalised to a d-dimensional system. Such a system possesses d Liapunov exponents, one for each spatial dimension, generally

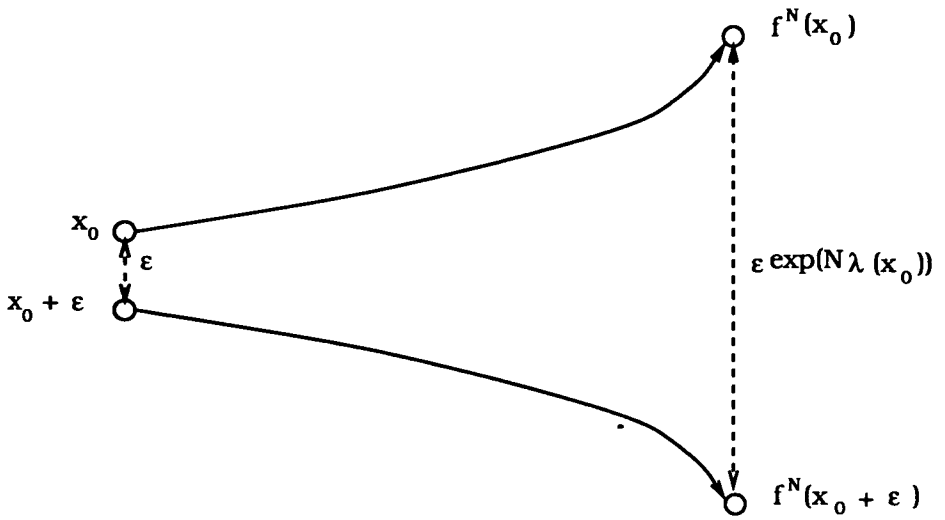


Figure 6–1: The Liapunov exponent (λ) measures the exponential separation of two points acted on by a chaotic dynamical system.

ordered largest first:

$$(\lambda_0, \lambda_1, \ldots, \lambda_d) = \lim_{N\to\infty} \frac{1}{N} \log(\text{magnitude of the eigenvalues of} \prod_{n=1}^{n=N} J(x_n)) \tag{6.2}$$

where $J(x)$ is the Jacobian matrix of the map $x_{n+1} = F(x_n)$:

$$J_{ij}(x) = \frac{\partial F_i}{\partial x_j}. \tag{6.3}$$

Liapunov exponents are invariant under phase space coordinate transformations.

Intuitively, the Liapunov exponents give an indication of the stretching and compressing in the attractor. Negative Liapunov exponents indicate that the system is contracting — adjacent points will be drawn together, onto the attractor. Chaotic motion is characterised by at least one positive Liapunov exponent; a positive Liapunov exponent indicates that the system is divergent in that direction. This divergence is manifested as a sensitive dependence on initial conditions, meaning that information about the initial conditions is lost under the evolution of the dynamics. This information loss may be measured by the Kolmogorov entropy which is simply the sum of the positive Liapunov exponents [53].

Computation of the Liapunov exponents for a multi-dimensional system cannot generally be achieved by a trivial substitution into formula (6.2). Computing the Jacobian in this formula would involve N nested matrix products, where N is typically of the order of 10^3 or 10^4. This has the effect of causing a single direction (corresponding to the largest eigenvector) to dominate, causing the other basis vectors to collapse and become indistinguishable. Additionally each vector will converge to 0 or diverge to ∞ in magnitude. Both these problems may be overcome by following a method originally proposed by Benettin and co-workers [17, 18] that is based around a repeated Gram-Schmidt Orthogonalisation. This procedure is repeated each timestep: as well as normalising the basis vectors, it subtracts the components of 'higher' (larger eigenvalue) basis vectors from lower ones. This technique enabled the accurate calculation of 7 Liapunov exponents for the 8-dimensional system under consideration here.

6.1.2 Fractal dimension

The Liapunov exponents give information about the local "stretching" in the attractor, but give no information about the global "folding". There have been several definitions of attractor dimension, that attempt to give static topological information about the attractor. Three of the most common dimensions are the fractal or Hausdorff dimension [123], the information dimension [53] and the correlation dimension [68]. The fractal dimension is a purely geometric quantity, independent of the frequency with which various parts of the attractor are visited, and is computed by considering the number of d-dimensional boxes of side length l needed to cover the attractor, considering the limit $l \rightarrow 0$. The information dimension takes account of non-homogeneity in the attractor by considering the probability that a point on the attractor will fall in a given box of size l^d. The correlation dimension (used in this work) is computed via a ball-scaling method, by considering the spatial correlations between pairs

of points on the attractor via the correlation integral $C(l)$, the fraction of pairs of points less than a distance l apart:

$$C(l) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{ij} \theta(|x_i - x_j| - l) \tag{6.4}$$

where θ is the Heaviside step function. Grassberger and Procaccia established a scaling law for $C(l)$:

$$C(l) \propto l^{D_2} \tag{6.5}$$

when l is small. D_2 is the correlation dimension.

A relationship between the various definitions of dimension was established by Halsey et al. [73] with the notion of generalised dimension, D_q, $-\infty \leq q \leq \infty$:

$$D_q = -\lim_{l\to 0} \frac{1}{q-1} \left|\frac{1}{\log l}\right| \log\left(\sum_i p_i^q\right), \tag{6.6}$$

where p_i is the probability that the attractor visits box i of side length l. Schuster [179] gives a lucid exposition of how this is related to the ball-scaling method of Grassberger and Procaccia, and demonstrates that the fractal dimension is D_0 (by definition, the maximal D_q), the information dimension is D_1 and the correlation dimension D_2.

6.2 Gain dependence

To investigate the effect of gain on the dynamics of an individual network ($N = 8$) the symmetry parameter was held constant ($a = 0$) giving the weight matrix equal symmetric and antisymmetric parts. The dynamics were investigated by varying r from 0 to 31.1: in effect from a linear transfer function to an approximated step transfer function.

The dynamics may be viewed using bifurcation diagrams for the outputs of each node. These were constructed by running the system for 20,000 iterations at each parameter value and recording the last 10,000 states of the network, which were presumed to be on the attractor. Figure 6–2 shows a bifurcation diagram for one node of the network for

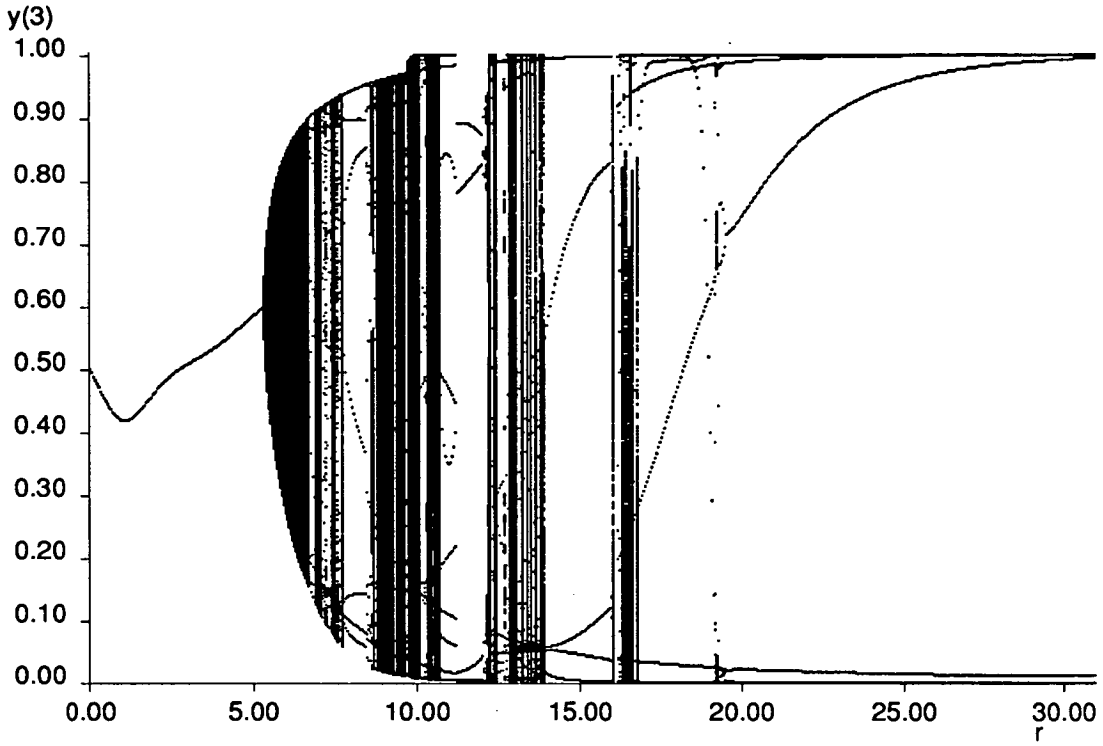
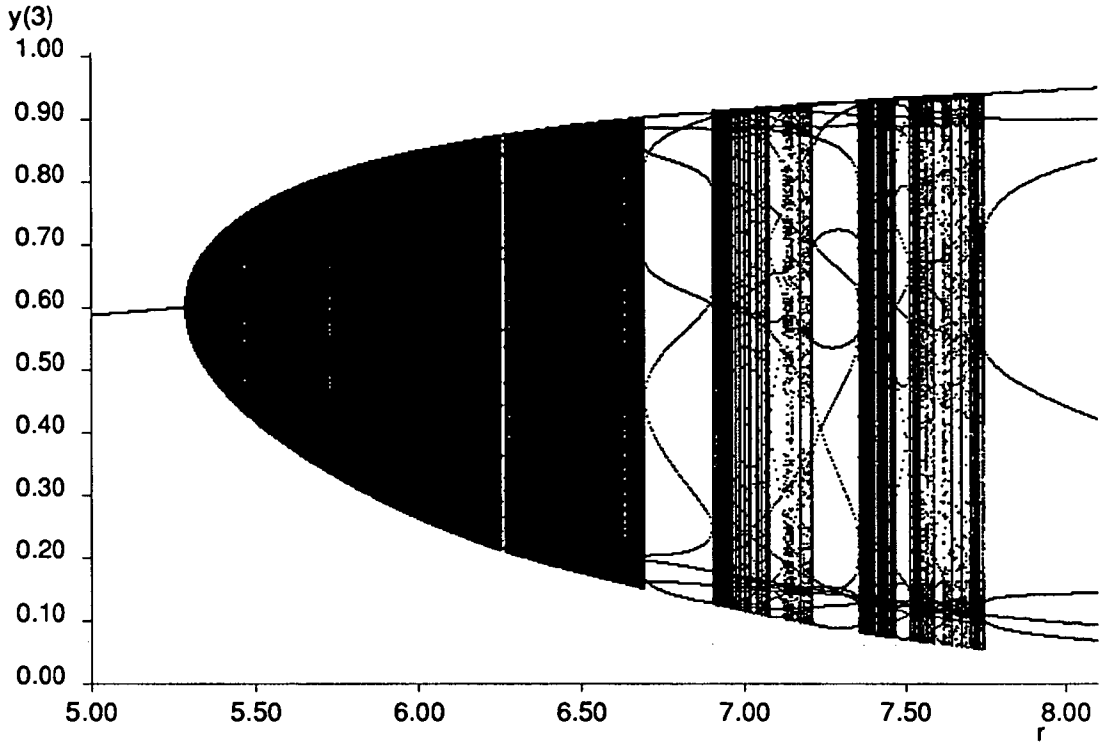


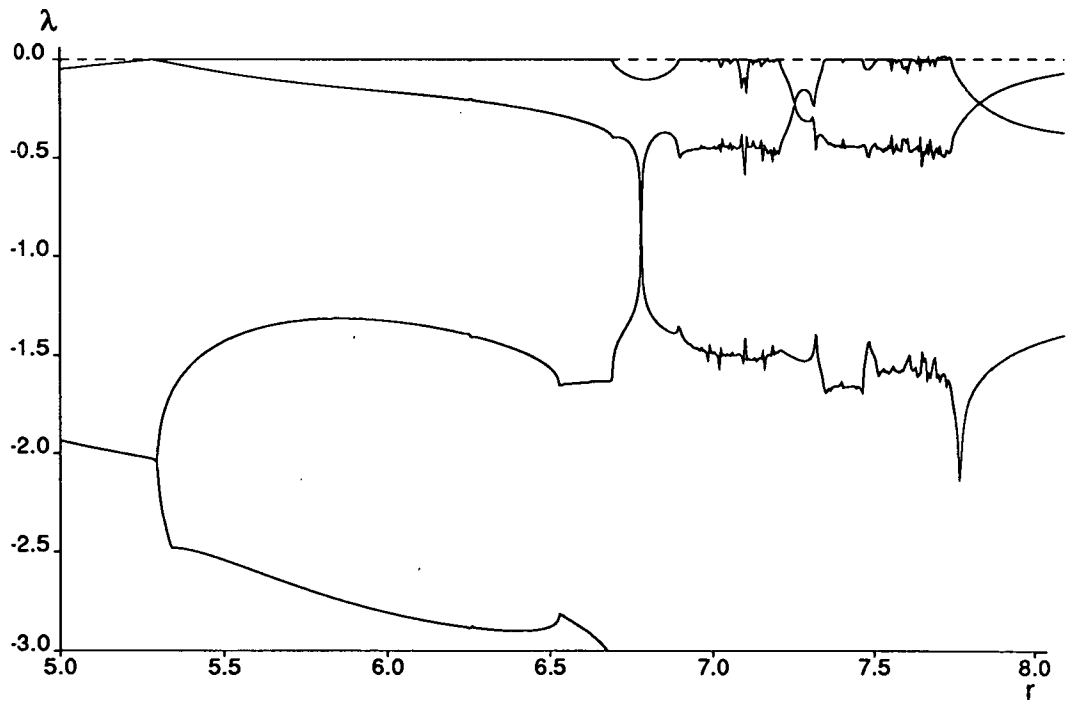
Figure 6–2: Bifurcation diagram for one node of an 8 node network ($a = 0.0$) with respect to varying r ($0.0 \leq r < 31.0$). One column of the diagram represents 10,000 iterations of the network (after 10,000 iterations to run out transients) at a particular value of r. The y axis represents the output of an individual node of the network.

$0.0 \leq r < 31.0$, with r increasing in steps of 0.05. At low values of r the system is close to linear, and thus displays fixed point behaviour. When $r \approx 5.3$ the system bifurcates to oscillatory behaviour; non-trivial oscillatory behaviour continues until $r > 20$, when a limit cycle (period 4) attractor dominates, the activity of all nodes in the network moving toward the saturated regions of the sigmoid nonlinearity, thus producing large amplitude oscillations.

The region around the initial Hopf bifurcation (when $r = 5.2822$) is shown in greater detail in figure 6–3(a). As r increases toward the bifurcation point the (negative) maximal Liapunov exponent λ_1 increases toward zero (figure 6–3(b)). For fixed point dynamics, λ_1 gives a qualitative measure of the rate of convergence of points in the basin of attraction



(a)



(b)

Figure 6–3: (a) Bifurcation diagram for one node of an 8 node network, together with (b) corresponding Liapunov diagram for $5.0 \leq r \leq 8.0$, when $a = 0.0$. The regions of mode-locking at winding numbers $\frac{3}{11}$, $\frac{5}{18}$ and $\frac{2}{7}$ may be observed.

to the fixed point: as λ_1 becomes less negative, so this rate decreases and initial points take longer to converge to the fixed point. This was observed in the transient behaviour of the dynamics as $\tau \to 5.2822$: long lived transients of up to 10,000 iterations were observed.

A second bifurcation from 1 frequency to 2 frequency motion occurs at $\tau = 5.2840$: this is not visible in the bifurcation diagram, but is observable in the power spectra (figures 6–4(a,b)). A 2-dimensional section of the resultant attractor (taken when $\tau = 6.0$; the oscillations are of greater amplitude, but are qualitatively similar to the motion immediately after the bifurcation) shows a circular geometry (figure 6–5(a)); this may be reparameterised in terms of the angle (θ_n) of each point on the circle with respect to the centroid of the circle. The angular return map (or "θ-map"), with the angles taken mod 1 in a circle of 2π, has a correspondingly linear form (figure 6–5(b)). This system has similar dynamics to the sine-map studied by Jensen et al. [87]:

$$\theta_{n+1} = \theta_n + \Omega - \frac{k}{2\pi}\sin(2\pi\theta_n) \bmod 1. \tag{6.7}$$

The motion of this system may be characterised by the winding number,

$$W = \lim_{n\to\infty}\left(\frac{\theta_n - \theta_0}{n}\right), \tag{6.8}$$

which measures the average rotation per iteration: here θ_n is not taken mod 1. The value of the winding number may also be obtained from the dominant peak of the power spectrum (figure 6–4(c)). If W is plotted against τ (figure 6–6), then a characteristic devils staircase structure is obtained. Each step of the self-similar staircase corresponds to a rational winding number at which the system is mode-locked. This structure may be generated by the Farey tree of rational numbers, with stronger mode-locking occurring with simpler (closer to the root of the Farey tree) rational numbers. Mode-locking is a common physical phenomenon, first observed by Huygens, when two resonators oscillating at different frequencies shifted frequency when coupled so as to oscillate at a simpler

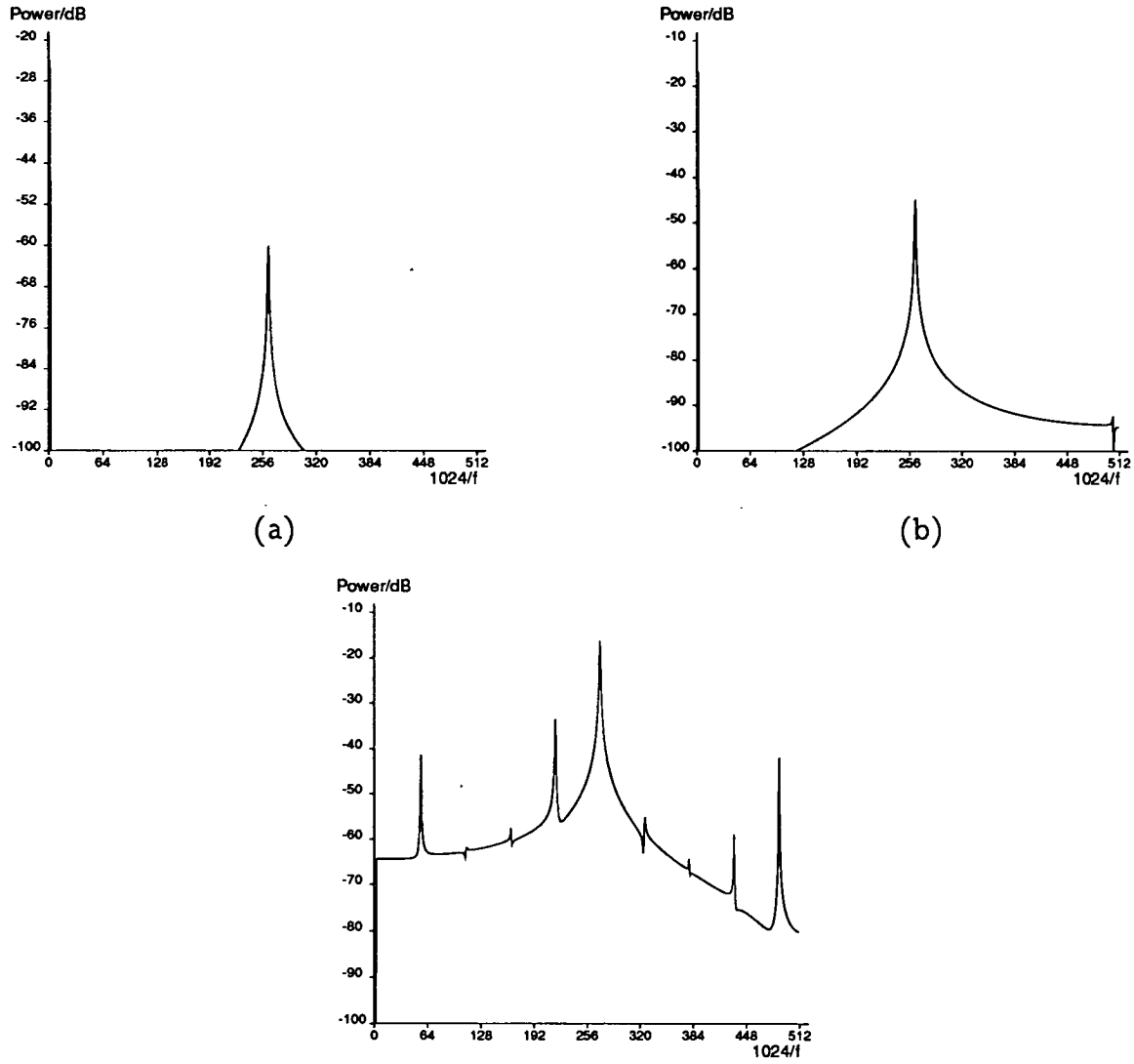
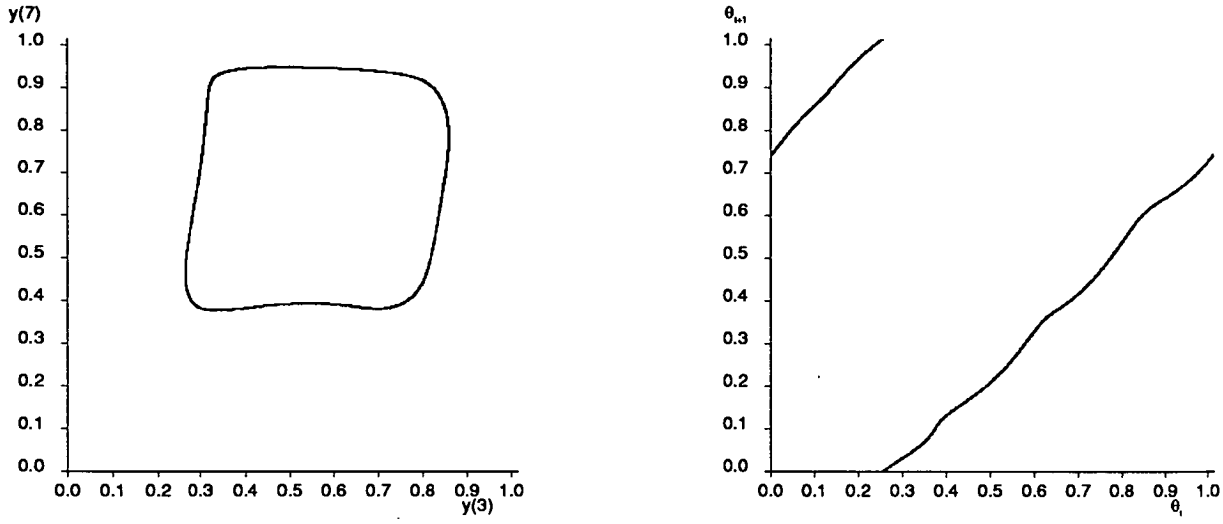


Figure 6–4: Power spectrum of the output of one node of an 8 node network taken at (a) $r = 5.2822$, (b) $r = 5.2840$ and (c) $r = 6.0$ with $a = 0.0$. In (a) there is single frequency motion; there is a bifurcation to 2 frequency motion resulting in a second peak in the power spectrum (b). This can be seen more clearly in (c), where various harmonics and linear combinations of the two frequencies are apparent in the power spectrum.



(a) (b)

Figure 6–5: (a) Attractor section y_7 vs. y_3 and (b) corresponding θ-map when $r = 6.0$ and $a = 0.0$. Note the linearity of the θ-map.

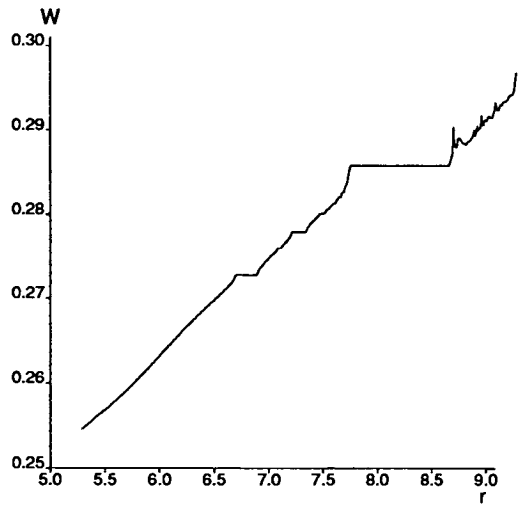


Figure 6–6: Graph of winding number W vs. r when $a = 0.0$. Mode-locking is apparent in the self similar devils staircase structure: note that $\frac{5}{18}$ is the Farey child of $\frac{3}{11}$ and $\frac{2}{7}$. The breakup of the staircase indicates a transition to chaos.

ratio of frequencies. The sine map (6.7) is perhaps the simplest system to display this phenomenon.

Mode-locking may also be observed in the Liapunov diagram (figure 6–3(b)). In non-mode-locked regions, the winding number is irrational[2] and these regions are characterised by a zero λ_1. Mode-locked regions have a negative λ_1. This may be understood by recalling the definition of Liapunov exponents. The Liapunov exponents are calculated by considering the behaviour of an infinitesimal ball centred about a point on the attractor. If there is a finite number of points on the attractor (which is the case for mode-locked motion) then the largest Liapunov exponent must be negative, since an infinitesimal ball will evolve to a single point under the dynamical law. However, an irrational winding number corresponds to an attractor which visits an infinite number of points on a torus; in this case the ball will not collapse to a point under the dynamical law and the largest Liapunov exponent will be zero, corresponding to an eigenvector directed along the flow.

The correlation dimension of the attractor was computed after generating 32,000 points assumed to be on the attractor (having discarded a previous 10,000 points assumed to be transients) using the method outlined in section 6.1.2. This computation was not accurate as the graph of $C(l)$ vs. $\log(l)$ was not linear (figure 6–7). Estimating the gradient of the part that most approximated a linear section (by eye) yielded a value of $D_2 \approx 1.6$. However, the previously presented evidence of the Liapunov exponents and the power spectra indicate that this was a (multi-frequency) periodic system, that would be expected to have an integer dimension: in fact it is to be expected that $D_2 = 2$ if this is two-frequency motion on a torus. The inaccuracy of this dimension computation is not unexpected: Ramsey and Yuan [157] have indicated that the cited er-

2 This is an approximation since: (1) there may be mode-locking at a rational number deep in the Farey tree, which at a given resolution is indistinguishable from an irrational number; and (2) all floating point numbers represented on a digital computer are rational anyway.

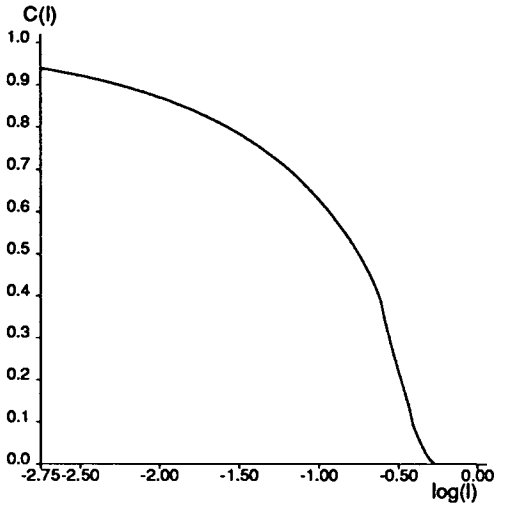


Figure 6–7: Graph of the correlation integral $C(l)$ plotted against $\log(l)$ for $r = 6.0$ and $a = 0.0$. The slope of the linear region of this graph should equal the correlation dimension, D_2.

rors for many empirically determined data sets are under-estimates by a factor of ten or more.

When $r = 6.9$ another bifurcation occurs and a third incommensurate frequency is introduced (figure 6–8). However, the system cannot sustain three-frequency motion: the toroidal form of the attractor breaks up and a transition to chaos occurs. This behaviour is characteristic of the Ruelle-Takens-Newhouse route to chaos [135]. The destruction of the torus is indicated by backfolding in an attractor section (figure 6–9(a)) and the corresponding θ-map (figure 6–9(b)) which contains a cubic point of inflection. As r is further increased, more backfolding occurs and the system becomes more turbulent. The breakup of the toroidal attractor may also be observed from the behaviour of the winding number (figure 6–6), with the devils staircase breaking up when the system becomes chaotic. Chaotic motion may be evidenced from the Liapunov diagram (figure 6–3(b)) — the largest Liapunov exponent fluctuates around 0, sometimes becoming positive, and from the low-power broadband activity in the power spectrum (figure 6–10). Figure 6–11 shows a section of a more turbulent attractor, at a greater value of r. The Liapunov diagram

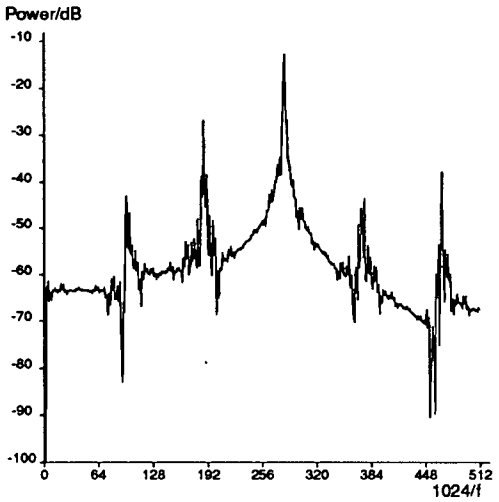
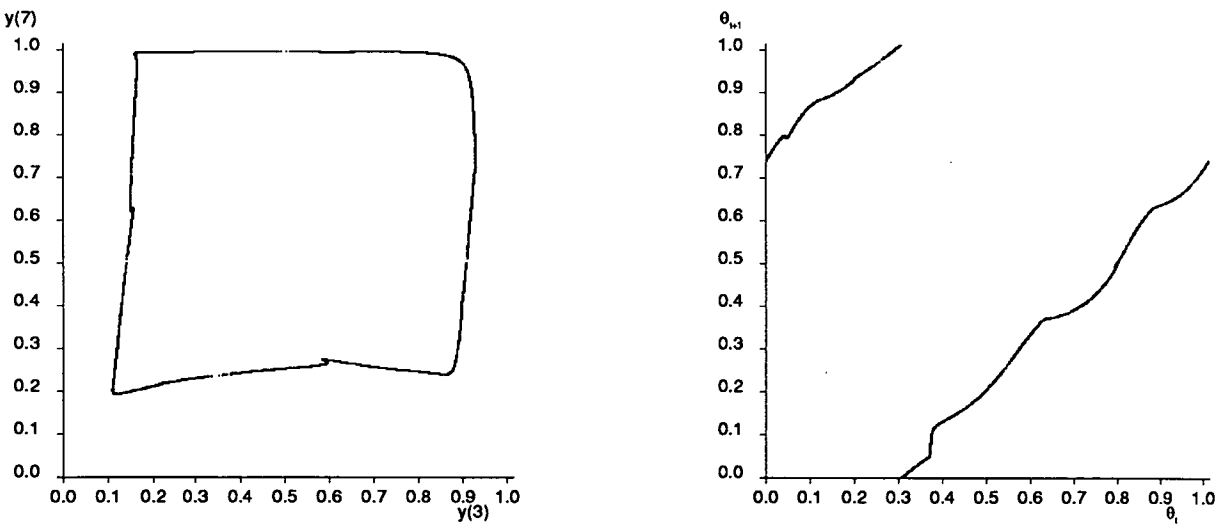


Figure 6–8: Power spectrum when $r = 6.9$ and $a = 0.0$. This occurs after a third bifurcation has imparted a new frequency into the system. The system is not stable enough to sustain three frequency motion, and chaos ensues — as evidenced by the broadband noise.



(a) (b)

Figure 6–9: (a) Attractor section and (b) θ-map at $r = 7.07$, $a = 0.0$. The destruction of the torus may be seen, by the back-folding in the attractor section and the cubic point of inflection in the θ-map.

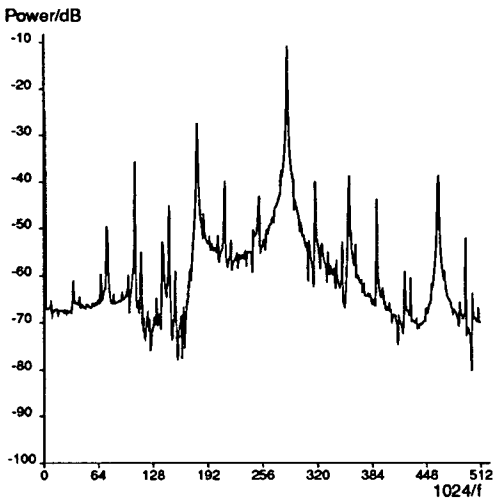


Figure 6–10: Power spectrum of a single node at $r = 7.07$ and $a = 0.0$. Chaotic motion is indicated by the broadband noise in the spectrum.

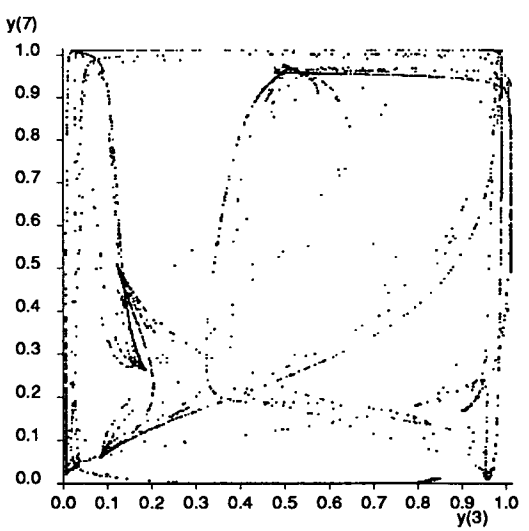


Figure 6–11: Section of the chaotic attractor at $r = 10.69$ and $a = 0.0$. No real structure is visible in this 2-dimensional projection of the attractor.

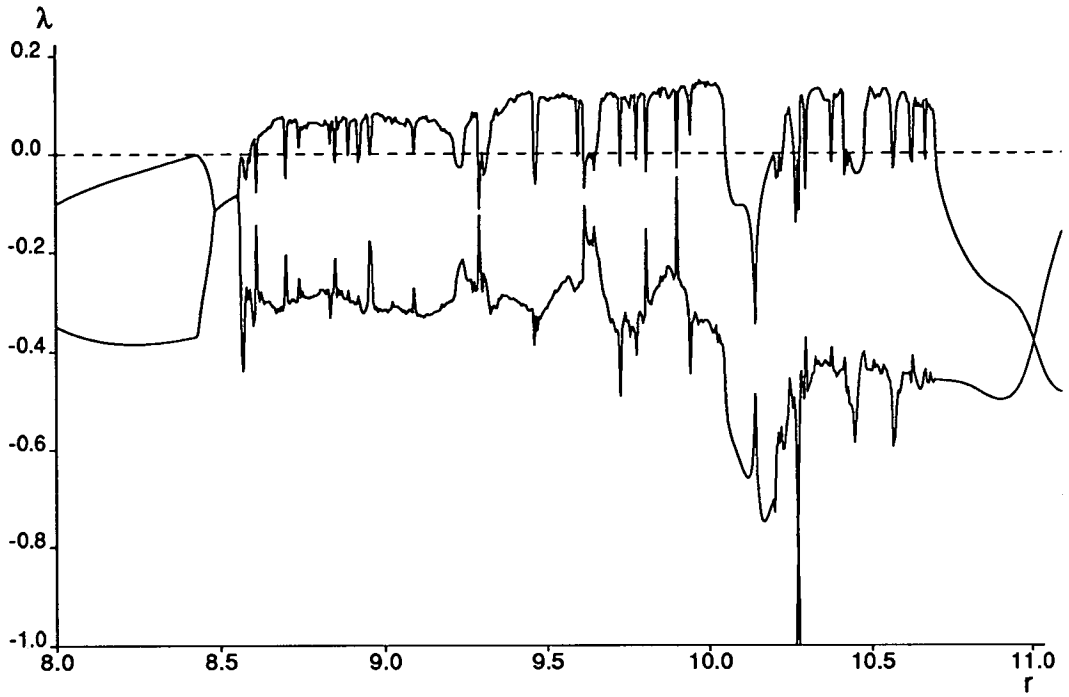


Figure 6–12: Liapunov exponents for the region $8.0 \leq r < 11.1$, when $a = 0.0$. The generally positive maximum Liapunov exponent exponent indicates that the system is undergoing chaotic motion. Regions where the largest Liapunov exponent is negative correspond to periodic windows.

in this region (figure 6–12) indicates chaotic behaviour: regions where the largest Liapunov exponent is positive correspond to chaos, a negative λ_1 corresponding to a periodic window. The full structure cannot be seen in this 2-dimensional projection, at least 3 dimensions being needed to display the full structure and complexity of this attractor. The correlation dimension (D_2) was computed using 32,000 points assumed to be on the attractor, having run the system for 10,000 timesteps to discard transients. The computed dimension was $D_2 \approx 2.05$. Although a larger region of this graph was linear, this must be regarded as an unreliable figure. This chaotic region also features large periodic windows. When $r > 10.69$, the system executes period 10 motion; the chaotic region is not revisited as r increases. When $r = 11.19$ there is a shift to a coexisting attractor, executing a period 5 limit cycle (figure 6–13). This is a discontinuous change in the dynamics, corresponding to the initial conditions

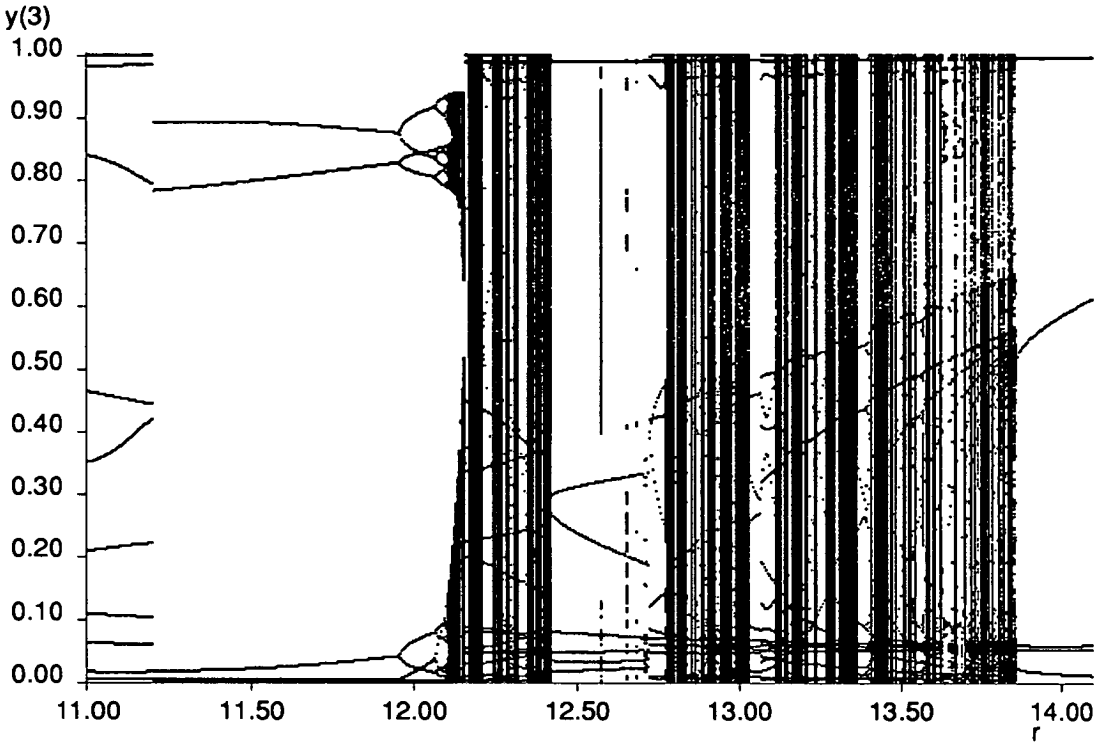
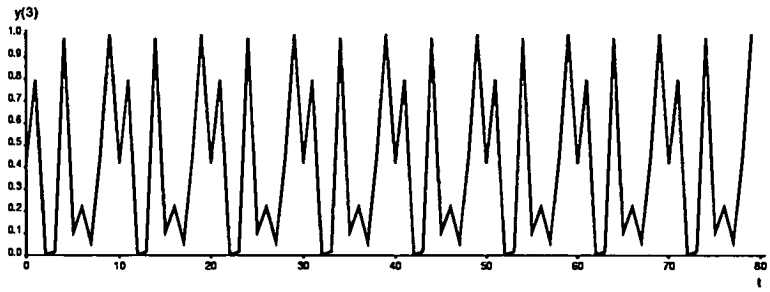


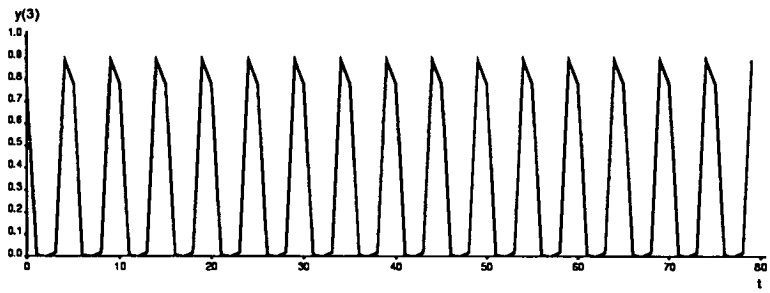
Figure 6–13: Bifurcation diagram for $11.0 \leq r < 14.1$ when $a = 0.0$. Period doubling to chaos occurs around $r = 12$.

moving from the basin of one attractor to another. The system becomes neutrally stable (ie $\lambda_1 = 0$) at this change. This is an initial conditions effect (figure 6–14): when a point on the period-5 attractor ($r = 11.20$) is used as an initial condition for the dynamics when $r = 11.19$, the same period-5 attractor results, rather than the previously seen period-10 attractor. The converse is also true.

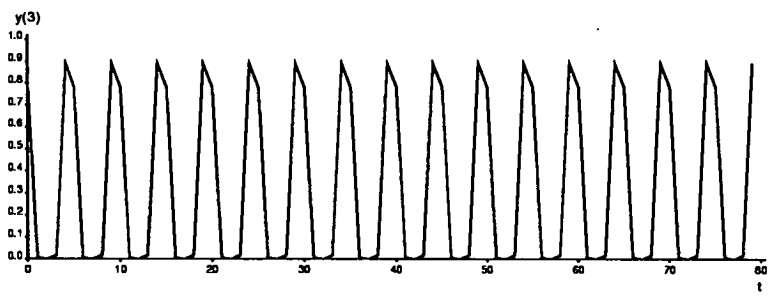
As r is further increased, the dynamics display period-doubling to chaos; this is may be observed in the bifurcation diagram (figure 6–15(a)) and in the Liapunov diagram (figure 6–15(b)). The period doubling may also be tracked using power spectra, with the corresponding new low frequency peak appearing at each period doubling (figure 6–16(a–e)). At the accumulation point of the period doubling the system bifurcates to chaotic behaviour and the power spectrum becomes broadband (figure 6–16(f)).



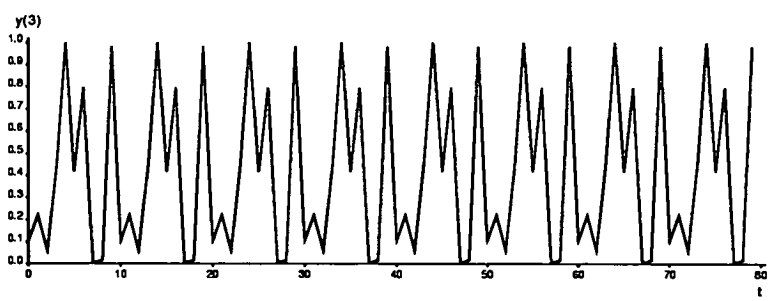
(a)



(b)

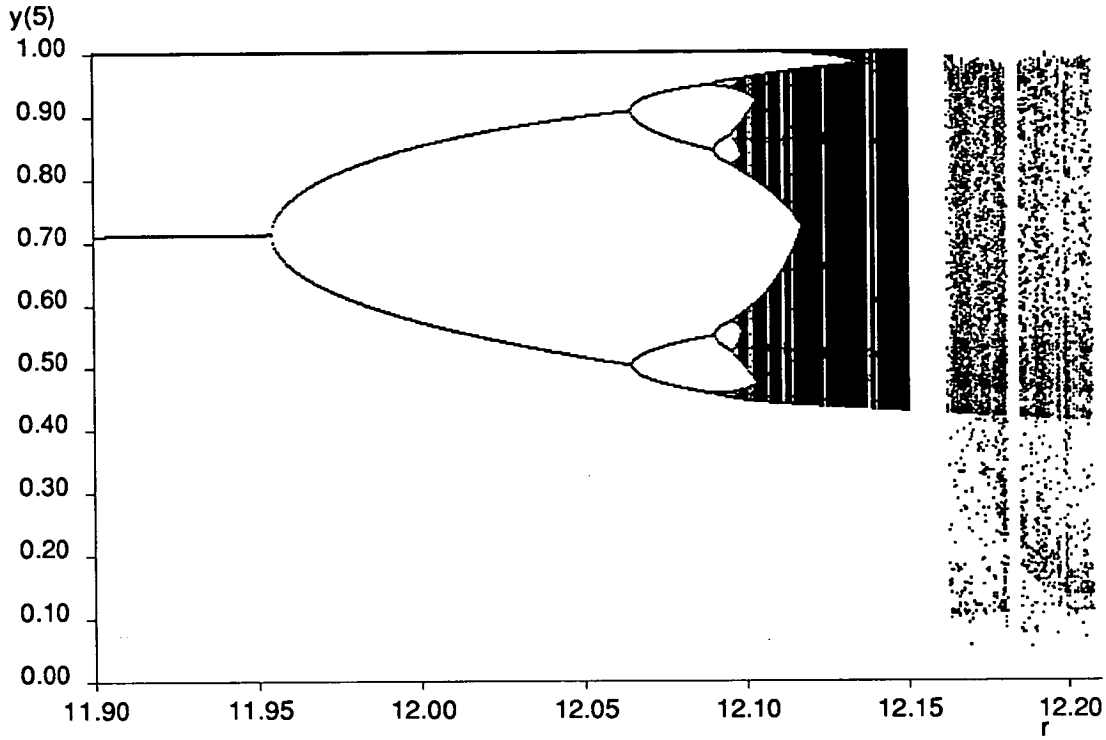


(c)

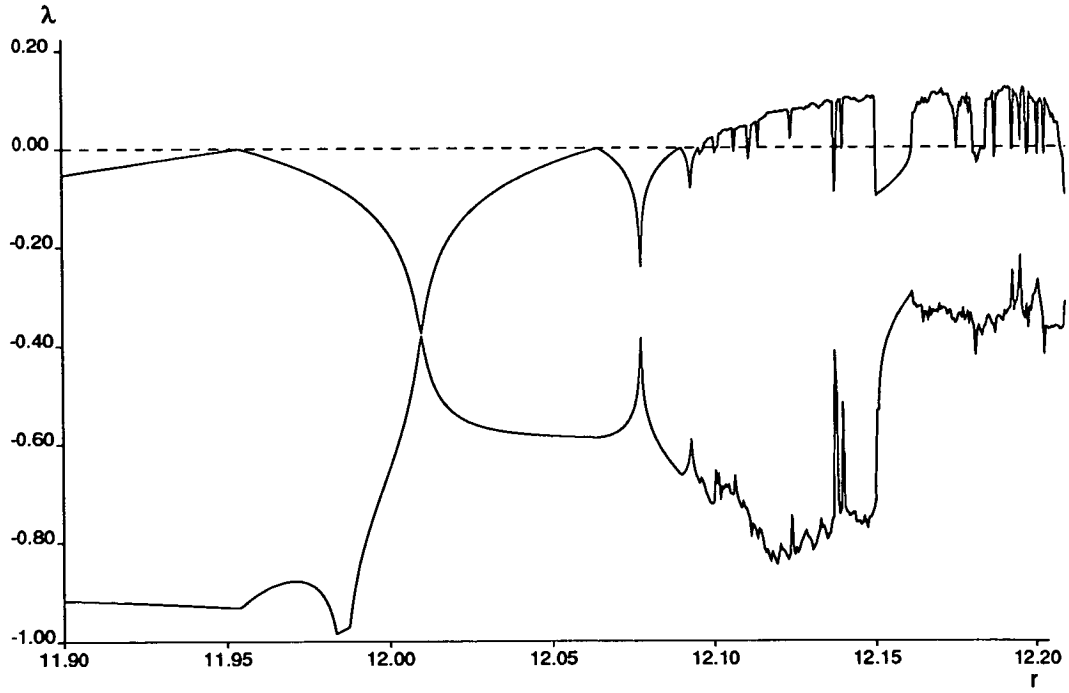


(d)

Figure 6–14: Time series at $r = 11.19$ (a, c) and $r = 11.20$ (b, d) at $a = 0.0$. These time series are taken at either side of a discontinuous change in the dynamics from period 10 motion to period 5 motion. (a) and (b) are time series taken using the same (random) initialisation as used elsewhere. (d) is initialised using a point on the attractor that is represented in (a) and (c) is initialised by a similar point from (b). (d) has similar period 10 dynamics to (a) and (c) has similar period 5 dynamics to (b). This indicates that phase change is caused by the movement of the attractor basin boundaries as r is increased. That is the original random starting point was initially in the basin of a period 10 attractor; however as r was increased, this point migrated into the basin of a period 5 attractor.



(a)



(b)

Figure 6–15: (a) Bifurcation diagram and (b) corresponding Liapunov diagram for the region of period doubling to chaos when $11.9 \leq r \leq 12.2$ and $a = 0.0$.

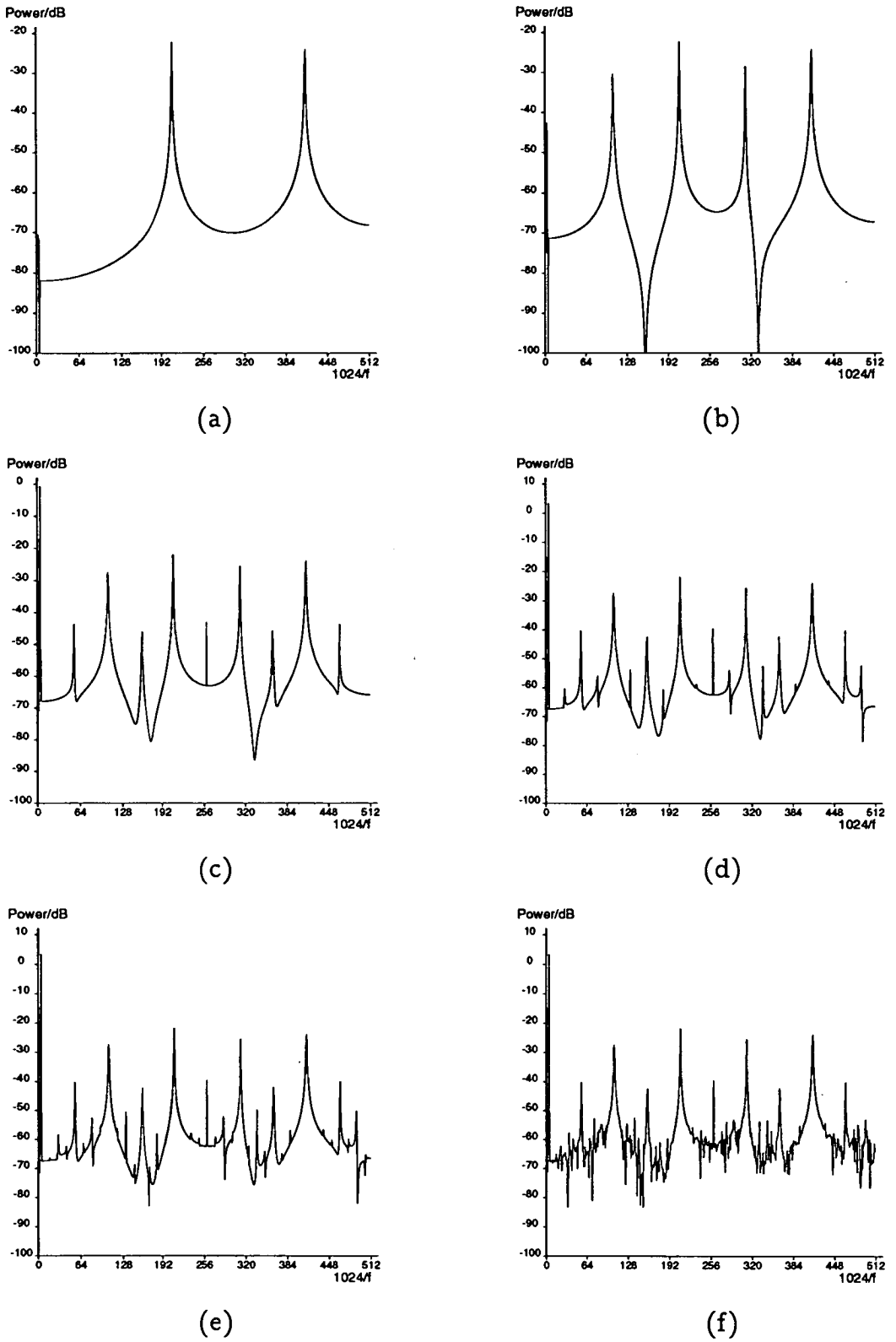


Figure 6–16: Power spectra taken with (a) $r = 11.9$, (b) $r = 12.0$, (c) $r = 12.075$, (d) $r = 12.093$, (e) $r = 12.096$ and (f) $r = 12.1$. The progression of period doublings may be clearly seen as there first period 5 motion (a) , then period 10 (b), then 20 (c), then 40 (d), then 80 (e). At $r = 12.1$ the system is chaotic; however a strong period 5 (or higher doubling) driving frequency is visible above the noise in the spectrum (f).

At the point of period doubling from period n to period $2n$, the fixed point of the map $f^n(x)$ becomes unstable. This is evidenced by the maximal Liapunov exponent becoming zero-valued as the system shifts from one limit cycle to another.

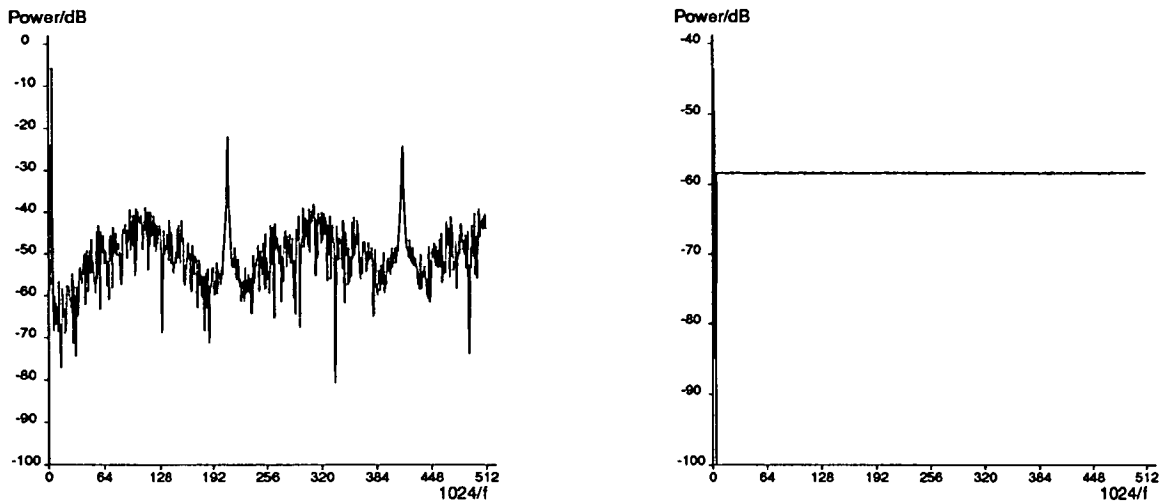
This classical route to chaos [55] is characterised by Feigenbaum's constant, δ:

$$r_n = r_\infty - k\delta^{-n} \qquad n >> 1, \tag{6.9}$$

where r_∞ is the accumulation point of the period doublings and r_n is the value of r at the nth period doubling. Within the floating point accuracy of the computer used to simulate this system, values of r were determined for the first 6 period doublings giving $\delta = 4.67 \pm 0.04$, which is in good agreement with the accepted value of $\delta = 4.6692\ldots$.

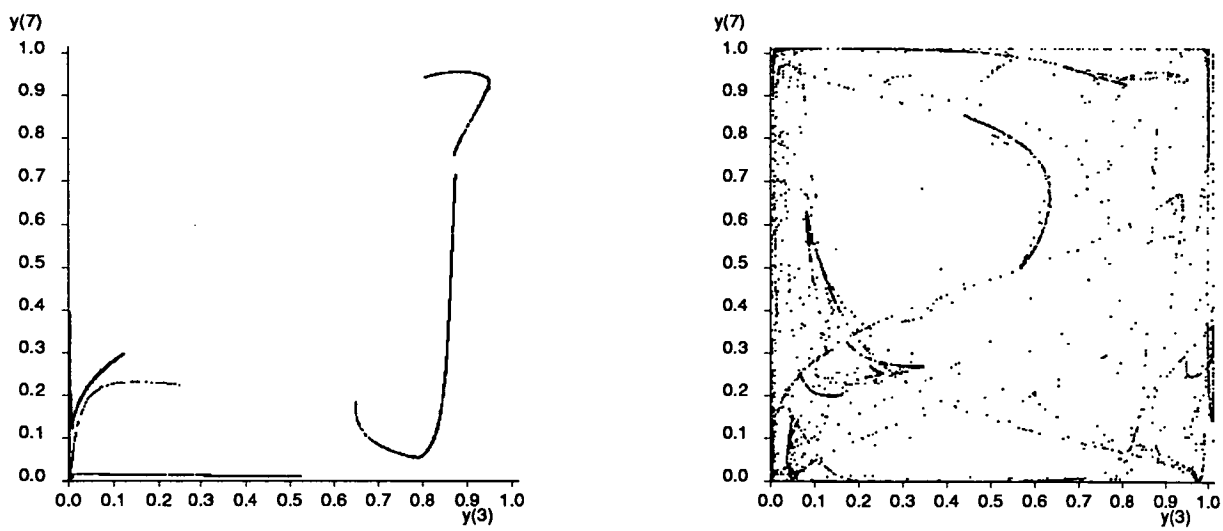
This route to chaos, and the value of the constant δ indicates that this system may be reduced to a 1-dimensional system with a quadratic maximum in this region of parameter space. Power spectra of the chaotic time series at $r = 12.15$ and $r = 12.18$ are shown in figure 6–17 and two-dimensional projections of the corresponding chaotic attractors are also plotted (figure 6–18).

The correlation dimension was again estimated from 32,000 points after discarding 10,000 as transients. The graphs of $C(l)$ vs. $\log(l)$ are shown in figures 6–19(a,b). The attractor sections, power spectra and bifurcation diagrams have all indicated that these are somewhat different attractors. The Grassberger-Procaccia computation gives $D_2 \approx 2.03$ when $r = 12.15$; when $r = 12.18$, $D_2 \approx 2.50$ — although there the linear portion of the graph is much smaller in this instance.



(a) (b)

Figure 6–17: Power spectra for chaotic motion at (a) $r = 12.15$ and (b) $r = 12.18$ ($a = 0.0$). As r is increased so the broad band chaos increases in power. The periodic frequency is still visible in (a); however a "phase change" occurs (visible in the bifurcation diagram after a wide periodic window), and power spectrum (b) is completely broad band.



(a) (b)

Figure 6–18: Two-dimensional projections of chaotic attractors at (a) $r = 12.15$ and (b) $r = 12.18$, when $a = 0.0$.

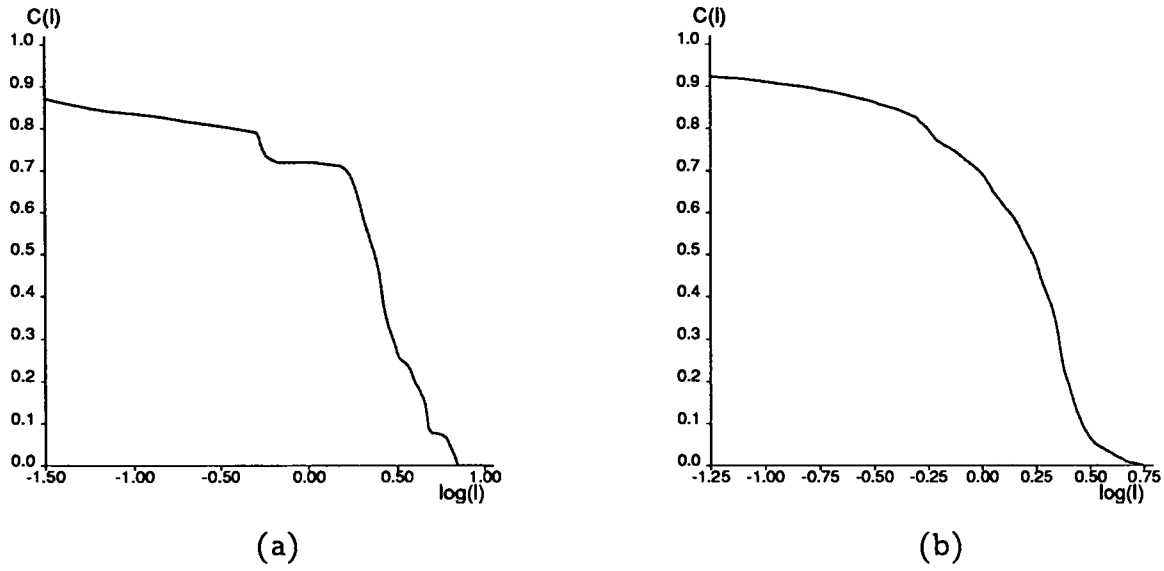


Figure 6–19: Graph of the correlation integral C(l) plotted against log(l) for (a) $\tau = 12.15$ and (b) $\tau = 12.18$ when $a = 0.0$. The slope of the linear region of this graph should equal the correlation dimension, D_2. (Note log(l) is scaled differently on the two graphs.)

6.3 Symmetry dependence

The dependence of the dynamics on the symmetry of the weight matrix was investigated by holding the gain constant ($\tau = 6.0$) and varying the symmetry parameter a. Fixed point behaviour resulted for a symmetric weight matrix ($a = 1.0$) and limit cycle period 4 for antisymmetric weight matrix ($a = -1.0$). The dynamics were more complex for values of a closer to zero (figure 6–20). Whilst fixed point behaviour dominates most of the symmetric region ($a > 0$), there are transitions between different fixed points. For the initial conditions used here, these transitions occurred at $a = 0.42$ and $a = 0.77$ (figure 6–21).

In both these transitions, the maximum Liapunov exponent vanished, indicating a neutrally stable state, at the boundary of 2 basins of attraction. This behaviour indicates that the dynamics of this network were considerably influenced by initial conditions, which were barely investigated here.

A Hopf bifurcation away from fixed point behaviour occurs at $a =$

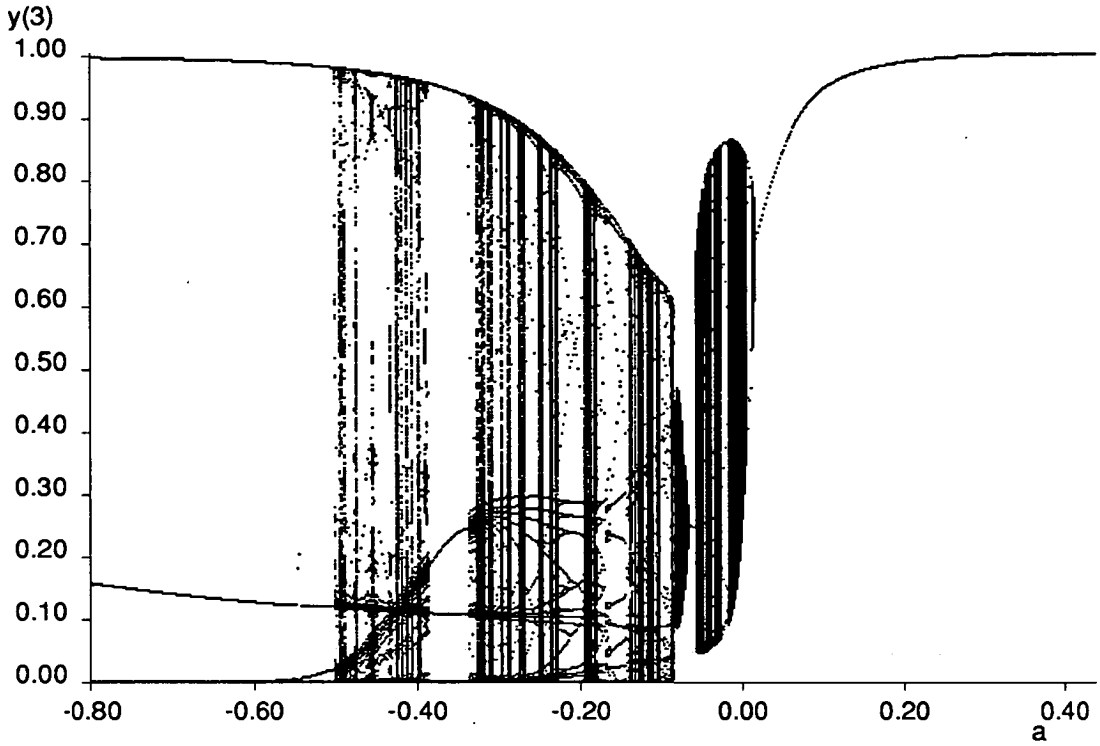
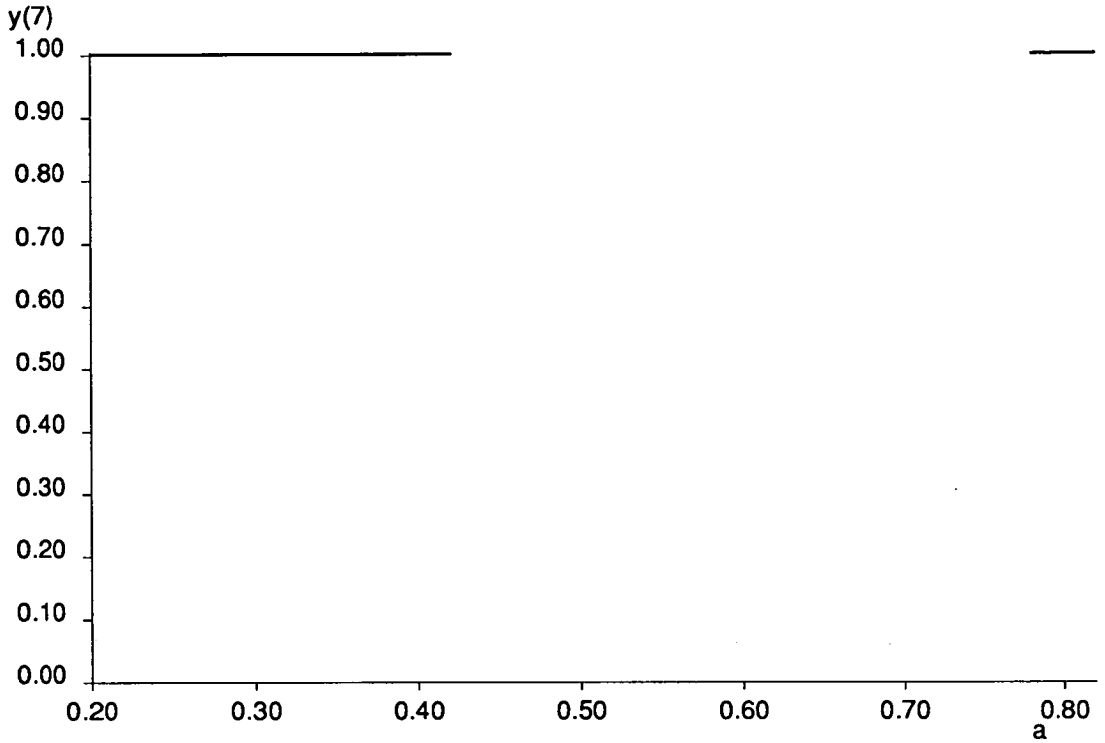
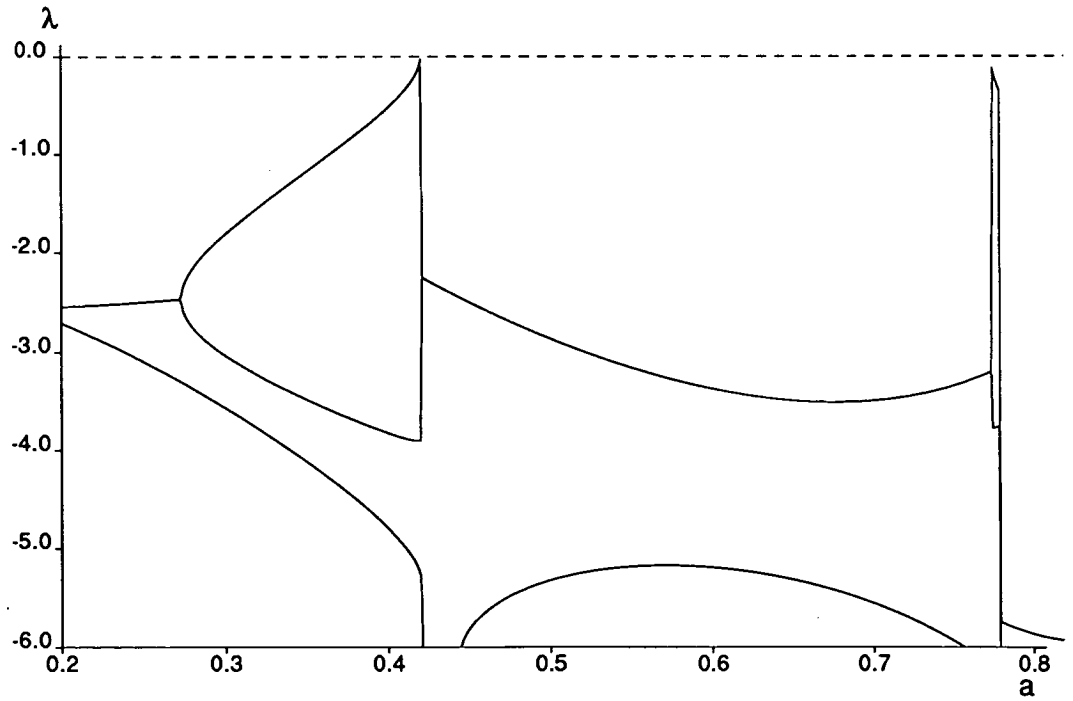


Figure 6–20: Bifurcation diagram for $-0.8 \leq a \leq 0.4$ and $r = 6.0$.

0.0153 (figure 6–22(a)) and this is almost immediately followed by a second Hopf bifurcation to two frequency behaviour. This can be seen by looking at power spectra taken in the region (figure 6–23). The system executes circle map dynamics and mode locking at rational winding numbers occurs (figure 6–24). The Hopf bifurcation and mode-locked regions are visible in the Liapunov diagram of this area (figure 6–22(b)). It is notable that W is not monotonic, displaying a maximum at $a = -0.0555$. At $a = -0.06$ there is an abrupt bifurcation back to fixed point behaviour; this transition is characterised by λ_2 vanishing. This change may be explained by the system moving from the basin of one attractor (a 2-torus) to another (a fixed point), with a neutrally stable state characterising the boundary. The importance of initial conditions is again emphasised here, but a fuller exposition of the dynamics of this system would require a much greater investigation of the coexisting attractors available to the system at any (a, r) setting.

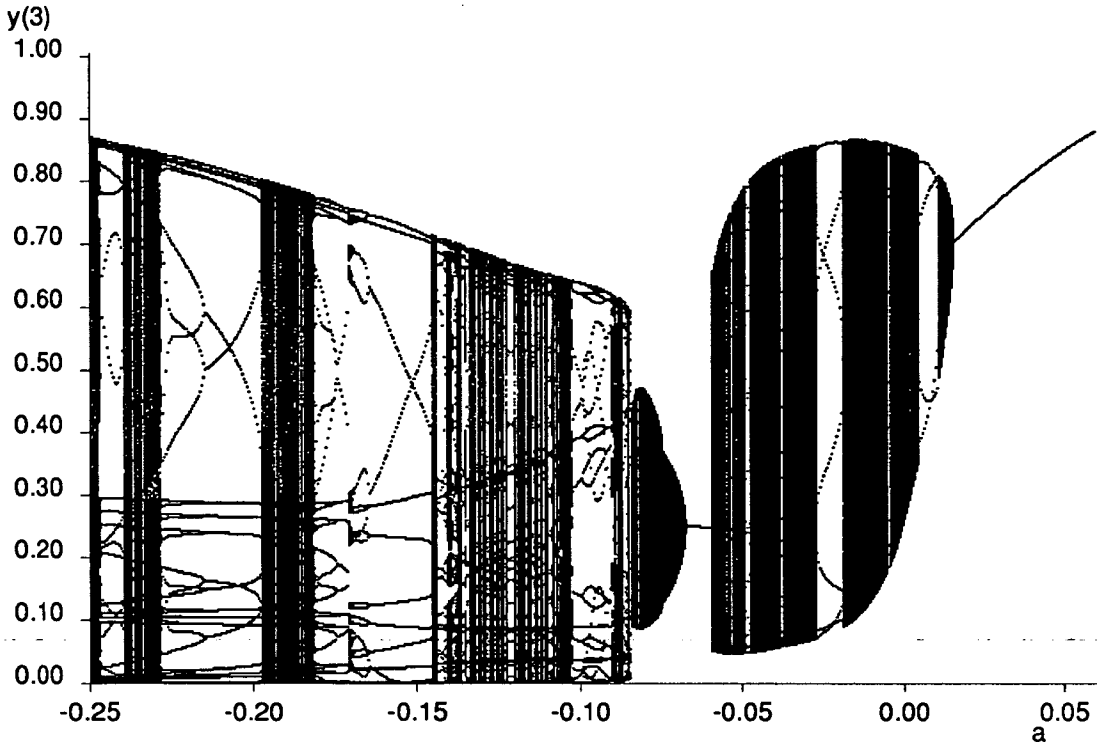


(a)

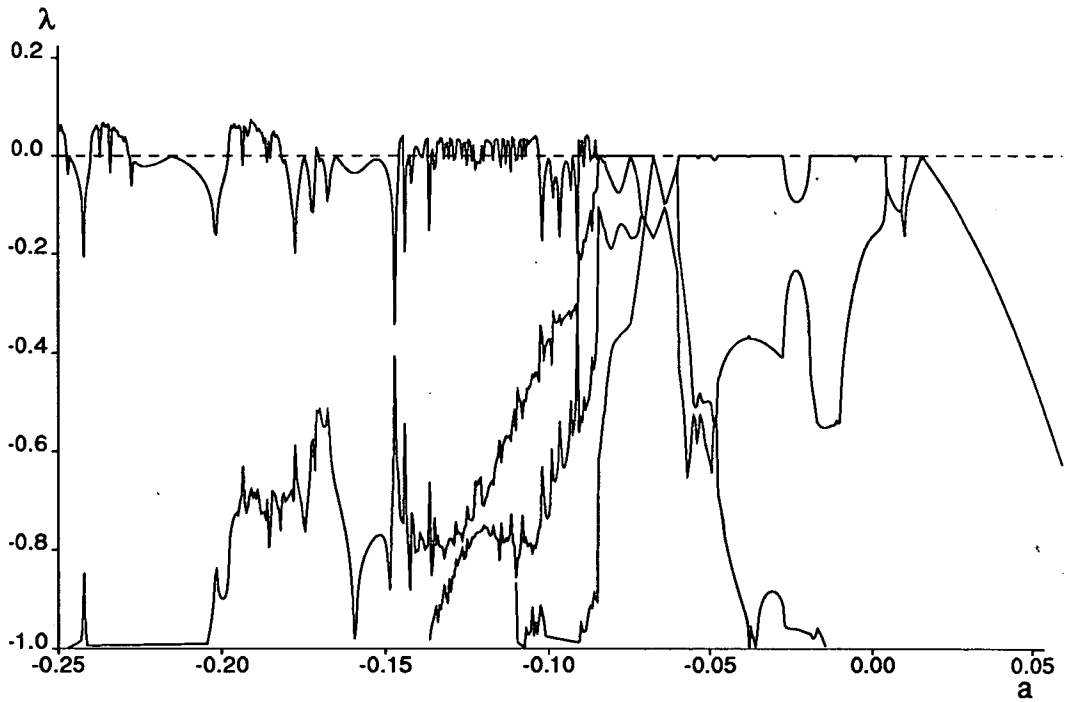


(b)

Figure 6–21: (a) Bifurcation diagram and (b) Liapunov diagram for $0.2 \leq a \leq 0.8$ when $r = 6.0$. This region is notable for transitions between coexisting fixed point attractors. The transitions are visible as their maximal Liapunov exponents vanish. These shifts between fixed points may be regarded as initial condition effects.



(a)



(b)

Figure 6–22: (a) Bifurcation diagram and (b) Liapunov diagram for $-0.25 \leq a < 0.05$, $r = 6.0$. A negative maximal Liapunov exponent signifies either fixed point or simple oscillatory behaviour.

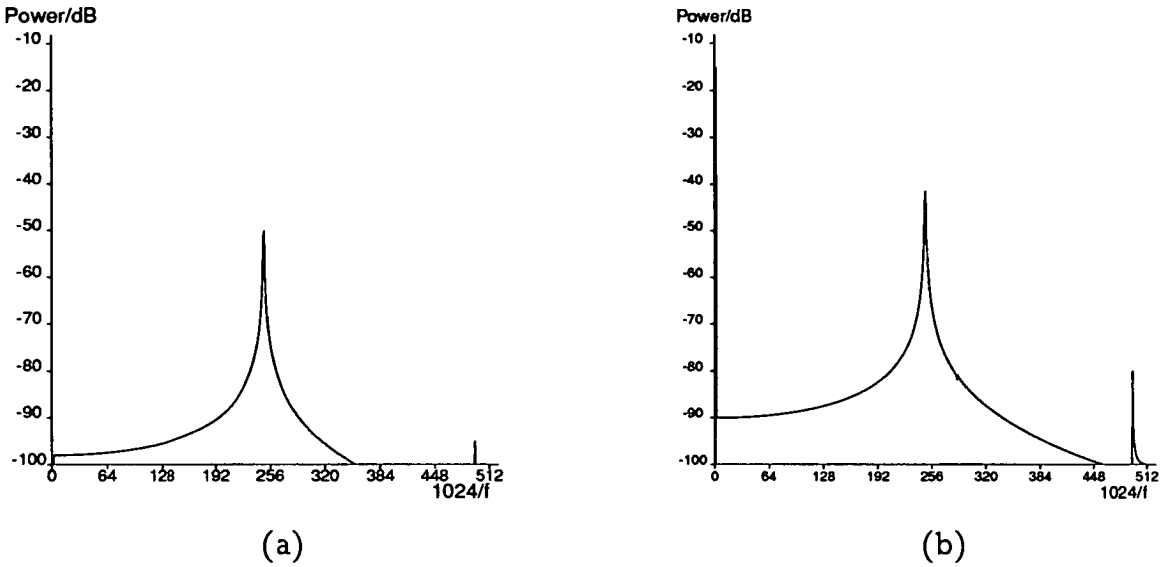


Figure 6–23: Power spectra taken at (a) $a = 0.0153$ and (b) $a = 0.01525$, when $r = 6.0$. In (b) a second frequency is visible, indicating a Hopf bifurcation from single frequency to two frequency behaviour.

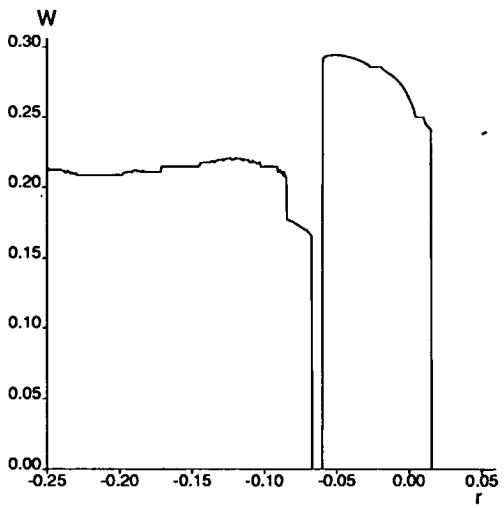


Figure 6–24: Graph of winding number W vs. a when $r = 6.0$. Note that the devil's staircase structure is not monotonically increasing and that it is interrupted by a region of fixed point behaviour — this is another example of behaviour governed by initial conditions.

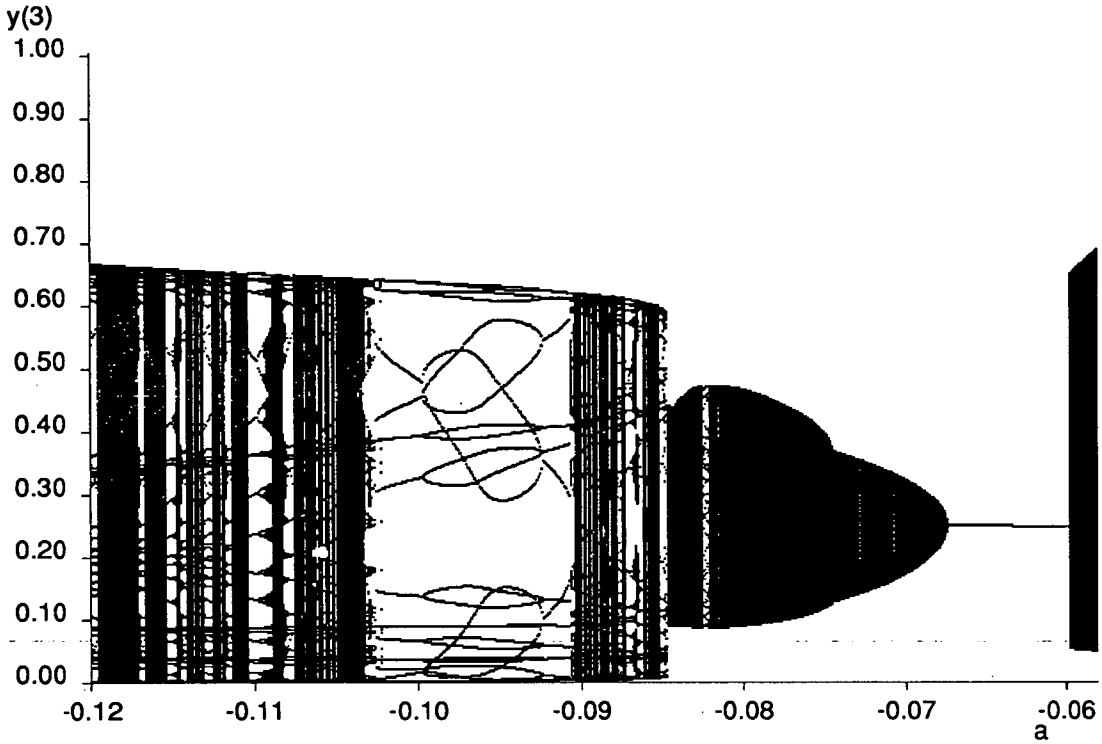


Figure 6–25: Magnified view of the bifurcation diagram $-0.12 \leq a < -0.06$ when $r = 6.0$ showing the abrupt transition from quasi-periodic to fixed point behaviour then back to fixed point behaviour. Chaotic motion ensues, via the Ruelle-Takens-Newhouse route, with the chaotic region showing periodic windows, period doublings and coexisting attractors.

The system resumed single frequency behaviour at $a = -0.0675$ and a second Hopf bifurcation to two frequency quasiperiodicity occurred at $a = -0.0685$. The system follows the Ruelle-Takens-Newhouse route to chaos, with the transition to chaos occurring at $a = -0.088$ (figure 6–25). This sequence may be followed by examining the Liapunov diagram (figure 6–22(b)).

The "circle map" behaviour exhibited here is again similar to that shown by systems such as (6.7). This system also uses two control parameters (K, Ω) which when plotted against each other give a phase diagram showing interlocked periodic and non-periodic behaviour. The areas of periodic behaviour appear as *Arnol'd tongues*. It seems that certain regions of (a, r) space display qualitatively similar dynamics to (6.7) and

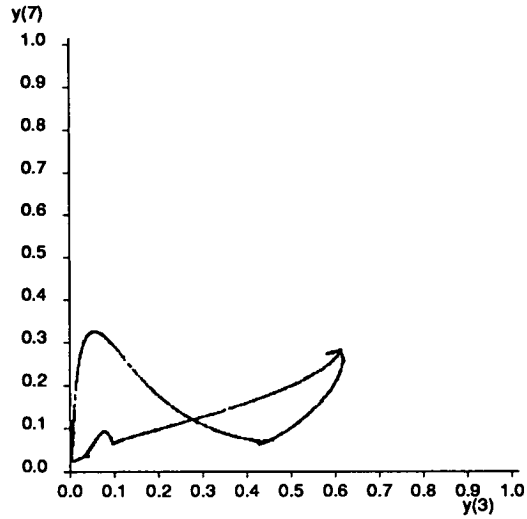


Figure 6–26: Section of chaotic attractor at $a = -0.088$, $r = 6.0$. Note the thin structure and cusps, corresponding to a low dimensionality ($D_2 \approx 1.54$).

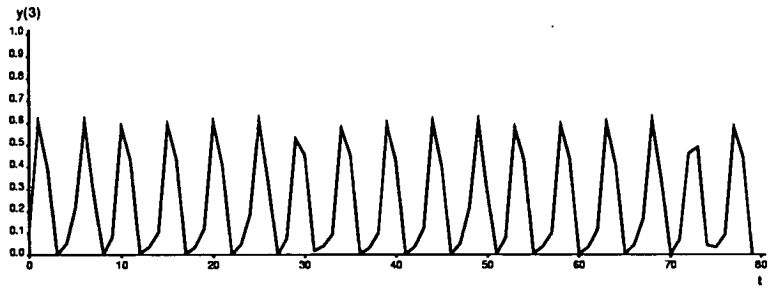


Figure 6–27: Time series of one node of the network at $a = -0.088$ and $r = 6.0$. Although this is a chaotic time series, there is an underlying periodicity.

that K and Ω may be regarded as nonlinear functions of (r, a).

A section of the resultant attractor (y_7 vs. y_3) is shown in figure 6–26. The correlation dimension of this attractor was computed, giving a value of $D_2 \approx 1.54$ at $a = -0.088$ (using 32000 points assumed to be on the attractor after 10000 iterations to run out transients).

This chaotic region continues until $a = -0.5$ and is somewhat complex, as evidenced by the bifurcation diagrams (figures 6–20 and 6–25). There are several large periodic windows and the chaotic regions (regions with positive λ_1) also show an underlying periodicity (figure 6–27). This

underlying periodicity is also visible in the bifurcation diagram.

6.4 Discussion

The results reported here are not surprising: a nonlinear system with a large number of degrees of freedom would be expected to display complex dynamical behaviour. However, an important facet of deterministic chaos is that high dimensional systems display behaviour that may be quantitatively described by a low dimensional equation of motion. For instance, the period doubling to chaos reported above is essentially one dimensional behaviour. This may be important if one wishes to engineer the dynamics of such networks, as problems of a seemingly intractably large dimension may in fact be reduced to a more managable lower dimension.

However, from an engineering perspective, dynamical neural networks will not be useful until powerful, general learning algorithms are developed to produce the desired attractors to perform a particular task: this is the temporal training problem. Several algorithms have been proposed for this task (e.g. [89, 94, 36, 14, 172, 210, 171, 148]), but there is (as yet) no algorithm to design a network to emulate a given dynamical process.

Baird [14] has offered an algorithm that allows the storage of N/2 periodic attractors, with specified Fourier components, in an N node network. This is a novel approach since it essentially inverts a set of analytical dynamical systems techniques to produce a method for constructing desired dynamical systems. Recently Baird has proposed a means of storing chaotic attractors in a neural network model (Baird, personal communication).

Shastri and Ajjanagadde [183] and Wang et al. [205] have both used phase relationships between stored periodic attractors in neural network models. Shastri and Ajjanagadde have devised a connectionist model to dynamically represent variable bindings via phase relationships. This system allows a natural means of performing inference using the dynamics

of the network. Wang et al. also employ the notion of encoding syntactical bindings as phase relationships between attractors, and demonstrate how a simple system may be devised to perform segmentation of patterns from a composite input. These models using phase relationships have an added biological plausibility following the recent results from the laboratories of Singer [69] and Eckhorn [47] that seem to suggest that phase-locking may be a means of establishing bindings between spatially separated areas of the visual field.

Neural networks displaying chaotic dynamics may be computationally useful. Rosenblatt [174] suggested a perceptron with sequential memory: this extension to perceptron theory utilised a set of feedback units that acted as a system clock. Rosenblatt's specification for these units was that they should display "non-repetitive and deterministic" dynamics — i.e. chaotic dynamics.

A strange attractor might be suitable for encoding a "don't know" or "waiting" situation [186, 144] in associative memory. In the absence of a recognised stimulus the system follows a strange attractor, allowing it to traverse state space in a non-repetitive fashion. Upon receipt of a known stimulus, the system may bifurcate to an attractor (typically a limit cycle) corresponding to a previously stored memory. A novel stimulus would cause the system to bifurcate to a new attractor corresponding to the new memory. A scheme for such a learning dynamics has been proposed by Lewenstein and Nowak [111], although they use random noise rather than deterministic chaos.

Gardner et al. [62] analytically calculated the storage capacity (where a stored pattern corresponded to a fixed point attractor) of a fully connected recurrent network (where each binary node possessed a step transfer function) with respect to the symmetry (a) and a stability parameter (κ) that indicated the size of the basin of attraction around fixed points. The theoretical curve they derived indicated that the region of maximal storage corresponded to weight matrices with larger symmetric components ($a = 0.3$ when $\kappa = 0$, $a \to 1.0$ as $\kappa \to \infty$). If these regions are

related to the results presented in the previous chapter, then the regions of maximal storage correspond to regions of simple dynamics (i.e. high power and low spectral entropy). This is not surprising, as regions with a high density of stable fixed points are unlikely to occur in regions with a high density of complex periodic or strange attractors.

The mode-locking behaviour indicates that the network may be regarded in terms of coupled oscillators. This may come about through (nonlinear) combinations of subnets, with frequencies determined by (r, a). A possible use for this phenomenon might be as a form of local system timing which could be useful for operations such as speech production and motor control. More trivially, mode-locking also has implications for fault-tolerance and learning: when a system moves into the appropriate region of parameter space, it will lock into a particular frequency ratio. Additionally mode-locked neural networks, could be used in the implementation of phase-locking schemes (discussed above) such as the one proposed by Shastri [183].

Attractors displaying intermittently chaotic behaviour might also be useful in associative memory. The characteristic motion of intermittency consists of long, regular periods together with bursts of irregular motion. In the context of a time-dependent associative memory, this might provide a mechanism for leaving a memory sometime after recall, needing no external stimulus. In effect, this could be regarded as a self-priming or reset operation arising naturally from the dynamics.

Chaotic attractors act as information generators in some directions (corresponding to positive Liapunov exponents) and information compressors in others (corresponding to negative Liapunov exponents). By partioning the phase space into labelled boxes, dynamical systems may be used to generate strings of symbols. Nicolis et al [136] show how the Rössler attractor may produce a symbol sequence that is closely approximated by a fifth-order Markov process. Such considerations may be important when applying dynamical systems to high level cognitive and linguistic problems. Crutchfield and Young [41] have presented a po-

tentially important technique for reconstructing minimal computational models (via a variable order Markov chain) from the structure of symbolic dynamic sequences.

Long-lived transients are often observed in the dynamical system studied here; for example in the initial Hopf bifurcation away from fixed point behaviour, the maximum Liapunov exponent tends toward zero and transient motion may last many thousands of timesteps. Kantz and Grassberger [90] and Crutchfield [40] have both argued that transient motion may be more significant than the underlying attractor. Crutchfield studied a simple one-dimensional lattice dynamical system. He demonstrated (by curve-fitting to experimental data) that transient length was hyperexponential[3] with respect to lattice size. This relationship suggests that an experimentalist could not expect to see attractor motion on lattices of more than $\mathcal{O}(10^2)$ sites, owing to extremely long transient lengths. Kantz and Grassberger have suggested that chaotic transients are more robust against noise than the true attractors, and may be regarded as being "more typical" of the dynamical system. Additionally, both Crutchfield and Kantz and Grassberger also demonstrate that chaotic transients may have convergent statistics and give no sign of convergence to an attractor. In particular, Crutchfield defines type I transients which do not have convergent statistics and show clear time-dependence and type II transients, which have convergent statistics and may be indistinguishable from an attractor.

These observations suggest that it might be more profitable to study the transients of neural network dynamical systems, rather then the underlying attractors: Crutchfield's work suggests that a large, spatially extended dynamical system (such as a neural network) is likely to have extremely long transients, growing exponentially (or hyperexponentially) with network size. However, whilst dynamical systems theory has reached a good understanding of attractor dynamics, the study of transients is

3 i.e. Of the form $\exp(x^{\alpha})$.

still ill-developed. Indeed, most theoretical papers ignore the issue of transients entirely.

Transient motion is likely to be important for modelling various physiological and perceptual signals. Consider speech production. Here, the articulatory system may be regarded as moving from one target to another, corresponding to the production of basic sounds (phones). Whilst long vowel sounds (for instance) may be regarded as attractor motion, in many instances the articulatory system does not reach its target state, before moving to a new target corresponding to the next phone. In this case, speech production might be regarded as a concatenation of transients, rather than as an attractor[4]. Hence, a better way to model speech production might be by engineering the transients of a dynamical system, rather then the attractors[5].

The observation that chaotic transients are more robust than the underlying attractors [90] may be important for schemes in which the learning dynamics are concurrent with the network dynamics. (Here, we assume that the attractors are being engineered.) A systems could exist on a robust, long-lived transient, according to the network dynamics, while the learning dynamics makes changes to the parameters of the dynamical system (and hence changes the underlying attractor). Exploitation of these properties would allow the system to explore parameter space to find a suitable attractor, whilst existing on a robust transient. A scheme similar to this has been proposed by Ott, Grebogi and Yorke [142] who use the observation that a chaotic attractor may be converted to any one of a large number of stable periodic motions, by making only small parameter perturbations.

4 Of course, the sequence of phones may be produced by some sort of attractor dynamics.

5 Engineering the transients is essentially engineering the attractor basins. However, the emphasis is somewhat different.

7

CONCLUSIONS

This thesis has made two principal contributions. Firstly, the radial basis functions (RBF) network was introduced and its properties related to other classifiers. This network architecture was then used in various speech recognition problems, culminating in a continuous speech recognition task with a 571 word lexicon. Secondly, a study was made of the dynamics of a simple recurrent network architecture. This study was conducted via a numerical survey of network power spectra and through a detailed investigation of the dynamics exhibited by a particular network. Complex dynamical behaviour, including chaos, was observed.

Word and sentence recognition errors were reported for a continuous speech recognition task using phoneme modelling based on an RBF network and a Viterbi smoothing algorithm, and the CSTR language model. In the CSTR cytopathology task domain the best RBF/Viterbi system returned first choice word errors of 6% and sentence errors of 14% when using a restricted grammar with a perplexity of 6. This compares well with the best results returned using the CSTR HMM system for phoneme modelling (word errors of 4%, sentence errors of 8%). In a system using no grammar (perplexity 571) the best RBF system (in a Gaussian classifier configuration) returned a word error rate of 42% (the best CSTR HMM system returned a word error of 53%).

Previous to the continuous speech experiments, the RBF networks were used for a static speech pattern classification task, vowel labelling. The hand-segmented vowels were excised from continuous speech; 20 vowel

classes were defined. The best RBF network resulted in a 26.9% classification error. Various MLP configurations, a pooled covariance Gaussian classifier and a discrete HMM were also tested on this task: all produced poorer error rates than the best RBF network. Statistics (taken over initial conditions) were also computed for the MLPs and RBF networks, in an attempt to see how robust these classification methods were as a function of the number of hidden units.

Great emphasis was placed on relating the RBF neural network model to other pattern recognition methodologies, both statistical and neural network. The estimation of probability density functions using Parzen windows and the determination of discrimination functions using the method of potential functions were both shown to be very similar to the method of RBFs. Additionally it was demonstrated how Bayesian classifiers using Gaussians or mixtures of Gaussians could be implemented as an RBF network. This manner of implementation of these classifiers highlighted the usual maximum likelihood training procedure and enabled the introduction of some discriminative training methods. RBF networks were also related to vector quantisation; a layer of RBF functions was regarded as a VQ codebook with overlapping continuous functions, rather than disjoint codewords. Using this relation, an isomorphism between semi-continuous hidden Markov models and Viterbi trained RBF networks was identified. The close relations between RBF networks and various other neural network models were discussed. Many proposed feed-forward networks featuring input dimensionality expansion were shown to be special cases of the the RBF network model.

The second part of this thesis was a study of the dynamics of a simple recurrent network model. This study was motivated by a desire to model the dynamics of the speech signal and a realisation that feed-forward networks (or hybrids of these with HMMs) might not be sufficient to model speech dynamics. Rather than attempt to develop another temporal training algorithm for recurrent networks, or apply currently proposed ones (which either lack power or are computationally infeasible for con-

tinuous speech recognition) it was decided to attempt to reach a greater understanding of the realm of dynamics accessible to this model.

The approach to this problem was computational. To reduce the number of free parameters in a set of networks, two control parameters representing the gain of the transfer function and the symmetry of the weight matrix were defined. Two additional parameters were network size and the timestep used in discretisation of the network's continuous equation of motion. A large number of networks were simulated and their dynamical properties were collected by computing summary statistics of network power spectra. This gave an indication of the temporal activity of the network with respect to parameter space. The results of this survey indicated that although the amount of temporal activity increased with network size, the regions of maximal temporal activity remained approximately invariant. This area corresponded to random or slightly antisymmetric weight matrices and high gain (although if the gain was increased too much, high frequency oscillations would result). Reducing the time step (Δt) used in discretisation resulted in less temporal activity, as was expected.

The detailed behaviour of a specific 8 node network was also studied, using bifurcation diagrams, the computation of Liapunov exponents, power spectra, attractor sections and the calculation of fractal dimensions. Complex dynamical behaviour was observed with Hopf bifurcations, the Ruelle-Takens-Newhouse route to chaos with mode-locking at rational winding numbers, the period-doubling route to chaos and the presence of multiple coexisting attractors. The existence of deterministic chaotic motion in a high dimensional neural network indicates that seemingly high dimensional motion may in fact be essentially low dimensional. This may be important for training large networks.

The underlying theme of this thesis has been constructing models of speech dynamics, however crude, in an attempt to perform speech recognition. Firstly a feed-forward network and a first order Markov process was used for a continuous speech recognition task. Surprisingly good re-

sults were obtained using this system. It was hypothesised that networks with feedback capable of executing complex dynamical motion might be able to model speech dynamics more faithfully. An investigation into the dynamics of these networks indicated which regions of parameter space favoured more complex dynamics and characterised the dynamics displayed by a small recurrent network. Future work may tie these areas together more strongly by researching algorithms to train recurrent networks to model speech dynamics.

BIBLIOGRAPHY

[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[3] A. Aktas, O. Schmidbauer, K.-H. Maier, and W. H. Feix. Classification of coarse phonetic categories in continuous speech: statistical classifiers vs. temporal flow connectionist network. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 89–92, Albuquerque, 1990.

[4] I. Aleksander and T. J. Stonham. Guide to pattern recognition using random-access memories. *Computers and digital techniques*, 2:29–40, 1979.

[5] Robert B. Allen. Several studies on natural language and back-propagation. In *Proceedings First IEEE International Conference on Neural Networks*, volume II, pages 335–341, San Diego, 1987.

[6] Svën Anderson, John Merrill, and Robert Port. Dynamic speech categorization with recurrent networks. Technical Report 258, Departments of Linguistics and Computer Science, Indiana University, 1988.

[7] J. D. Aplevich. Models of certain nonlinear systems. In E. R. Caianiello, editor, *Neural Networks*, pages 110–115. Springer-Verlag, Berlin, 1968.

[8] Y. Ariki, F. R. McInnes, and M. A. Jack. Hierarchical phoneme discrimination by hidden Markov model using cepstrum and formant information. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 663–666, Glasgow, 1989.

[9] K. L. Babcock and R. M. Westervelt. Dynamics of simple electronic neural networks. *Physica*, 28D:305–316, 1987.

[10] A. Babloyantz and A. Destexhe. Low-dimensional chaos in an instance of epilepsy. *Proceedings of the National Academy of Sciences USA*, 83:3513–3517, 1986.

[11] L. R. Bahl, R. Bakis, J. Bellegarda, P. F. Brown, D. Burshtein, S. K. Das, P. V. de Souza, P. S. Gopalakrishnan, F. Jelinek, D. Kanevsky, R. L. Mercer, A. J. Nadas, D. Nahamoo, and M. A. Picheny. Large vocabulary natural language continuous speech recognition. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 465–467, Glasgow, 1989.

[12] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 49–52, Tokyo, 1986.

[13] Bill Baird. Nonlinear dynamics of pattern formation and pattern recognition in the rabbit olfactory bulb. *Physica*, 22D:150–175, 1986.

[14] Bill Baird. Bifurcation theory methods for programming static or periodic attractors and their bifurcations in dynamic neural networks. In *Proceedings Second IEEE International Conference on Neural Networks*, volume I, pages 9–16, San Diego, 1988.

[15] M. Bauer and W. Martienssen. Quasi-periodic route to chaos in neural networks. *Europhysics Letters*, 10:427–431, 1989.

[16] Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[17] Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; A method for computing all of them. Part 1: Theory. *Meccanica*, 15:9–20, 1979.

[18] Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; A method for computing all of them. Part 2: Numerical application. *Meccanica*, 15:21–30, 1979.

[19] M. Blomberg, R. Carlson, K. Elenius, and B. Granstrom. Auditory models as front ends in speech recognition systems. In J. S. Perkell and D. H. Klatt, editors, *Invariance and Variability in Speech Processes*, pages 108–122. Lawrence Erlbaum Associates, Hillsdale NJ, 1986.

[20] H. Bourlard, Y. Kamp, H. Ney, and C. J. Wellekens. Speaker-dependent connected speech recognition via dynamic programming and statistical methods. In M. R. Schroeder, editor, *Speech and Speaker Recognition*, volume 12 of *Bibliotheca Phonetica*, pages 115–148. Karger, Basel, 1985.

[21] H. Bourlard and N. Morgan. A continuous speech recognition system embedding MLP into HMM. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 413–416. Morgan Kaufmann, San Mateo CA, 1990.

[22] H. Bourlard and C. J. Wellekens. Links between Markov models and multilayer perceptrons. Technical Report M263, Philips Research Laboratory, Brussels, 1988.

[23] H. Bourlard and C. J. Wellekens. Speech dynamics and recurrent neural networks. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 33–36, Glasgow, 1989.

[24] H. Bourlard and C. J. Wellekens. Speech pattern discrimination and multilayer perceptrons. *Computer Speech and Language*, 3:1–19, 1989.

[25] John S. Bridle. Alpha-nets: a recurrent neural network architecture with a hidden Markov model interpretation. *Speech Communication*, 9:83–92, 1990.

[26] John S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 211–217. Morgan Kaufmann, San Mateo CA, 1990.

[27] D. S. Broomhead and David Lowe. Multi-variable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[28] Peter F. Brown. *The Acoustic-Modelling Problem in Automatic Speech Recognition*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1987.

[29] A. E. Bryson Jr. and Yu-Chi Ho. *Applied Optimal Control*. Blaisdell Publishing Co., 1969.

[30] Martin Casdagli. Nonlinear prediction of chaotic time series. *Physica*, 35D:335–356, 1989.

[31] S. Chen, C. F. N. Cowan, P. M. Grant, and S. A. Billings. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, to appear.

[32] M. Y. Choi and B. A. Huberman. Digital dynamics and the simulation of magnetic systems. *Physical Review B*, 28:2547–2554, 1983.

[33] M. Y. Choi and B. A. Huberman. Dynamic behaviour of nonlinear networks. *Physical Review A*, 28:1204–1206, 1983.

[34] Y. Chow, M. Dunham, O. Kimball, M. Krasner, G. F. Kubala, J. Makhoul, P. Price, S. Roucos, and R. Schwartz. BYBLOS: the BBN continuous speech recognition system. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 89–92, Tokyo, 1987.

[35] Michael A. Cohen and Stephen Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 13:815–826, 1983.

[36] A. C. C. Coolen, J. J. Denier van der Gon, and T. W. Ruijgrok. An exact dynamical solution for generalized Hopfield networks. In L. Personnaz and G. Dreyfus, editors, *Neural Networks from Models to Applications*, pages 269–276. I. D. S. E. T., Paris, 1989.

[37] Piero Cosi, Yoshua Bengio, and Renato de Mori. Phonetically-based multi-layered neural networks for vowel classification. *Speech Communication*, 9:15–29, 1990.

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

[39] A. Crowe and M. A. Jack. A globally optimising formant tracker using generalised centroids. *Electronics Letters*, 23:1019–1020, 1987.

[40] James P. Crutchfield. Subbasins, portals and mazes: transients in high dimensions. *Nuclear Physics B (Proc. Suppl.)*, 5A:287–292, 1988.

[41] James P. Crutchfield and Karl Young. Inferring statistical complexity. *Physical Review Letters*, 63:105–108, 1989.

[42] J. Dalby, A. Crowe, and A. M. Sutherland. Formant-based vowel classification in continuous speech. In *Proceedings European Conference on Speech Communication and Technology*, volume 1, pages 244–247, 1989.

[43] J. Demmel. The geometry of ill-conditioning. *Journal of Complexity*, 3:201–229, 1987.

[44] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39:1–38, 1977.

[45] J. Duchon. Spline minimizing rotation-invariant semi-norms in Sobolev spaces. In W. Schempp and K. Zeller, editors, *Constructive Theory of Functions of Several Variables*, volume 571 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1977.

[46] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley Interscience, New York, 1973.

[47] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, and H. J. Reitboeck. Coherent oscillations: a mechanism of feature linking in the visual cortex? *Biological Cybernetics*, 60:121–129, 1988.

[48] J.-P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Reviews of Modern Physics*, 57:617–656, 1985.

[49] Jeffrey L. Elman. Finding structure in time. Technical Report CRL-8801, Center for Research in Language, University of California, San Diego, 1988.

[50] Jeffrey L. Elman and David Zipser. Learning the hidden structure of speech. *Journal of the Acoustical Society of America*, 83:1615–1626, 1988.

[51] F. Fallside, H. Lucke, T. P. Marsland, T. P. O'Shea, M. St. J. Owen, R. W. Prager, A. J. Robinson, and N. H. Russell. Continuous speech recognition for the TIMIT database using neural networks. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 445–448, Albuquerque, 1990.

[52] B. G. Farley and W. A. Clark. Activity in networks of neuron-like elements. In C. Cherry, editor, *Information Theory*, pages 242–251. Butterworths, Washington, 1961.

[53] J. D. Farmer. Information dimension and the probabilistic structure of chaos. *Z. Naturforsch.*, 37a:1304, 1982.

[54] J. Doyne Farmer and John J. Sidorowich. Exploiting chaos to predict the future and reduce noise. In Y. C. Lee, editor, *Evolution, Learning and Cognition.* World Scientific, Singapore, 1988.

[55] Mitchell J. Feigenbaum. Universal behaviour in nonlinear systems. *Los Alamos Science*, 1:4–27, 1980.

[56] Michael A. Franzini, Kai-Fu Lee, and Alex Waibel. Connectionist Viterbi training: a new hybrid method for continuous speech recognition. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 425–428, Albuquerque, 1990.

[57] Michael A. Franzini, Michael J. Witbrock, and Kai-Fu Lee. A connectionist approach to continuous speech recognition. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 425–428, Glasgow, 1989.

[58] W. J. Freeman. *Mass Action in the Nervous System.* Academic Press, New York, 1975.

[59] A. Frumkin and E. Moses. The physicality of the Little model. *Physical Review A*, 34:714–716, 1986.

[60] T. Fukai and M. Shiino. Asymmetric neural networks incorporating the Dale hypothesis and noise driven chaos. *Physical Review Letters*, 64:1465–1468, 1990.

[61] Stephen I. Gallant and Donald Smith. Random cells: an idea whose time has come and gone... and come again? In *Proceedings First IEEE International Conference on Neural Networks*, volume II, pages 671–678, 1987.

[62] E. Gardner, H. Gutfreund, and I. Yekutieli. The phase space of interactions in neural networks with definite symmetry. *Journal of Physics A: Math. Gen.*, 22:1995–2008, 1989.

[63] Michael Gherrity. A learning algorithm for analog, fully recurrent neural networks. In *Proceedings International Joint Conference on Neural Networks*, volume I, pages 643–644, Washington, DC, 1989.

[64] Oded Ghitza. Auditory neural feedback as a basis for speech processing. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 91–94, New York, 1988.

[65] Bernard Gold and Richard Lippmann. Neural net classifiers useful for speech recognition. In *Proceedings First IEEE International Conference on Neural Networks*, volume IV, pages 417–426, San Diego, 1987.

[66] J. P. Gollub and Harry L. Swinney. Onset of turbulence in a rotating fluid. *Physical Review Letters*, 35:927–930, 1975.

[67] Marco Gori, Yoshua Bengio, and Renato de Mori. BPS: a learning algorithm for capturing the dynamic nature of speech. In *Proceedings International Joint Conference on Neural Networks*, volume II, pages 417–423, Washington, DC, 1989.

[68] Peter Grassberger and Itmar Procaccia. Measuring the strangeness of strange attractors. *Physica*, 9D:189–208, 1983.

[69] Charles M. Gray, Peter König, Andreas K. Engel, and Wolf Singer. Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties. *Nature*, 338:334–337, 1989.

[70] Augustine H. Gray, Jr. and John D. Markel. Distance measures for speech processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24:380–391, 1976.

[71] Michael R. Guevara, Leon Glass, Michael C. Mackey, and Alvin Shrier. Chaos in neurobiology. *IEEE Transactions on Systems, Man and Cybernetics*, 13:790–798, 1983.

[72] H. Gutfreund, J. D. Reger, and A. P. Young. The nature of attractors in an asymmetric spin glass with deterministic dynamics. *Journal of Physics A: Math. Gen.*, 21:2775–2797, 1988.

[73] Thomas C. Halsey, Mogens H. Jensen, Leo P. Kadanoff, Itamar Procaccia, and Boris I. Shraiman. Fractal measures and their singularities: the characterisation of strange sets. *Physical Review A*, 33:1141–1151, 1986.

[74] John B. Hampshire II and Alex Waibel. Connectionist architectures for multi-speaker phoneme recognition. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 203–210. Morgan Kaufmann, San Mateo CA, 1990.

[75] D. J. Hand. *Kernel Discriminant Analysis.* Research Studies Press, 1982.

[76] R. L. Hardy. Multiquadratic interpolation at arbitary points made simple. *Journal of Geophysics Research*, 76:1905–1915, 1971.

[77] J. Harrington, J. Laver, and D. Cutting. Word-structure reduction rules in automatic continuous speech recognition. *Proceedings of the Institute of Acoustics*, 8:451–459, 1986.

[78] G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

[79] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 79:2554–2558, 1982.

[80] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences USA*, 81:3088–3092, 1984.

[81] William Huang, Richard Lippmann, and Ben Gold. A neural net approach to speech recognition. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 99–102, New York, 1988.

[82] X. D. Huang and M. A. Jack. Semi-continuous hidden Markov models for speech signals. *Computer Speech and Language*, 3:239–251, 1989.

[83] A. W. F. Huggins and R. S. Nickerson. Speech quality evaluation using 'phoneme-specific' sentences. *Journal of the Acoustical Society of America*, 77:1896–1906, 1985.

[84] Jeng-Neng Hwang, John A. Vlontzos, and Sun-Yuan Kung. A systolic neural network architecture for hidden Markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:1967–1979, 1989.

[85] Ken-ichi Iso and Takao Watanbe. Speaker-independent word recognition using a neural prediction model. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 441–444, Albuquerque, 1990.

[86] Ajay N. Jain and Alex H. Waibel. Incremental parsing by modular recurrent networks. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 364–371. Morgan Kaufmann, San Mateo CA, 1990.

[87] Mogens Høgh Jensen, Per Bak, and Tomas Bohr. Transition to chaos by interaction of resonances in dissipative systems. I. Circle maps. *Physical Review A*, 30:1960–1969, 1984.

[88] Michael I. Jordan. Serial order: a parallel distributed processing approach. Technical Report ICS-8604, Institute for Cognitive Science, University of California, San Diego, 1986.

[89] Pentti Kanerva. *Self-propagating search: A unified theory of memory.* PhD thesis, CSLI, Stanford University, 1984.

[90] H. Kantz and P. Grassberger. Repellors, semi-attractors and long-lived chaotic transients. *Physica*, 17D:75–86, 1985.

[91] Hideki Kawahara and Toshio Irino. Introduction to saturated projection algorithm for artificial network design. Preprint, NTT Basic Research Laboratories, November 1988.

[92] James D. Keeler. Comparisons between Kanerva's SDM and Hopfield-type neural networks. *Cognitive Science*, 12:299–329, 1988.

[93] Athanasios Kehagias. Optimal control for training: the missing link between hidden Markov models and connectionist networks. Preprint, Division of Applied Mathematics, Brown University, 1989.

[94] D. Kleinfeld. Sequential state generation by model neural networks. *Proceedings of the National Academy of Sciences USA*, 83:9469–9473, 1986.

[95] Teuvo Kohonen. *Self-Organisation and Associative Memory.* Springer-Verlag, Berlin-New York-London, 2nd edition, 1988.

[96] Teuvo Kohonen. The "neural" phonetic typewriter. *IEEE Computer*, pages 11–22, March 1988.

[97] Gary Kuhn, Raymond L. Watrous, and Bruce Ladendorf. Connected recognition with a recurrent network. *Speech Communication*, 9:41–48, 1990.

[98] K. E. Kürten and J. W. Clark. Chaos in neural systems. *Physics Letters*, 114A:413–418, 1986.

[99] Peter Lancaster and Kęstutis Šalkauskas. *Curve and Surface Fitting: An Introduction.* Academic Press, 1986.

[100] Kevin J. Lang. *A Time-Delay Neural Network Architecture for Speech Recognition.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1989.

[101] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–44, 1990.

[102] Alan Lapedes and Robert Farber. How neural nets work. In Dana Z. Anderson, editor, *Neural Information Processing Systems, Denver CO 1987*, pages 442–457. American Institute of Physics, New York, 1988.

[103] Yann Le Cun. A theoretical framework for back-propagation. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 21–29. Morgan Kaufmann, San Mateo CA, 1988.

[104] Yann Le Cun, John S. Denker, and Sara A. Solla. Optimal brain damage. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605. Morgan Kaufmann, San Mateo CA, 1990.

[105] Kai-Fu Lee. *Large Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1988.

[106] Yuchun Lee and Richard Lippmann. Practical characteristics of neural network and conventional pattern classifiers on artificial and speech problems. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 168–177. Morgan Kaufmann, San Mateo CA, 1990.

[107] Hong C. Leung and Victor W. Zue. Some phonetic recognition experiments using artificial neural nets. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 422–425, New York, 1988.

[108] Hong C. Leung and Victor W. Zue. Phonetic classification using multilayer perceptrons. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 525–528, Albuquerque, 1990.

[109] Esther Levin. Word recognition using hidden control neural architecture. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 433–436, Albuquerque, 1990.

[110] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to

automatic speech recognition. *Bell System Technical Journal*, 62:1035–1074, 1983.

[111] Maciej Lewenstein and Andrzej Nowak. Fully connected neural networks with self-control of noise levels. *Physical Review Letters*, 62:225–228, 1989.

[112] Zhaoping Li and J. J. Hopfield. Modeling the olfactory bulb. *Biological Cybernetics*, in press.

[113] A. Libchaber, C. Laroche, and S. Fauve. Period doubling cascade in mercury, a quantitative measurement. *Le Journal de Physique*, 43:L211–L216, 1982.

[114] Richard P. Lippmann. Review of neural networks for speech recognition. *Neural Computation*, 1:1–38, 1989.

[115] W. A. Little. The existance of persistent states in the brain. *Mathematical Biosciences*, 19:101–120, 1974.

[116] D. Lowe and A. Webb. Adaptive networks, dynamical systems and the predictive analysis of time series. In *IEE Conference Publication 313, Artificial Neural Networks*, pages 95–99, 1989.

[117] David Lowe. Adaptive radial basis function nonlinearities and the problem of generalisation. In *IEE Conference Publication 313, Artificial Neural Networks*, pages 171–175, 1989.

[118] David Lowe and Andrew R. Webb. On networks, optimised feature extraction and the Bayes decision. Memorandum 4342, Royal Signals and Radar Establishment, 1990.

[119] Helmut Lucke. Compositional representation: a representation of linguistic data for neural network recognisers. MPhil thesis, Cambridge University Engineering Department, 1989.

[120] Richard F. Lyon and Carver Mead. An analog electronic cochlea. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36:1119–1134, 1988.

[121] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, Bekeley CA, 1967.

[122] John Makhoul, Salim Roucos, and Herbert Gish. Vector quantization in speech coding. *Proceedings of the IEEE*, 73:1551–1588, 1985.

[123] B. B. Mandelbrot. *The Fractal Geometry of Nature.* Freeman, San Francisco, 1982.

[124] C. M. Marcus and R. M. Westervelt. Dyanamics of analog neural networks with time delay. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 568–576. Morgan Kaufmann, San Mateo CA, 1988.

[125] C. M. Marcus and R. M. Westervelt. Dynamics of iterated-map neural networks. *Physical Review A*, 40:501–504, 1989.

[126] J. D. Markel and A. H. Gray, Jr. *Linear prediction of speech.* Springer-Verlag, Berlin-Heidelberg-New York, 1976.

[127] Erik McDermott and Shigeru Katagiri. Shift-invariant, multi-category phoneme recognition using Kohonen's LVQ2. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 81–84, Glasgow, 1990.

[128] F. R. McInnes, Y. Ariki, and A. A. Wrench. Enhancement and optimisation of a speech recognition front end based on hidden Markov models. In *Proceedings European Conference on Speech Communication and Technology*, volume 2, pages 461–464, 1989.

[129] David McKelvie. Lexical access and language modelling for continuous speech recognition. Preprint, Centre for Speech Technology Research, Edinburgh University, 1989.

[130] Charles A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

[131] Marvin L. Minsky and Seymour Papert. *Perceptrons.* MIT Press, Cambridge MA, expanded edition, 1988.

[132] Masanori Miyatake, Hidefumi Sawai, Yasuhiro Minami, and Kiyohiro Shikano. Integrated training for spotting Japanese phonemes using large phonemic time-delay neural networks. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 449–452, Albuquerque, 1990.

[133] John Moody and Christian J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294, 1989.

[134] Yuhei Mori, Peter Davis, and Shigetoshi Nara. Pattern retrieval in an asymmetric neural network with embedded limit cycles. *Journal of Physics A: Math. Gen.*, 22:L525–L532, 1989.

[135] S. Newhouse, D. Ruelle, and F. Takens. Occurence of strange Axiom-A attractors near quasiperiodic flow on T^m, $m \leq 3$. *Communications of Mathematical Physics*, 64:35, 1978.

[136] G. Nicolis, C. Nicolis, and J. S. Nicolis. Chaotic dynamics, Markov partitions and Zipf's law. *Journal of Statistical Physics*, 54:915–924, 1989.

[137] Les T. Niles and Harvey F. Silverman. Combining hidden Markov models and neural network classifiers. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 417–420, Albuquerque, 1990.

[138] M. Niranjan and F. Fallside. Neural networks and radial basis functions in classifying static speech patterns. *Computer Speech and Language*, 4:275–289, 1990.

[139] Mahesan Niranjan and Frank Fallside. Speech feature extraction using neural networks. In L. B. Almeida and C. J. Wellekens, editors, *Neural Networks*, volume 412 of *Lecture Notes in Computer Science*, pages 197–204. Springer-Verlag, 1990.

[140] Steven J. Nowlan. Maximum likelihood competitive learning. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 574–582. Morgan Kaufmann, San Mateo CA, 1990.

[141] Stephen M. Omohundro. Efficient algorithms with neural network behaviour. *Complex Systems*, 1:273–347, 1987.

[142] Edward Ott, Celso Grebogi, and James A. Yorke. Controlling chaos. *Physical Review Letters*, 64:1196–1199, 1990.

[143] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a time series. *Physical Review Letters*, 45:712–716, 1980.

[144] Giorgio Parisi. Asymmetric neural networks and the process of learning. *Journal of Physics A: Math. Gen.*, 19:L675–L680, 1986.

[145] David B. Parker. Learning-Logic. Technical Report TR-47, Sloan School of Management, MIT, 1985.

[146] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[147] Roy D. Patterson. A pulse ribbon model of monaural phase perception. *Journal of the Acoustical Society of America*, 82:1560–1586, 1987.

[148] Barak A. Pearlmutter. Learning state-space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269, 1989.

[149] S. Peeling, R. Moore, and M. J. Tomlinson. The multi-layer perceptron as a tool for speech pattern processing research. *Proceedings of the Institute of Acoustics*, 8:307–314, 1986.

[150] Fernando J. Pineda. Dynamics and architecture in neural computations. *Journal of Complexity*, 4:216–245, 1988.

[151] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. AI Memo 1140 and CBIP paper 31, MIT, 1989.

[152] Alan B. Poritz. Hidden Markov models: a guided tour. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7–13, New York, 1988.

[153] M. J. D. Powell. Radial basis functions for multi-variable interpolation: a review. Technical Report DAMPT/NA12, Dept. of Applied Mathematics and Theoretical Physics, University of Cambridge, 1985.

[154] R. W. Prager, T. J. W. Clarke, and F. Fallside. The modified Kanerva model: results for real time word recognition. In *IEE Conference Publication 313, Artificial Neural Networks*, pages 105–109, 1989.

[155] R. W. Prager and F. Fallside. The modified Kanerva model for automatic speech recognition. *Computer Speech and Language*, 3:61–82, 1989.

[156] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vettering. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 1988.

[157] James B. Ramsey and Hsiao-Jane Yuan. The statistical properties of dimension calculations using small data sets. *Nonlinearity*, 3:155–176, 1990.

[158] Richard A. Redner and Homer F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26:195–239, 1984.

[159] Douglas L. Reilly, Leon N. Cooper, and Charles Elbaum. A neural model for category learning. *Biological Cybernetics*, 45:35–41, 1982.

[160] Stephen Renals. A connectionist approach to speech recognition. MSc thesis, Department of Artificial Intelligence, Edinburgh University, 1987.

[161] Steve Renals, Richard Rohwer, and Mark Terry. A comparison of speech recognition front ends using a connectionist classifier. In *Proceedings of FASE Speech '88*, pages 1381–1388, Edinburgh, 1988.

[162] U. Riedel, R. Kühn, and J. L. van Hemmen. Temporal sequences and chaos in neural nets. *Physical Review A*, 38:1105–1108, 1988.

[163] Richard A. Roberts and Clifford T. Mullis. *Digital Signal Processing.* Addison-Wesley, Reading MA, 1987.

[164] A. J. Robinson and F. Fallside. Static and dynamic error propagation networks with application to speech coding. In Dana Z. Anderson, editor, *Neural Information Processing Systems, Denver CO 1987*, pages 632–641. American Institute of Physics, New York, 1987.

[165] A. J. Robinson and F. Fallside. A dynamic connectionist model for phoneme recognition. In L. Personnaz and G. Dreyfus, editors, *Neural Networks: From Models to Applications*, pages 541–550. I. D. S. E. T., Paris, 1989.

[166] A. J. Robinson, M. Niranjan, and F. Fallside. Generalising the nodes of the error propagation network. Technical Report CUED/F-INFENG/TR.25, Cambridge University Engineering Department, 1988.

[167] Tony Robinson and Frank Fallside. Phoneme recognition from the TIMIT database using recurrent error propagation networks. Technical Report CUED/F-INFENG/TR.42, Cambridge University Engineering Department, 1990.

[168] N. Rochester, J. H. Holland, L. Haibt, and W. L. Duda. Tests on cell assembly theory of the action of the brain using a large digital computer. In *IRE IT-2 Synposium*, pages 80–93, 1956.

[169] Richard Rohwer. Instant solutions to perceptron-like nets. Preprint, CSTR, University of Edinburgh, 1988.

[170] Richard Rohwer. N-gram: A model for statistical processing of data. Preprint, CSTR, Edinburgh University, 1988.

[171] Richard Rohwer. The 'Moving Targets' training algorithm. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 558–565. Morgan Kaufmann, San Mateo CA, 1990.

[172] Richard Rohwer and Steve Renals. Training recurrent networks. In L. Personnaz and G. Dreyfus, editors, *Neural networks from models to applications*, pages 207–216. I. D. S. E. T., Paris, 1988.

[173] Frank Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York, 1962.

[174] Frank Rosenblatt. A model for experimental storage in neural networks. In J. T. Tou and R. H. Wilcox, editors, *Computer and Information Sciences.* Spartan Books, Washington DC, 1964.

[175] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Procressing*, volume 1, pages 318–362. MIT Press, Cambridge MA, 1986.

[176] Hiroaki Sakoe, Ryosuke Isotani, Kazunaga Yoshida, Ken-ichi Iso, and Takao Watanbe. Speaker-independent word recognition using dynamic programming neural networks. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 29–32, Glasgow, 1989.

[177] Eric Saund. Dimensionality reduction using connectionist networks. MIT AI Memo 941, MIT, 1987.

[178] J. Scheurich. Phase effects in random networks. Undergraduate dissertation, Edinburgh University, 1990.

[179] Heinz Georg Schuster. *Deterministic Chaos.* VCH Publishers, London — New York, 2nd edition, 1988.

[180] T. J. Sejnowski. Storing covariance with nonlinearly interacting neurons. *Journal of Mathematical Biology*, 4:303–321, 1977.

[181] Terrence J. Sejnowski and Charles R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[182] Stephanie Seneff. A computational model for the peripheral auditory system: application to speech recognition research. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 37.8.1–37.8.4, Tokyo, 1986.

[183] Lokendra Shastri and Venkat Ajjanagadde. From simple associations to systematic reasoning: a connectionist representation of rules, variables and dynamic bindings. Technical Report MS-CIS-90-05, Dept. of Computer and Information Science, University of Pennsylvania, 1990.

[184] G. L. Shaw and G. Palm, editors. *Brain Theory — Reprint Volume.* World Scientific, Singapore, 1988.

[185] P. Y. Simard, M. B. Ottaway, and D. H. Ballard. Fixed point analysis for recurrent neural networks. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 149–158. Morgan Kaufmann, San Mateo CA, 1989.

[186] Christine A. Skarda and Walter J. Freeman. How brains make chaos in order to make sense of the world. *Behavioural and Brain Sciences*, 10:161–195, 1987.

[187] Neil J. A. Sloane. *Lengths of cycle times in random neural networks.* PhD thesis, Cornell University, 1967.

[188] Sara A. Solla, Esther Levin, and Michael Fleisher. Accelerated learning in layered neural networks. *Complex Systems*, 2:625–640, 1988.

[189] H. Sompolinsky, A. Crisanti, and H. J. Sommers. Chaos in random neural networks. *Physical Review Letters*, 61:259–262, 1988.

[190] H. Sompolinsky and I. Kanter. Temporal association in asymmetric neural networks. *Physical Review Letters*, 57:2861–2864, 1986.

[191] Craig Stanfill and David Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228, 1986.

[192] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer-Verlag, 1980.

[193] D. W. Tank and J. J. Hopfield. Neural computation by concentrating information in time. *Proceedings of the National Academy of Sciences USA*, 84:1896–1900, 1987.

[194] Joe Tebelskis and Alex Waibel. Large vocabulary recognition using linked predictive neural networks. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 437–440, Albuquerque, 1990.

[195] Mark Terry, Steve Renals, Richard Rohwer, and Jonathan Harrington. A connectionist approach to speech recognition using peripheral auditory modelling. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 699–702, New York, 1988.

[196] Naftali Tishby. A dynamical systems approach to speech processing. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 365–368, Albuquerque, 1990.

[197] Naftali Tishby, Esther Levin, and Sara A. Solla. Consistent inference of probabilities in layered networks: predictions and generalization. In *Proceedings International Joint Conference on Neural Networks*, volume II, pages 403–409, Washington DC, 1989.

[198] Ichiro Tsuda, Edgar Koerner, and Hiroshi Shimizu. Memory dynamics in asynchronous neural networks. *Progress in Theoretical Physics*, 78:51–71, 1987.

[199] L. G. Valiant. A theory of the learnable. *Comminications of the ACM*, 27:1134–1142, 1984.

[200] Han L. J. van der Maas, Paul F. M. J. Verschure, and Peter C. M. Molenaar. A note on chaotic behaviour in simple neural networks. *Neural Networks*, 3:119–122, 1990.

[201] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

[202] C. von der Malsburg. Self-organisation of orientation selective cells in striate cortex. *Kybernetic*, 14:85–100, 1973.

[203] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:328–339, 1989.

[204] Alexander Waibel, Hidefumi Sawai, and Kiyohiro Shikano. Modularity and scaling in large phonemic neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:1888–1897, 1989.

[205] DeLiang Wang, Joachim Buhmann, and Christoph von der Malsburg. Pattern segmentation in associative memory. *Neural Computation*, 2:94–106, 1990.

[206] Raymond L. Watrous. Phoneme discrimination using connectionist networks. *Journal of the Acoustical Society of America*, 87:1753–1772, 1990.

[207] Andrew Webb and David Lowe. A hybrid optimisation strategy for adaptive feed-forward layered networks. Memorandum 4209, Royal Signals and Radar Establishment, 1988.

[208] Paul Werbos. *Beyond Regression*. PhD thesis, Harvard University, 1974.

[209] B. Widrow and D. S. Stearns. *Adaptives signal processing.* Prentice-Hall, Englewood Cliffs NJ, 1985.

[210] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[211] D. J. Willshaw and C. von der Malsburg. How patterned neural connections can be set up by self-organisation. *Proceedings of the Royal Society of London B*, 194:431–445, 1976.

[212] Hung-Chun Yau and Michael T. Manry. Iterative improvement of a Gaussian classifer. *Neural Networks*, 3:437–443, 1990.

[213] S. J. Young. Competitive training in hidden Markov models. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 681–684, Albuquerque, 1990.

[214] E. Zwicker. Subdivision of the audible frequency range into critical bands (frequenzgruppen). *Journal of the Acoustical Society of America*, 33:248–249, 1961.