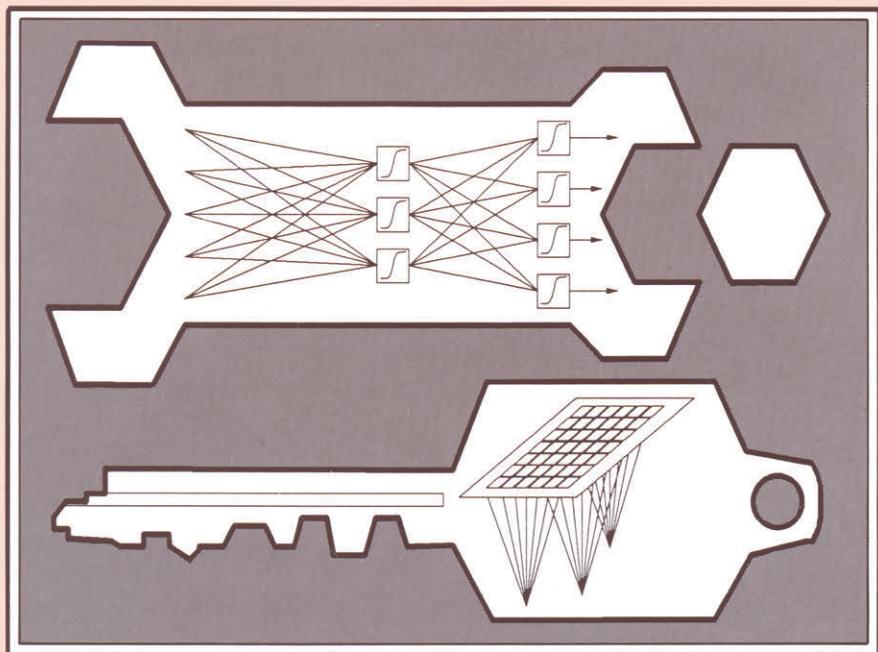

APPLICATIONS OF NEURAL NETWORKS

Edited by
ALAN F. MURRAY



Applications of Neural Networks

APPLICATIONS OF NEURAL NETWORKS

Edited by

ALAN F. MURRAY

The University of Edinburgh



Springer Science+Business Media, LLC

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4419-5140-3

ISBN 978-1-4757-2379-3 (eBook)

DOI 10.1007/978-1-4757-2379-3

Printed on acid-free paper

All Rights Reserved

© 1995 Springer Science+Business Media New York
Originally published by Kluwer Academic Publishers in 1995
Softcover reprint of the hardcover 1st edition 1995

No part of the material protected by this copyright notice may be reproduced or
utilized in any form or by any means, electronic or mechanical,
including photocopying, recording or by any information storage and
retrieval system, without written permission from the copyright owner.

DEDICATION

To Peter LeComber - who was an open-minded sceptic.

He was just beginning to believe in this neural stuff.

CONTENTS

PREFACE

xi

SECTION A - INTRODUCTION

1 Neural Architectures and Algorithms

<i>Alan Murray</i>	1
1. Introduction	1
2. The Single Layer Perceptron (SLP)	2
3. The Multi-Layer Perceptron (MLP)	8
4. Radial Basis Function Networks	14
5. Kohonen's Self-Organising Feature Map Networks	17
6. Alternative Training Approaches for Layered Networks	21
7. Summary	28

SECTION B - SUPERVISED TRAINING

B1: Pattern Recognition and Classification

2 Face Finding in Images

<i>John M. Vincent</i>	35
1. Introduction	35
2. Generation of Feature Maps	37
3. Feature Location in the High Resolution Image	53
4. Experiments with Freshly Grabbed Sequences	58
5. Conclusions	58

3 Sex Recognition from Faces Using Neural Networks

<i>B. Golomb, T. Sejnowski</i>	71
1. Introduction	71
2. Methods	75
3. Results	82
4. Discussion	83

B2 : Diagnosis and Monitoring

4	ANN Based Classification of Arrhythmias	
	<i>M. Jabri, S. Pickard, P. Leong, Z. Chi, E. Tinker, R. Coggins and B. Flower</i>	93
1.	Introduction	93
2.	Data, Preprocessing and Feature Extraction	96
3.	Single Chamber Classification	98
4.	Dual Chamber Based Classification	106
5.	Microelectronic Implementations	107
6.	Conclusions	111
5	Classification of Cells in Cervical Smears	
	<i>Mathilde E. Boon and L.P. Kok</i>	113
1.	The Need for Automatic Prescreening	113
2.	Application of Artificial Neural Networks	114
3.	The PAPNET System	115
4.	Devising a Working Protocol for PAPNET-Assisted Screening	116
5.	PAPNET-Assisted Screening Versus Conventional Screening	118
6.	Practical Implications of PAPNET-Assisted Screening	120
7.	PAPNET-Assisted Rescreening for Quality Control	121
8.	PAPNET for Detection of Cancer Cells in False-Negative Smears	122
6	Multiphase flow monitoring in oil pipelines	
	<i>Chris M. Bishop</i>	133
1.	Introduction	133
2.	Gamma Densitometry and Multi-Phase Flows	135
3.	Generation of Datasets	139
4.	The Neural Network Approach	142
5.	Prediction of Phase Fractions	144
6.	Effects of Noise on Inputs	146
7.	Novelty Detection and Network Validation	150
8.	Discussion	153

B3 : Prediction and Control

7	Electrical Load Forecasting	
	<i>Yuan-Yih Hsu and Chien-Chun Yang</i>	157
1.	Introduction	157
2.	The Load Forecasting Problem	159
3.	Kohonen's Self-Organising Feature Maps	163
4.	Application of Self-Organising Feature Maps to Day Type Identification	167
5.	Application of Multilayer Feedforward ANN to Peak Load and Valley Load Prediction	177
6.	Conclusions	181
8	On the Application of Artificial Neural Networks to Process Control	
	<i>M.J. Willis, G.A. Montague and C. Peel</i>	191
1.	Introduction	191
2.	Feedforward Artificial Neural Networks	193
3.	Dynamic Artificial Neural Networks	196
4.	Inferential Estimation	199
4.	Industrial Examples	199
5.	Concluding Remarks	216
9	Nested Networks for Robot Control	
	<i>Arjen Jansen, Patrick van der Smagt, and Frans Groen</i>	221
1.	Introduction	222
2.	The Nested Network Method	225
3.	Simulation Results	232
4.	Conclusion	235

B4 : Signal Processing

10	Adaptive Equalisation using Neural Networks	
	<i>Sheng Chen.....</i>	241
1.	Introduction	241
2.	Transversal Equaliser	245
3.	Bayesian Transversal Equaliser	249
4.	Decision Feedback Analyser	257
5.	A comparison with the MVLA	260
6.	Conclusions	262

SECTION C - UNSUPERVISED TRAINING

C1 : Temporal Sequences - Reinforcement Learning

11	TD- Gammon: A Self- Teaching Backgammon Program	
	<i>Gerald Tesauro</i>	267
1.	Introduction	267
2.	A Case Study: TD Learning of Backgammon Strategy	270
3.	TD Learning with Built-in Features: TD-Gammon 1.0	276
4.	Current Status of TD-Gammon	278
5.	Conclusions	282
12	Temporal Difference Learning: A Chemical Process Control Application	
	<i>Scott Miller and Ronald J. Williams</i>	287
1.	Introduction	287
2.	The Application	291
3.	Discussion	297
4.	Conclusion	298

C2 : Mixed- Mode (Supervised and Unsupervised) Training

13	Automated Sleep EEG Analysis using an RBF Network	
	<i>Stephen Roberts and Lionel Tarassenko</i>	305
1.	Introduction	306
2.	EEG Database	306
3.	Data Representation	306
4.	Unsupervised Learning	309
5.	Supervised Learning	313

Preface

Another book on neural networks? Surely the world does not need such a thing? This is a natural reaction - and it would be mine, were I not the editor. Any emergent and dynamic field of research needs several modes of dissemination, at different stages in its evolution. In the early stages, isolated and often seminal articles in journals break the ground and subsequent specialist conferences and journals continue to provide a platform for new theoretical and practical results and ideas. In parallel, "tutorial" style books may emerge, which gather together the fundamental results to date. Finally, there is a need for books which set out to match the currency and immediacy of a conference, but focus on a particular aspect of this topic. In the case of neural computation, I identified a need for such a volume looking at real applications. Fortunately, the field has now arrived at a level of maturity where impressive examples of important applications are not rare.

Selecting suitable material for inclusion (by invitation) was difficult, therefore, because there is now a plethora of good work. In the end, the choice was driven by a need to balance the book's coverage across a range of application areas and algorithms, a desire to include the work of some of my heroes, and personal contacts. The rule was, however, that the tasks performed by the neural networks had to be real and significant.

I see this book as targeted at both neural enthusiasts, who can skip the introductory chapter, and interested, open-minded skeptics. I hope the book will lead the latter through the fundamentals into a convincing and varied series of neural success stories - described carefully and honestly, without over-claiming. For the reader who exists somewhere between those extremes, perhaps the book will provide inspiration - perhaps your application, task or problem (the one you thought neural networks might solve) is similar to one described in this book. For the editor, this would be the best result of all. If the book can catalyse a new neural project, it will justify its publication.

The book is divided loosely into three major sections. Section A is simply an introduction to neural networks for non-specialists. Section B looks at examples of applications using "Supervised Training", and finally, section C

presents a smaller number of examples of “Unsupervised Training”. The balance represents the way things are at present. The divisions are diffuse, however, as any neural enthusiast will confirm. “Supervision” is an equivocal term, and one man’s “Signal Processing” is another’s “Pattern Recognition”.

I would like to acknowledge the support of the contributing authors, along with the flexibility and pragmatic attitude of Mike Casey at Kluwer. On the domestic front, I am grateful for the support of my University, Department, and colleagues and the forbearance of my wife, Glynis and children, Paul and Suzanne. They have had to put up with even more neuralophilia from me than usual as this book moved from gestation, into labour and through to a successful delivery.

NEURAL ARCHITECTURES AND ALGORITHMS

Alan Murray

*Department of Electrical Engineering
University of Edinburgh, Edinburgh EH9 3JL, UK*

1.1. Introduction

Artificial Neural Networks (ANNs) are intelligent, thinking machines. They work in the same way as the human brain. They learn from experience in a way that no conventional computer can and they will shortly solve all of the world's hard computational problems.

This is exactly the form of humbug that we neither want nor need. The term "Artificial Neural Network" is an emotive one, that tends to encourage such hype and nonsense - particularly in the popular media. On the other hand, ANNs are not just "hype and nonsense". Certainly, their actions may often be "just curve-fitting", but many hard problems do in fact reduce to curve-fitting in high dimensional spaces. The other common argument - that ANNs are nothing new and that they simply reformulate well-known algorithms from statistics and signal processing - is also specious. It is certainly true that ANNs do often (but not always) re-express earlier algorithms. However, the way in which they do so can provide new insights into the problem and a better implementation of the algorithm.

This book contains working examples of neural solutions to real problems. You will not find a single XOR or parity-encoder "toy problem" in these pages. The tasks are as diverse as are the neural architectures and algorithms themselves, although no attempt has been made to include an example of every shape and form of ANN. The editor's explicit rule has been that every chapter deals with an interesting and real task, in which a neural network has been shown to offer a good solution. This selection criterion has brought

about the inclusion of work from some of the neural world's leading practitioners - in this the editor is both delighted and proud.

The book is organised as follows. In chapter 1, the architectures and associated algorithms for the ANNs used in the subsequent chapters will be presented. This is intended to offer the reader new to neural networks a gentle introduction and to protect those who already know the basics from yet more descriptions of back-propagation. The remaining chapters are not "cluttered" by basics - the authors have been encouraged to concentrate on the application itself, and the details of the ANN, the training procedure and the results. The level of detail included should be sufficient to allow reimplementation, although some of the minutiae will be left to the references. For a more thorough and exhaustive review of neural algorithms and architectures *per se*, the book by Krogh, Hertz and Palmer[1] is devoted entirely to a lengthy exposition of the theory of neural computation while Lipmann's two excellent review papers[2, 3] offer a more formal condensed summary of architectures and algorithms than is attempted here. Finally[4], presents a collation of papers drawn from the 1980s and early 1990s that indicates the breadth and scope of the work that has been done since the neural renaissance that began some 14 years ago.

1.2. The Single Layer Perceptron (SLP)

Despite Minsky and Papert's success in ridiculing Perceptrons[5] we will use the SLP as the route into neural networks. The SLP sets out to act as a classifier, labelling data items in classes according to their attributes. We shall propose a simple example of a classification task involving a small number of attributes. Let us deal with a body of individuals whose heights and weights have been measured and who have been labelled as "Large" or "Not Large" by a human expert. Table 1.1 shows what a section of this database might look like.

Clearly the judgement is somewhat subjective - we all have our own ideas about what constitutes a "Large" person. At the *extrema*, however, agreement is likely. A 1.95m(6'6") individual weighing 104kg(230lbs) is large in any book! In classifier terms, each attempt at labelling will produce a slightly different decision boundary.

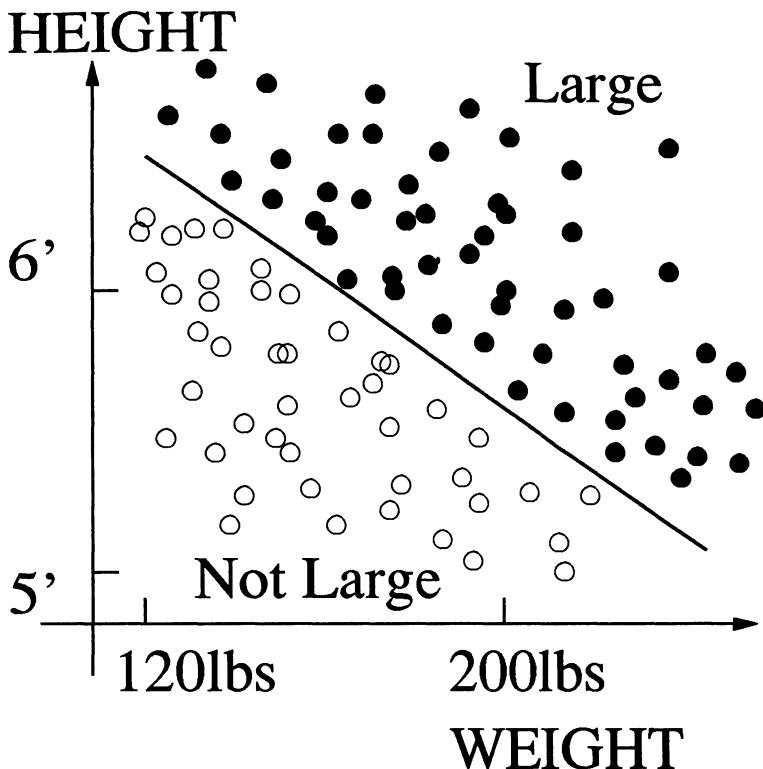


Figure 1.1 : Fictitious labelled database of individuals, labelled as "Large" (black circle) and "Not Large" (open circle) by a human expert, and an example "decision boundary".

Fig. 1.1 shows what this distribution might look like, shown as a 2-D scatter plot, with a possible decision boundary drawn in. A SLP classifier will accept the height and weight of an individual as inputs and produce a binary "1" indicating a "Large" person and a binary "0" for "Not Large" as its single output. The architecture and training for the SLP sets out to adjust (train) the internal parameters of the network to achieve this classification task for an ensemble of labelled individuals (the training set).

Fig. 1.2 shows a suitable SLP.

Individual	Height $V_{(i=1)}$	Weight $V_{(i=2)}$	Large?	Class $\tilde{V}_{(j=1)}$
1	1.95m(6'6")	104kg(230lbs)	Yes	1
2	1.6m(5'4")	54kg(120lbs)	No	0
3	1.8m(6')	60kg(130lbs)	No	0
4	1.68m(5'6")	91kg(200lbs)	Yes	1
...	
100	1.68m(5'6")	57kg(125lbs)	No	0

Table 1.1 : Fictitious labelled database of individuals, labelled as "Large" and "Not Large" by a human expert

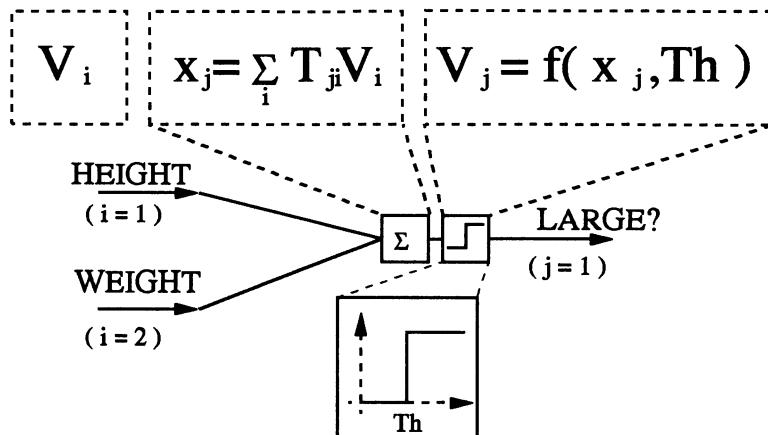


Figure 1.2 : Single-Layer Perceptron (SLP) for the database in Fig. 1.1.

The output V_j of perceptron j is given by:-

$$V_j = f_h(x_j, Th), \quad x_j = \sum_i T_{ji} V_i \quad 1.1$$

where $V_{(i=1)}$ is the height and $V_{(i=2)}$ the weight of the individual presented to the SLP, and $f_h(\cdot)$ is a hard - limiting threshold function such that:-

$$V_j = 1 \text{ if the individual is "Large", } x_j > Th$$

$$V_j = 0 \text{ if the individual is "Not Large", } x_j \leq Th$$

The quantities x_j and V_j are referred to as the **neural activity** and **state** respectively. Simply stated, the SLP output is a weighted sum of its inputs, passed through a **hard-limiting** function, with a **threshold** Th to produce a "decision".

The **synaptic weights** $\{ T_{ji} \}$ and the **threshold** Th encapsulate the SLP's function (the classification task being performed) and are the parameters that are adjusted during training. Different weight and threshold sets produce different mappings and thus different functions. At the start of training, the synaptic weights are initialised with small random values. Naturally, the perceptron's output at this stage will not be correct. Training proceeds as shown in Procedure 1.

Procedure 1 - Single Layer Perceptron

- (SLP_1) : Present the training data to the SLP
- (SLP_2) : Observe the (initially incorrect) classifier outputs
- (SLP_3) : Adjust the synaptic weights each by a small amount, to move the SLP closer to correct functionality
- (SLP_4a) : If classification is now correct for the training set, stop
 - ... or ...
- (SLP_4b) : If classification is incorrect for the training set,
 \rightarrow (SLP_1)

Thus the training set is presented to the SLP many times and weights are modified gradually, until correct classification is achieved. The *perceptron learning rule* [6] is an error-correction procedure which requires the values of the $\{ T_{ji} \}$ weights to be adapted according to the following equation, at stage SLP_3 above:-

$$\Delta T_{ji} = \eta V_i (\tilde{V}_j - V_j) \quad 1.2$$

where η is a small-valued parameter called the learning rate (usually $0.01 < \eta < 0.1$) and \hat{V}_j is the desired or target output - the label from the human expert who prepared the training database, shown in the final column of Table 1.1. In fact, with a simple example such as that in Figs 1.1 and 1.2, the weights and threshold can generally be arrived at by inspection. Simple algebra will show that weight values of 1.4 (for Height) and 0.012, (for Weight) with a threshold value of 10, will form the decision boundary shown, as:-

$$x_j = 1.4 \times \text{Height} + 0.012 \times \text{Weight} - 10, V_j = f_h(x_j, Th)$$

Naturally, these values are not unique - for instance, doubling them all has no effect. During training, therefore, the input patterns are presented to the network in random order and equation 1.2 is used to adjust the weights for each pattern in turn until all the patterns are correctly classified, *if such a solution exists*. Once the SLP has been trained to completion on the training data, the aspiration is that it will also classify novel data (i.e. individuals not included in the training data) correctly - it will generalise. It should be apparent at this stage that expanding the SLP to deal with more than 2 input characteristics - to include shoe size and hat size, for instance, is straightforward. It simply adds extra inputs to Fig. 1.2 and equation 1.1. The example quoted does, however, make the scatter plot (c.f. Fig. 1.1) 4-dimensional and thus more difficult to visualise.

In summary, the SLP is a classifier that can in principle be trained to replicate human labelling on a training set. It can deal with an arbitrarily large number of input attributes and a separate SLP must exist for every possible output classification. For instance, a separate SLP would be required to label individuals as "Male" or "Female" on the basis of the above attributes (it would also have a much more difficult task!).

1.2.1. The Problem with SLPs

In fact, the difficulty of many tasks can overwhelm an SLP easily. Fig. 1.3 shows an apparently straightforward classification task that a simple SLP cannot perform. The task is the same, but the distribution of labels is different. The crucial point is that the two classes are not linearly separable

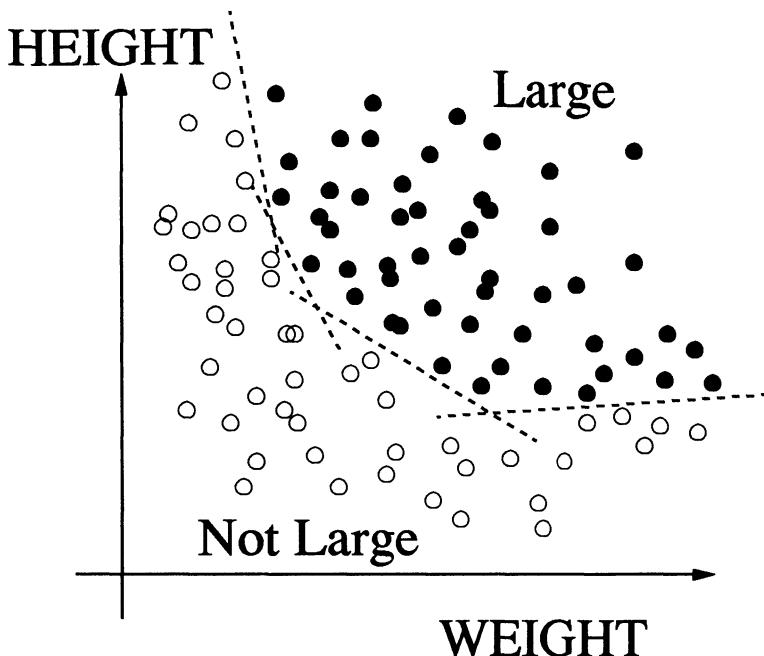


Figure 1.3 : Fictitious labelled database of individuals, where the classes are NOT linearly separable. The individuals are again labelled as "Large" (black circle) and "Not Large" (open circle) by a human expert

- it is not possible to draw a single straight line, or decision boundary, separating the "Large" from the "Not Large". With 3 input attributes, the equivalent statement is that it is not possible to draw a single flat plane (in 3 dimensions) separating the "Large" from the "Not Large". With N -dimensional inputs, planes become N -dimensional hyperplanes, but the result still holds. This was the hatchet used by Minsky and Papert to dismember the SLP in 1969 and it is indeed a fundamental limitation[5]. Minsky and Papert's exposition of this caused neural network research to stall for more than a decade.

In the early 1980s, Hopfield published his seminal papers[7, 8] and revitalised neural research. In retrospect the Hopfield network is a curiosity, it is of little practical significance and we will not discuss it further. However, it formed the spur to what is now a huge research effort, and was

thus of immense importance. The other major factor in the neural renaissance was the development[9] of the *back-propagation algorithm* for the training of Multi-Layer Perceptrons. This proved to be a much more significant *practical* move forward and has led to the application of feedforward networks to a wide range of problems in pattern recognition and classification over the last few years.

1.3. The Multi-Layer Perceptron (MLP)

The extension from SLP to MLP is straightforward. The activities of a set of perceptrons (a set of x_j as above) are passed through smooth threshold functions - often called **activation functions** or **squashing functions**. The activation function performs essentially the same overall function as the hard-limiting version above, in mapping an unbounded neural activity x_j into a bounded neural state (say $0 < V_j < 1$), with an associated threshold activity around which the transition $V_j \rightarrow 1$ occurs. However, it is S-shaped, or sigmoidal, as indicated in Fig. 1.4, such that as $x_j = \sum T_{ji}V_i$ increases, the unit's output V_j increases smoothly from 0 to 1 - not abruptly as in the SLP. This output V_j is then passed via another bank of synaptic weights $\{ T_{kj} \}$ to form another of new neural activities $\{ x_k \}$ and states $\{ V_k \}$ through another (optional) set of sigmoidal activation functions, such that:-

$$V_j = f_{sigmoid}(x_j, Th_j), \quad x_j = \sum_i T_{ji}V_i \quad 1.3$$

and

$$V_k = f_{sigmoid}(x_k, Th_k), \quad x_k = \sum_j T_{kj}V_j \quad 1.4$$

Further layers may be added *ad nauseam*, but an MLP with two layers of units, as shown in Fig. 1.4, is generally sufficient.

The units j in Fig. 1.4 are often referred to as **hidden units**, as they do not connect directly to either the input attributes $\{ V_i \}$ or to the output classification V_k .

The effect of including extra layers in this way is dramatic. Each hidden unit can now form a decision boundary (actually a graded decision boundary,

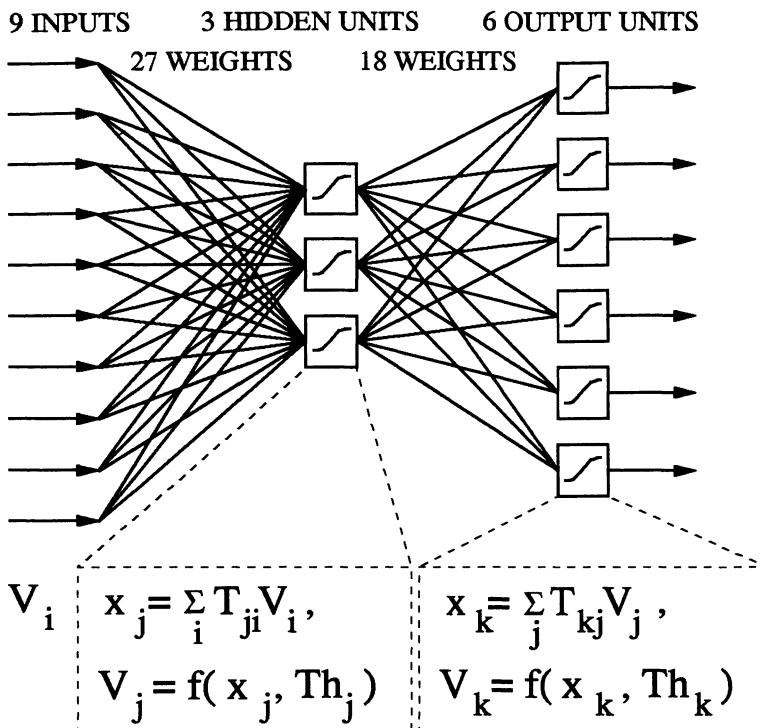


Figure 1.4 : A 9:3:6 Multi-Layer Perceptron (MLP) Network

owing to the sigmoid) and these are combined in a nonlinear manner to form the output classification. The restriction to linearly separable problems is removed and arbitrarily complex decision surfaces can be constructed. In addition, if the output is taken before application of the non-linear sigmoid function, using $x_k = \sum T_{kj} V_j$ as the MLP output rather than $V_k = f_{sigmoid}(x_k)$, then the MLP can act as a **trainable nonlinear mapping** from a set of analogue inputs $\{ V_i \}$ to a set of analogue outputs $\{ x_k \}$. In this way, the utility of the MLP is extended well beyond simple classification, as later chapters in this book show.

In classification, however, a task that is not linearly separable and thus beyond the abilities of an SLP - such as that shown in Fig. 1.3 - can be solved by an MLP, as each hidden unit contributes a (hyper-planar) decision surface

and a nonlinear combination of these is taken to perform the full classification function.

Training an MLP proceeds precisely as for an SLP - the steps are identical to SLP_1 → SLP_4 above. The only complication is brought about by the hidden units. The question arises - *How can the effect of input-hidden node weights { T_{ij} } on the outputs { V_k } be evaluated, with nonlinear hidden units interposed between them?* The most common solution is to apply the mathematical chain rule for differentiation, such that errors are back-propagated from output to input. Adaptation of the weights is achieved by minimising the mean square output error, which is defined for a single pattern as:-

$$\varepsilon = \frac{1}{2} \sum_{outputs k} (V_k - \hat{V}_k)^2 \quad 1.5a$$

For a set of N patterns, this becomes

$$\varepsilon = \frac{1}{2N} \sum_{patterns p} \sum_{outputs k} (V_{kp} - \hat{V}_{kp})^2 \quad 1.5b$$

where the error is summed over the neurons in the output layer { V_k } and over all the training patterns. The error, ε , is minimised in most cases by using gradient descent, whereby weights are adjusted in such a manner that ε is always made smaller - for example in the case of a weight T_{ab} :-

$$\Delta T_{ab} = -\eta \frac{\partial \varepsilon}{\partial T_{ab}} \quad 1.6$$

Other methods are occasionally used - such as the method of *conjugate gradients*, which involves a search along selected directions in weight space, as opposed to simple gradient descent. Such methods can sometimes yield improved learning times, but it is at the expense of added complexity.

Back-propagation of errors through the network simply involves calculating the derivative of the error with respect to the weights:-

$$\frac{\partial \varepsilon}{\partial T_{ab}} = \frac{\partial \varepsilon}{\partial x_a} \times \frac{\partial x_a}{\partial T_{ab}} = \delta_a V_b \quad 1.7$$

where $\delta_a = \frac{\partial \varepsilon}{\partial x_a}$. We can then write for an output neuron k :-

$$\delta_k = \frac{\partial \varepsilon}{\partial V_k} \times \frac{\partial V_k}{\partial x_k} = (\tilde{V}_k - V_k) \cdot V'_k \quad 1.8$$

where again \tilde{V}_k is the desired output, V_k is the actual output, and V'_k is its derivative with respect to x_k . Note that this derivative is non-zero only on or around the sloping part of the activation function. Its inclusion in equation 1.8 thus discourages learning on weights feeding into neurons that are firmly ON or OFF. If the neuron is an internal neuron j from the hidden layer, then

$$\delta_j = V'_j \sum_k \delta_k T_{kj} \quad 1.9$$

where k is over *all neurons in the layer after the hidden layer* - the output layer for our 2-layer, single-hidden-layer network. Knowledge of all the δ 's in the next layer is therefore required to compute the weight update equation for a hidden layer neuron. It is worth noting that, strictly, back-propagation will only work if the terms V'_j and V'_k can be evaluated. If there are discontinuities in the activation function, back-propagation will run into difficulties.

Training continues, as before, until ε has decreased below a given acceptable value. It can be difficult to decide on the value of this threshold, as continued training would in most cases further reduce ε but also result in the decision surface being adjusted to fit the noise rather than just the data. This is illustrated in Fig. 1.5, which shows what can happen when a network with a large number of hidden units is trained until the output error as measured on the training set reaches an absolute minimum. In such circumstances, the training process will maximise the distance between the hyperplane segments (here the decision boundary lines in 2D) and the training data at the boundary between the classes. The decision boundary will thus "wriggle" between the classes, forming a rather baroque and peculiar class separation. It will, in fact, reflect the idiosyncrasies of the class-class boundary, rather than the broad shape of the "real" decision boundary - which should ideally be as smooth and near-linear as possible. This can be proved, but it is easier to justify by looking at Fig. 1.5. It is not hard to see that the smooth decision boundary is more likely to produce the "correct" answer for a new individual,

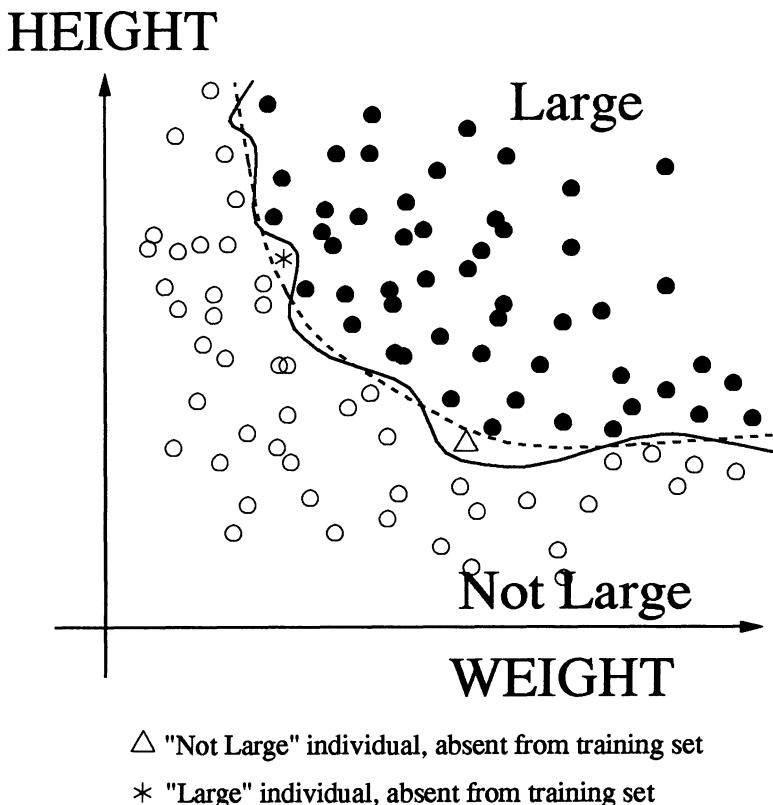


Figure 1.5 : Overfitting in MLP training. The convoluted decision boundary is formed by over-training and results in incorrect classification of "new" individuals as shown. The smoother decision boundary, formed by a shorter training phase, demonstrates better generalisation.

particularly one close to the boundary, than the convoluted boundary produced by over-fitting.

1.3.1. Over-Training in MLPs

Such "over-training" is a common cause of poor generalisation with MLPs. One solution to this problem is to divide the data set of input patterns into a training set, which is used for weight adaptation, a cross-validation set for use

when deciding when to stop training, and a test set to evaluate the generalisation performance of the trained network. In this scheme, regular experiments during training - testing the performance with the cross-validation set - will show when overfitting is becoming a danger (the performance on the validation set will decrease). Training should then cease. Alternatively, the data set can be expanded - either by collecting more data or artificially, by introducing noise-corrupted versions of the existing data. It may seem perverse that training is terminated before the training error decreases to its lowest possible value. In fact, aiming for an absolute minimum in the training set error is only a good approach when the training set is enormous - in particular when it contains many more elements than there are degrees of freedom (i.e. weights and thresholds) in the network.

There are, as one would expect, several other approaches to MLP training. Some adopt a "top-down" approach, attempting to modify the algorithm to adapt it to certain needs, such as those of analog hardware[10-12]. Others adopt a completely different approach, effectively "measuring" the derivatives in equation 1.6 above by "Weight perturbation"[13]. Superficially, this approach looks dreadful, as each weight must be perturbed in turn and its effect on the output error measured. However, it is highly successful and forms a useful alternative to back-propagation. Once again, it may be more suited to the abilities of analog hardware.

Other modifications to back-propagation abound. For instance, "momentum" is occasionally introduced - in other words, weights are given a tendency to keep changing in the same direction (rather than, perhaps, bouncing backwards and forwards)[9]. This may improve performance ... or it may not. Literally hundreds of more-or-less *ad hoc* "add-ons" and simplifications of back-propagation exist. Their success is problem-dependent, however and it is not uncommon to hear the phrase "*we just used plain vanilla back-prop*" bandied around at conferences, indicating that all such ruses have been eschewed.

Perhaps more importantly, techniques are emerging which seek to reduce the complexity of an MLP - either by starting training with a smaller network, or by "network pruning" - removing synaptic links, after training has run to completion[14]. This reduces the number of degrees of freedom (thus discouraging over-fitting) and also has the potential to reduce the

computational load in an implementation of the network. As none of the applications in this book have used such techniques, we will not cloud the main issue by dealing with them here.

In the past 3-4 years, an alternative form of layered network has proved useful in classification tasks - the Radial Basis Function (RBF) network[15, 16].

Although it resembles an MLP topologically, its functionality is quite different and will be dealt with briefly in the next section.

1.4. Radial Basis Function Networks

Fig. 1.6 shows a Radial Basis Function network. The function of an MLP may be viewed as **reducing the dimensionality of the data** - projecting a large input space on to a smaller number of hidden units and forcing the data through a bottleneck. The RBF does precisely the opposite.

An RBF network has a large number of **kernel function nodes**, in place of the hidden nodes of an MLP. These kernel nodes, however, do not implement the same multiply-and-add (weighted summation) as the hidden nodes in an MLP, however. Instead, each kernel function computes a **receptive field** and the receptive fields from individual kernels overlap. A kernel node j has a set of "weights" $\{ T_{ji} \}$ associated with it and it computes the Euclidian distance d between the set of inputs that constitute an input pattern $\{ V_i \}$ and the respective $\{ T_{ji} \}$. The RBF centre j then computes a radially symmetric function of d (usually a Gaussian) which outputs a maximum value when the input pattern $\{ V_i \}$ is near in Euclidian distance to the **prototype pattern** $\{ T_{ji} \}$ stored in that node's weights ($d = 0$). This may be expressed, for instance, as

$$V_j = \exp - \left[\frac{\sum_i (T_{ji} - V_i)^2}{2\sigma_j^2} \right] \quad 1.10$$

This may also be expressed vectorially, which aids in understanding. The kernel function computes the Euclidian distance between the vector defined by the $\{ V_i \}$, \underline{V} , and the vector defined by the $\{ T_{ji} \}$, \underline{T}_j . This distance is

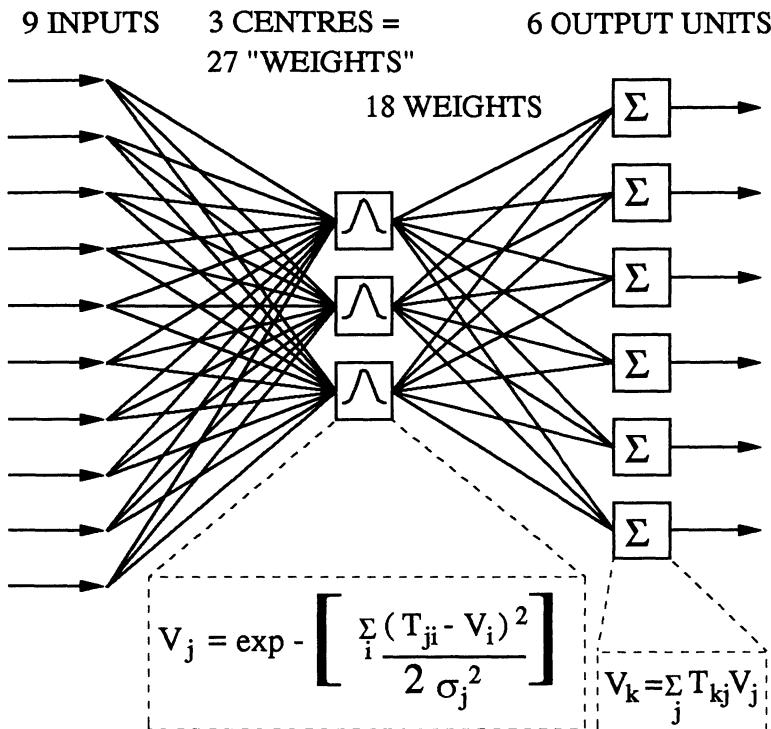


Figure 1.6 : A 9:3:6 Radial Basis Function (RBF) Network

given by $d = |\underline{Y} - \underline{T}_j|$. The RBF kernel's output (equation 1.10) is then expressed as:-

$$V_j = \exp - \left[\frac{(T_j - V)^2}{2\sigma_j^2} \right] \quad 1.11$$

This makes the function of the kernel more graphically apparent. Fig 1.7a shows the 3 kernel sites ($j=1$ to $j=3$) for a two-dimensional input (vector) \underline{V} (i.e. reduced from the 9 dimensional input in Fig. 1.6 for ease of illustration). In Fig. 1.7b, the Euclidian distances from \underline{V} to the kernel centres \underline{T}_1 , \underline{T}_2 and \underline{T}_3 (d_1 , d_2 and d_3 respectively) are shown as the *abscissae* of a Gaussian, whose ordinates are the kernel site outputs. Thus kernel site 3 is closest and provides the largest output, kernel site 2 is further down the Gaussian, while

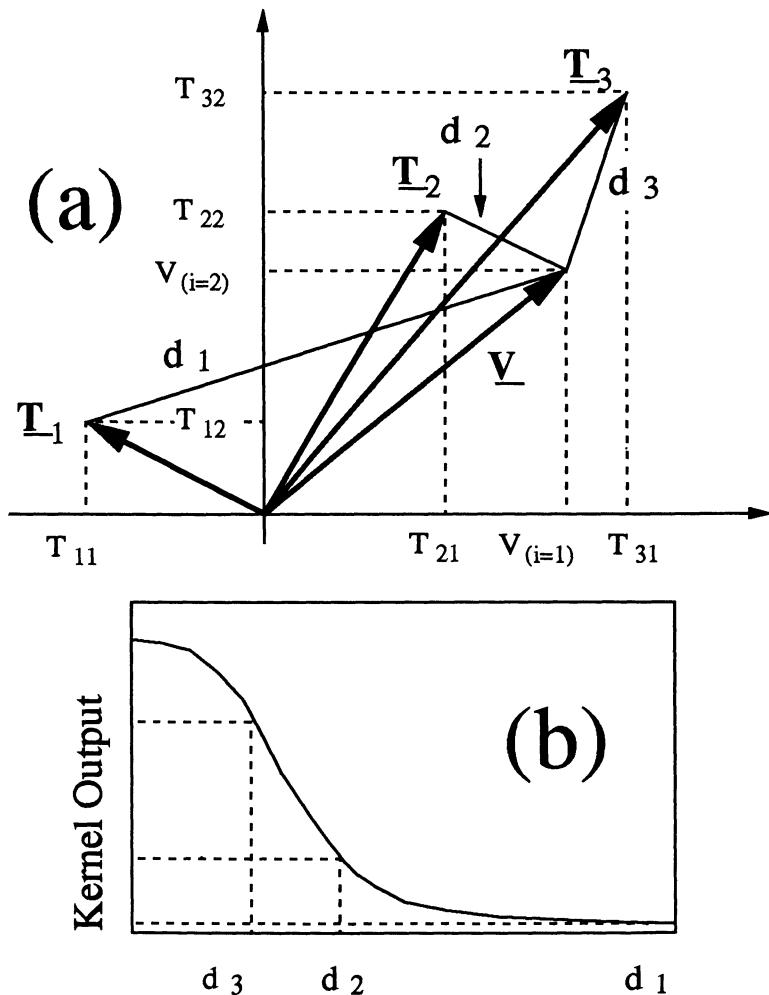


Figure 1.7 : (a) A 2-input, 3-kernel RBF kernel space and (b) the Euclidian distances d_1 , d_2 , and d_3 as abscissae to a suitable RBF curve.

site 1 produces almost no output at all.

Training strives to optimise the location in input space of these prototype patterns by using a clustering procedure (*unsupervised* learning) such as the adaptive K-means algorithm[17] or Kohonen's feature map algorithm[18],

The latter will be dealt with in the next section.

The hidden-output layer is a straightforward linear perceptron-like structure, which may, for instance, map the pattern of activation across the radial basis function nodes $\{ V_j \}$ to a classifier (perhaps 1-out-of-6) output. In this way, RBF classifiers typically give results which are comparable with those obtained from multi-layer perceptrons, and indeed the two approaches are in competition in many pattern-clustering and recognition tasks. RBF networks can potentially be trained rapidly, as only the kernel-output layer requires training *per se*. They are also in many ways more intuitive than MLPs. However, they are not biologically plausible (this may or may not matter, it depends upon your prejudices) and they present a problem for analog hardware.

1.5. Kohonen's Self-Organising Feature Map Networks

Fig. 1.8 shows a Kohonen Feature Map (K-Map) network[18].

Once again, it is a feed-forward network (actually a feed-upward network in Fig. 1.8), although a computational expedient in the training process has an effect not dissimilar to lateral inhibition. Lateral inhibition, a technique found in biological neural systems that is often useful in artificial networks, implies that each neuron provides a positive (or excitatory) input to itself and a negative (or inhibitory) input to neighbouring neurons. This results in a "Winner-Takes-All" effect - useful in making firm decisions out of tentative ones.

However, the K-Map does not implement lateral inhibition directly. The active layer in the K-Map is that shown as a planar array of neurons with no overt intra-layer links. In this respect it is topologically identical to the hidden layer of an MLP, arranged in a 2-D array. Each neuron receives a signal from each data input (shown below the neural layer in Fig. 1.7) and there are no downward links. The aim in training a K-Map is to cause neurons that are physically close to one another in the 2-D array to respond to data inputs that are similar in feature-space. Fig. 1.9 shows what this means - "Large" individuals will cause a splash of activity in one area, while "Small" ones will cause a splash in another. The training process aims to organise the weights from the inputs to the neurons in an arguably

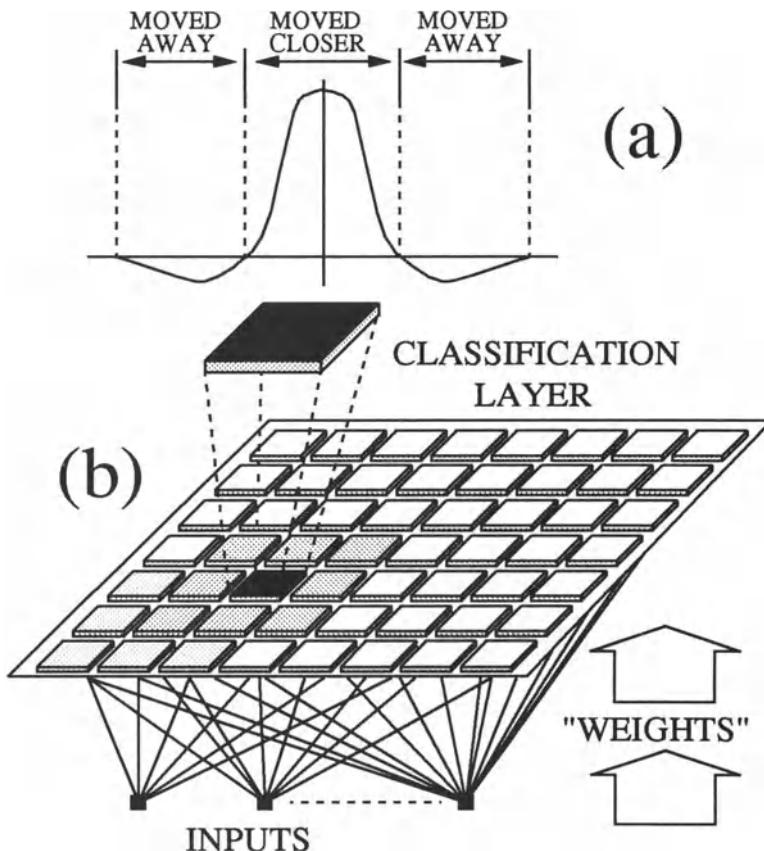
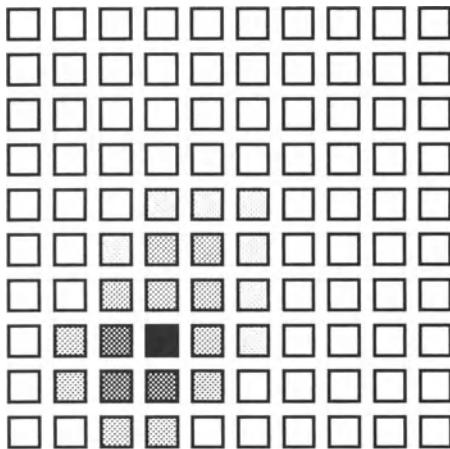


Figure 1.8 : A Kohonen Self-Organising Feature Map Network - inputs fed upward into a K-Map layer with no intra-layer connections.

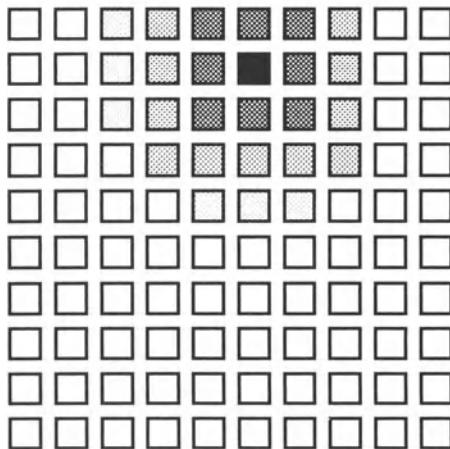
unsupervised manner (i.e. there is no *a priori* "correct" answer built in), starting from a random set of weights.

The training process is given in Procedure 2.

This procedure will cause the K-Map self-organisation referred to above to "emerge". Neurons in close proximity in the K-Map will have their weights adjusted such that they respond to similar input stimuli.



CLASS 1
("Large")



CLASS 2
("Small")

Figure 1.9 : Activity in a trained K-map - different areas in the K-Map respond to different classes in input space.

There is, in truth, a small subtlety not made clear in (K-Map_1) to (K-Map_6). When the winning neuron has its weights moved closer in input space to the input pattern (K-Map_5 above), neurons a small distance away in K-Map space are moved by a slightly smaller amount. Neurons a greater distance away may not be moved at all and neurons still further away will

Procedure 2 - Kohonen Self-Organising Feature Map

- (K-Map_1) : Initialise all weights to small random values.
- (K-Map_2) : Present a set of inputs $\{ V_i \}$ via the upward links.
- (K-Map_3) : Compute the (magnitude of the) Euclidian difference between the $\{ V_i \}$ and the $\{ T_{ji} \}$ for each neuron j .
- (K-Map_4) : Select the neuron with the smallest difference (this is the Winner-Takes-All bit).
- (K-Map_5) : Move the $\{ T_{ji} \}$ for the winning neuron j and its neighbours in the K-Map closer to $\{ V_i \}$.
- (K-Map_6) : Return to (K-Map_2) with a new set of inputs.

have their weights moved away from the input. This has the effect of driving the "splashes" of activity apart. This rather odd-sounding scheme is usually implemented by making the weight movements in (K-Map_5) proportional to a "Mexican hat" function, such as that shown above the K-Map in Fig. 1.8. This function has the effect alluded to in moving near-neighbours close, while neurons slightly further away (in the K-Map) are moved further away in input space. As training progresses, it is normal to narrow the Mexican hat to restrict weight modifications to the winning neuron only. This allows refinement of the K-Map activity patterns as training reaches its conclusion.

The K-Map is thus effectively a clustering algorithm, which causes input patterns to be grouped in K-Map (feature) space as patterns of more-or-less localised neural activity, indicating a classification. The classification is not into pre-determined, labelled classes - hence the *Self-Organising Feature Map*. This is precisely what happens in a more conventional clustering algorithm - inputs are grouped by class and labelling the classes is a separate exercise.

Fig. 1.9 shows the clustering process as patches of activity. Occasionally, the clustering is represented as a distortion of the neuron "positions" in the map - pulling neurons that are close together in input space as a result of the algorithm, close together in K-Map space. As none of the authors in the remainder of the book have chosen such a representation, we will not discuss

it further here.

In concluding this section on K-Maps, it is worth remarking that the clustering process performed by a K-Map provides a good method for selecting sites for the kernel function centres in an RBF classifier. Although this is certainly not the only use for a K-Map, it is used in this book and provides an example of the links between unsupervised training procedures and supervised classification.

1.6. Alternative Training Approaches for Layered Networks

The above architectures - Single- and Multi-Layer Perceptron, Radial Basis Function and Kohonen Map - constitute the forms used in the remaining chapters of this book. Thus far, I have outlined the operation of the back-propagation algorithm[19] and indicated how the Kohonen Map can be used to cluster input data and to infer useful kernel sites for an RBF layer. I have also alluded to variants and alternatives to back-propagation, but without expanding upon these. All of the existing relatives of the back-propagation algorithm are supervised - they require that the environment provide the correct classification or answer, in order that an error can be calculated to propagate backwards. Of the networks discussed above, only the Kohonen Map is capable of performing its clustering function without an overt teacher.

This section describes two alternative approaches to training layered networks. In the first (which is described only in outline, as it relates to an entire *genre* of approaches) the teacher merely provides a critique of the network's current performance - not an explicit "correct answer". In the second, the aim is to train a network to produce a prediction and consistency of prediction is made the goal, as opposed to simple correct prediction at all times. This apparently perverse method can result in improved training characteristics and performance for predictive ANNs.

The reason for including these apparently esoteric methods is that both have been used in the work described in later chapters. In fact, both offer radically different approaches to training that are not simply close relatives of existing algorithms. That they have found their way into working applications, such as form the foundation of this book, is remarkable and augurs well for the importance of both techniques.

1.6.1. Reinforcement Learning

The concept is simple and is illustrated schematically in Fig. 1.10.

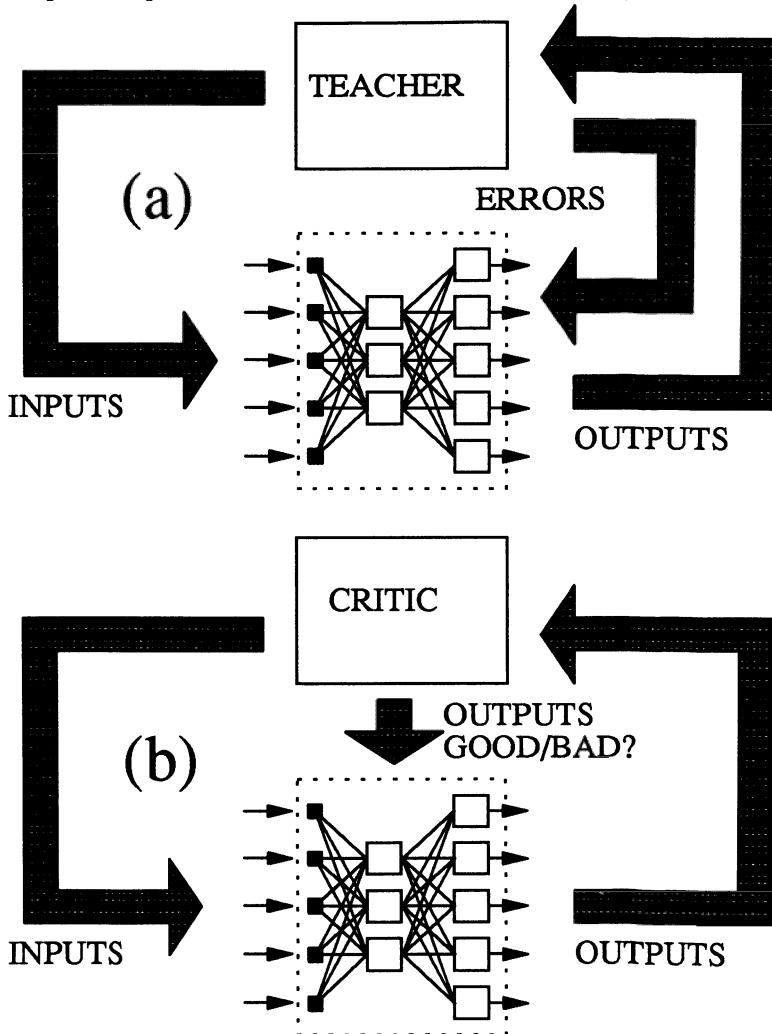


Figure 1.10 : (a) *Supervised training by a teacher and (b) Reinforcement training with the help of a critic (schematic).*

In conventional, fully-supervised learning - such as is shown in Fig. 1.9(a) -

the teacher (who generated the training set) calculates an error signal and passes it back into the NN for back-propagation. In reinforcement learning (Fig. 1.9b), the "teacher" simply offers up a criticism of the result of a network's action, which is then used to determine the next set of weight changes. As a very simple example - if weights are allowed to evolve from a random start point by adding Gaussian noise, the output from a network with all weights corrupted by a single injection of noise may be better or worse than that before the noise-corruption. A simple Reinforcement Learning algorithm would accept and retain the new weight set if the output was better and reject it if the output was worse. This crude (but surprisingly effective) algorithm has been referred to as *chemotaxis*[20], and bears a superficial resemblance to more complex genetic algorithms.

Training proceeds then takes the form shown in Procedure 3.

Procedure 3 - Reinforcement Learning

- (RL_1) : Present the training data to the ANN.
- (RL_2) : Observe the classifier outputs of this network (Network A).
- (RL_3) : Disturb each synaptic weight by a small amount, at random and under control of a Gaussian distribution.
- (RL_4) : Observe the classifier outputs of this network (Network B).
- (RL_5) : If performance is better, issue a reinforcement signal.
- (RL_6) : If a reinforcement signal has been issued, overwrite Network A with Network B.
- (RL_7) : Return to (RL_2)

The result is a form of "random walk" in weight space, modulated by the reinforcement signals (or otherwise) from the teacher, or "critic". Other reinforcement learning schemes - such as[21, 22] for example, are more complex, but all incorporate an element of randomness in order that the

solution space can be explored until a satisfactory solution is found. The reinforcement learning technique has found applications in several areas, but particularly in tasks where, by their very nature, the task involves an element of trial-and-error. For example, in robotics, the problem of obstacle avoidance has yielded to a reinforcement learning strategy[23].

1.6.2. Temporal Difference Learning

This algorithmic approach seeks to capture information relating to sequences in time more efficiently than conventional supervised learning[24].

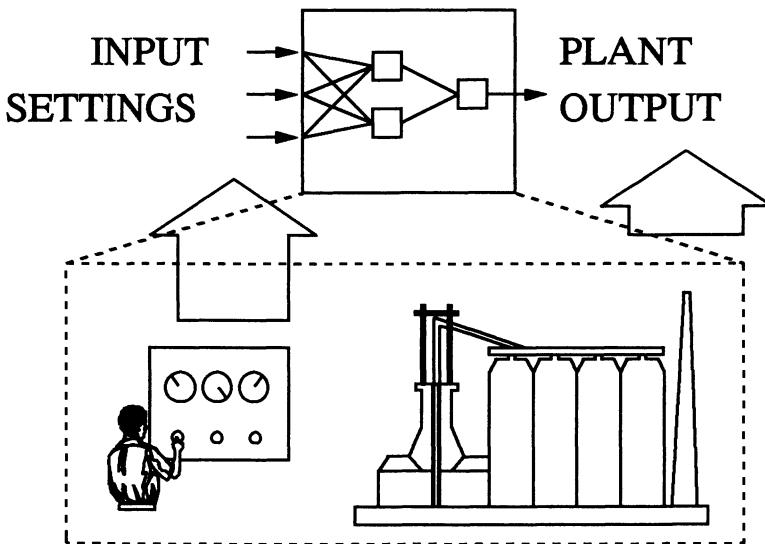


Figure 1.11 : Fictitious industrial plant and associated ANN model (predictor). The plant operator has access to 3 settings, which form the ANN inputs. The ANN attempts to predict the output at time 500s.

Fig. 1.11 shows a fictitious industrial process and Table 1.2 shows a sequence of actions and outcomes for this fictitious industrial plant, where the operator has control of three dial settings, to attempt to optimise the plant output (an output of 100% is the ultimate goal). The ANN is intended to assist this optimisation by predicting, from a sequence of operator actions, the outcome at some time in the relatively distant future.

Time t (secs)	Setting 1	Setting 2	Setting 3	Plant output
1	0.3	5.0	-3.0	70%
2	0.4	4.95	-3.5	71%
3	0.5	4.9	-4.0	72%
4	0.6	4.85	-3.5	73%
5	0.7	4.8	-4.0	74%
6	0.8	4.75	-3.5	76%
7	0.9	4.7	-4.0	76%
8	1.0	4.65	-3.5	78%
9	1.05	4.6	-4.0	79%
...
...
...
500	2.0	3.8	-3.0	91%

Table 1.2 : Fictitious sequence of actions and outcomes for an industrial plant, where the operator has control of three dial settings, to attempt to optimise the plant output.

It is intended that the ANN will attempt to predict the outcome of the sequence after 500 seconds, on the basis of a set of measurements at some time N . A conventional supervised-learning approach would form input-outcome pairs between each and every combination of settings in the sequence, using the actual sequence outcome (91%) as a "desired" output for each pair. The training data for this form of fully-supervised learning would be as shown in Table 1.3.

In temporal difference learning, the "error function" is not the difference between the desired ANN output (the 91%) and the actual ANN output. Rather it is the **difference between successive predictions**. In other words, the actual outcome of the sequence does not impact upon the training process until the end of the sequence. The sequence of inputs and predictions will then be as shown in Table 1.4.

Time t	Setting 1	Setting 2	Setting 3	Desired ANN output
1	0.3	5.0	-3.0	91%
2	0.4	4.95	-3.5	91%
3	0.5	4.9	-4.0	91%
...
500	2.0	3.8	-3.0	91%

Table 1.3 : Training data set for supervised training - plant prediction problem.

Time t	Setting 1	Setting 2	Setting 3	ANN Prediction $V_{out}(t)$
1	0.3	5.0	-3.0	Prediction 1
2	0.4	4.95	-3.5	Prediction 2
3	0.5	4.9	-4.0	Prediction 3
...	Prediction N
499	Prediction 499
500	2.0	3.8	-3.0	91% = "Prediction" 500

Table 1.4 : Training data set for Temporal Difference (TD) training - plant prediction problem.

The training process will then strive to make Prediction 1 = Prediction 2, Prediction 2 = Prediction 3 and etc., until eventually training attempts to make Prediction 499 = Prediction 500, the actual outcome of the sequence. The whole process is iterated until the predictions reach the target level of consistency. Actually, this is only one *extremum* of Temporal Difference (TD) learning - within which only predictions adjacent in time affect one another. The generic algorithm described more formally below uses a "window" during which training strives to render prediction consistent and supervised learning is the opposite *extremum*, whereby all predictions are targeted at the same value (91%).

It is not at all intuitive that this procedure - which seems to ignore the actual sequence outcome until the last possible second - should learn at all, far less learn optimally. However, it has been shown to capture temporal information better in several systems, and has been used to great effect in two chapters of

Procedure 4 - Temporal Difference Learning

- (TD_1) : Present the first set of training inputs (e.g "settings", in the example above) to the network
- (TD_2) : Observe the ANN predictor output $V_{out}(t = 1)$
- (TD_3) : Move forward one timestep $t = t + 1$
- (TD_4) : Present the next set of training inputs to the network
- (TD_5) : Observe the ANN predictor output $V_{out}(t)$
- (TD_6) : Adjust the synaptic weights each by a small amount according to equation 1.12, to move successive predictions closer together.
- (TD_7) : If sequence outcome ($t = 500$) has not yet been reached, \rightarrow TD_3
- (TD_8a) : If weight changes have become vanishingly small, stop.
... or ...
- (TD_8b) : Start again at TD_1, $t = 1$, thus presenting the entire training sequence again.

this book.

More formally, the weight changes are calculated according to:-

$$\Delta T_{ab}(t) = \eta \left[V_{out}(t + 1) - V_{out}(t) \right] \sum_{\tilde{t}=1}^{t=t} \lambda^{t-\tilde{t}} \frac{\partial V_{out}(\tilde{t})}{\partial T_{ab}} \quad 1.12$$

and the training procedure is given by Procedure 4.

In equation 1.12, λ is the adjustable parameter that sets the "window size" for $TD(\lambda)$ temporal differencing. For $\lambda = 1$, it can be shown that the resultant $TD(1)$ procedure[24] reduces to the Widrow-Hoff rule (see[25] for example) - a close relative of back-propagation. At the opposite extreme, $\lambda = 0$, only predictions adjacent in time are included in equation 1.12 and we have the most exaggerated Temporal Difference procedure, $TD(0)$. The optimal value for λ is generally between these *extrema* for prediction problems - naturally, it will be $TD(1)$ for problems which lend themselves naturally to

"conventional" back-propagation-style learning.

1.6.3. And The Rest ...

Other architectures abound - from the Boltzmann machine, where computation is probabilistic[26, 27], to the cellular neural network, where connectivity is limited[28]. VLSI implementations of neural nets are also numerous - there are several examples of analogue systems[29-31], some of which use "inaccurate" circuit components[32] while others incorporate biologically-inspired learning processes[33]. Hybrid analog-digital "pulsed" circuits have also been developed. Some are essentially analog circuits with some 0-5V signalling[34-37], while others are closer to stochastic digital computers[38]. Of course, digital accelerators - many with awe-inspiring speed specifications - have also been developed[39, 40]. Theoretical advances - large and small - abound. Recently, for instance, the cause of "intelligibility" was advanced by [41] in which a method was developed that allows a neural network to be constructed from a rule base, with surplus rules removed. Information theoretic methods are used to prune the rule base. This form of "coming together" of theoretical perspectives is a healthy result of, and hope for, the neural explosion. Coupled with successful applications such as are the subject of this book, these advances assure the longevity and credibility of neural computation as an approach to "artificial intelligence"

1.7. Summary

Hopefully, this condensed and selective summary of neural architectures and algorithms has provided the reader who is new to neural computation with a gentle "learning curve" to climb. Detail has been suppressed quite deliberately. Furthermore, fictitious "realistic" examples have been used, to avoid the ubiquitous, stylised and irritating reliance on XOR or parity problems. The remainder of the book proves conclusively that the neural paradigm, with all its variants, has matured and become almost a "mainstream" technique. The problems tackled are diverse and the neural solutions impressive. The editor hopes that this collection of working, practical ANN applications will help to let the wind out of the balloon that is over-enthusiasm, while giving a lie to the argument that "Neural Networks are just Hot Air".

If the book has a message, it is that neural computation has come of age.

References

1. J. Hertz, A. Krogh, and R.G. Palmer, in *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.
2. R.P. Lippmann, “An Introduction to Computing with Neural Nets”, *IEEE ASSP Magazine*, pp. 4 - 22, April, 1987.
3. R.P. Lippmann, “Pattern Classification using Neural Networks”, *IEEE Communications Magazine*, pp. 47 - 64, November, 1989.
4. E. Sanchez-Sinencio and C. Lau, in *Artificial Neural Networks : Paradigms, Applications and Hardware Implementations*, IEEE Press, 1991.
5. M.L. Minsky and S.A. Papert, *Perceptrons : An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1969.
6. F. Rosenblatt, “The Perceptron : A Probabilistic Model for Information Storage and Organisation in the Brain”, *Psychological Review*, vol. 65, pp. 386-408, 1958.
7. J.J. Hopfield, “Neural Networks and Physical Systems with Emergent Collective Computational Abilities”, *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554 - 2558, April, 1982.
8. J.J. Hopfield, “Neural Networks and Physical Systems with Graded Response have Collective Properties like those of Two-State Neurons”, *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088 - 3092, May, 1984.
9. D.E. Rumelhart and J.D. McLelland, in *Parallel Distributed Processing : Explorations in the Microstructures of Cognition Volume 1*, MIT Press, 1986.

10. A.F. Murray, "Multi-Layer Perceptron Learning Optimised for On-Chip Implementation - a Noise-Robust System", *Neural Computation*, vol. 4, no. 3, pp. 366-381, 1992.
11. R. Rohwer, "The "Moving Targets" Training Algorithm", *Neural Information Processing Systems (NIPS) Conference*, pp. 558-565, Morgan Kaufmann, 1990.
12. D. Nabutovsky, T. Grossman, and E. Domany, "Learning by CHIR without Storing Internal Representations", *Complex Systems*, vol. 4, pp. 519-541, 1990.
13. M. Jabri and B. Flower, "Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks", *Neural Computation*, vol. 3, pp. 546-565, 1991.
14. Y. Le Cun, J.S. Denker, and S. Solla, "Optimal Brain Damage", *Neural Information Processing Systems (NIPS) Conference*, pp. 598-605, Morgan Kaufmann, 1990.
15. D.S. Broomhead and D. Lowe, "Multivariate Functional Interpolation and Adaptive Networks", *Complex Systems*, vol. 2, pp. 321-355, 1988.
16. J. Moody and C. Darken, "Fast learning in Networks of Locally-Tuned Processing Units", *Neural Computation*, vol. 1, pp. 281-294, 1989.
17. C. Darken and J. Moody, "Note on Learning Rate Schedules for Stochastic Optimisation", *Neural Information Processing Systems (NIPS) Conference*, pp. 832-838, Morgan Kaufmann, 1991.
18. T. Kohonen, *Self-organisation and Associative Memory*, Springer-Verlag, Berlin, 1984.
19. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation", *Parallel Distributed Processing : Explorations in the Microstructure of Cognition*, vol. 1, pp. 318 - 362, 1986.

20. H.J. Bremermann and R.W. Anderson, "An alternative to back-propagation : A simple rule for synaptic modification for neural net training and memory", *Internal Report, Univ. of California, Berkeley.*, 1989.
21. P.J. Werbos, "A menu for designs of reinforcement learning over time", in *Neural networks for control*, ed. W.T.Miller III, R.S. Sutton, and P. J. Werbos, MIT Press, Cambridge, MA., 1990.
22. A.G. Barto and P. Anandan, "Pattern Learning Stochastic Learning Automata", *IEEE Trans Systems, Man and Cybernetics*, vol. 15, pp. 360-375, 1985.
23. T. Prescott and J. Mayhew, "Obstacle Avoidance through Reinforcement Learning", *Neural Information Processing Systems (NIPS) Conference*, pp. 523-530, 1991.
24. R.S. Sutton, "Learning to predict by methods of temporal difference", *Machine Learning*, vol. 3, pp. 9-44, 1988.
25. B. Widrow and M.A. Lehr, "30 years of adaptive Neural networks: Perceptron, Madaline, and backpropagation", *Proc. IEEE*, vol. 78, pp. 1415-1442, 1990.
26. G.E. Hinton, T.J. Sejnowski, and D.H. Ackley, "Bolzmann Machines: Constraint Satisfaction Networks that Learn", *Cognitive Science*, vol. 9, pp. 147 - 169, May, 1984.
27. D.G. Bounds, "Numerical Simulation of Boltzman Machines", in *AIP Conference Proceedings 151, Neural Networks for Computing, Snowbird*, ed. John S. Denker, pp. 59 - 64, American Institute of Physics, 1986.
28. F. Zou and J. Nossek, "A Chaotic Attractor with Cellular Neural Networks", *IEEE Trans CAS*, vol. 38, 1991.

29. H.P. Graf, L.D. Jackel, R.E. Howard, B. Straughn, J.S. Denker, W. Hubbard, D.M. Tenant, and D. Schwartz, "VLSI Implementation of a Neural Network Memory with Several Hundreds of Neurons", *Proc. AIP Conference on Neural Networks for Computing, Snowbird*, pp. 182 - 187, 1986.
30. C. Mead, in *Analog VLSI and Neural Systems*, Addison-Wesley, 1988.
31. A.G. Andreou, K.A. Boahen, P.O. Pouliquen, A. Pavasovic, R.E. Junkins, and K. Strohbehn, "Current-Mode Subthreshold MOS Circuits for Analog VLSI Neural Systems", *IEEE Transactions Neural Networks*, vol. 2, no. 2, pp. 205-213, 1991.
32. L.A. Akers, M.R. Walker, D.K. Ferry, and R.O. Grondin, "A Limited-Interconnect, Highly Layered Synthetic Neural Architecture", in *VLSI for Artificial Intelligence*, ed. J.G. Delgado-Frias and W.R. Moore, pp. 218-226, Kluwer, July 1988.
33. C.R. Schneider and H.C. Card, "Analog CMOS Contrastive Hebbian Networks", *Applications of Artificial Neural Networks III SPIE Proc.*, p. 1709, 1992.
34. A.F. Murray and A.V.W. Smith, "Asynchronous Arithmetic for VLSI Neural Systems", *Electronics Letters*, vol. 23, no. 12, pp. 642-643, June, 1987.
35. A.F. Murray, A. Hamilton, D.J. Baxter, S. Churcher, H.M. Reekie, and L. Tarassenko, "Integrated Pulse-Stream Neural Networks - Results, Issues and Pointers", *IEEE Trans. Neural Networks*, pp. 385-393, 1992.
36. J. Meador, A. Wu, C. Cole, N. Nintunze, and P. Chintrakulchai, "Programmable Impulse Neural Circuits", *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 101-109, 1990.
37. L.M. Reyneri, F. Gregoreti, and C. Truzzi, "Interfacing Sensors and Actuators to CPWM and CPEM Neural Networks", *Proc. International Conference on Microelectronics for Neural Networks, Edinburgh*, pp. 11-20, 1993.

38. J. Tomberg and K. Kaski, "Some IC Implementations of Artificial Neural Networks Using Synchronous Pulse-Density Modulation Technique", *Int. Journal of Neural Systems*, vol. 2, no. 1/2, pp. 101-114, 1991.
39. D. Hammerstrom, "A Highly Parallel Digital Architecture for Neural Network Emulation", *VLSI for AI and Neural Networks*, pp. 357-366, Plenum, 1991.
40. U. Ramacher, W. Raab, J. Anlauf, U. Hachmann, J. Beichter, N. Bruls, R. Manner, J. Glas, and A. Wurz, "Multiprocessor and Memory Architecture of the Neurocomputer SYNAPSE-1", *Proc. International Conference on Microelectronics for Neural Networks, Edinburgh*, pp. 227-232, 1993.
41. R.M. Goodman, P. Smyth, C.M. Higgins, and J.W. Miller, "Rule-Based Neural Networks for Classification and Probability Estimation", *Neural Computation*, vol. 4, no. 6, pp. 781-804, 1992.

FACE FINDING IN IMAGES

John M. Vincent

*BT Laboratories
Martlesham Heath
Ipswich
UK IP5 7RE*

1 INTRODUCTION

This chapter describes work undertaken at BT Laboratories to produce a system, based on the use of neural network feature detectors, to robustly locate and track features in digital image sequences. We have concentrated on the location of eyes and mouths in human head-and-shoulders images, although the techniques described should be applicable to determining the position of localised features in other objects.

Our solution to the feature location problem is the Hierarchical Perceptron Feature Locator (HPFL) system. This consists of a coarse resolution stage followed by a high resolution stage. The first stage generates search regions for eyes and mouth and the second stage searches inside these regions to accurately locate the individual feature points. Both stages employ Multi-Layer Perceptrons (MLPs) for feature detection and their outputs are post-processed in order to protect against errors.

The possibility of training MLPs to detect and locate eyes and mouths was investigated within a connectionism project called CONNEX [1]. Early work showed that neural networks appeared to have advantages over more traditional pattern recognition techniques [2], [3]. Translation invariance was achieved by using a window which was scanned across the image and feeding data into feature detecting neural nets. Furthermore, reduced computational complexity and greater reliability was achieved by adopting the two-stage multi-resolution approach [2], [4], [5], [6], [7], [8], [9].

HPFL has undergone a series of developments and improvements. Early versions simply used neural nets to locate features at different resolutions. The more advanced type of HPFL system is shown schematically in Figure 1. The low resolution version of the image to be searched is generated by optimised filtering and subsampling. The

low resolution stage became a search region generator and performance of the MLPs was improved by developing a customised version of backpropagation. MLP feature locators are scanned across the low resolution image to generate candidate search regions for each type of feature (centres of eyes, centre of mouth and centre of face). A low resolution binary image containing search regions for one feature is called a feature map. However, it was realised that it would be impossible to rely on the outputs of MLPs without having a mechanism for detecting and protecting against errors; otherwise it is prone to the location of spurious features, and sometimes fail to locate features that are present. Therefore it was decided to post-process the outputs of the MLPs using symbolic reasoning, thereby turning HPFL into a hybrid intelligent system. Eyes and mouths form an isosceles triangle in 3D object space and this geometric constraint leads to a set of rules with which to generate cleaned up search regions. A second technique to improve performance is to use inter-frame knowledge. Constraints can be put on the extent of allowable motion between frames, and this is used to remove transient spurious errors in the feature maps.

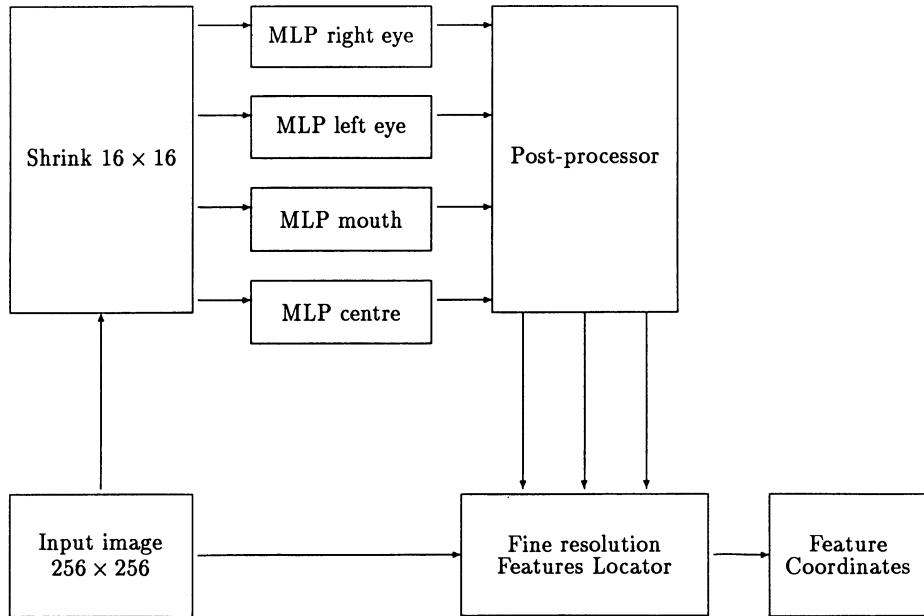


Figure 1 Scheme of the Hierarchical Perceptron Feature Locator.

The high resolution stage was also much improved by having a structured representation of eyes and mouths [7], [8]. Each of these compound features is represented by a

constellation of simple localised features called micro-features. Neural networks are trained to detect these micro-features. The window scanning technique is used to build up an array of responses for each micro-feature inside its search region. The neural networks are trained to act as Bayesian classifiers and this means that probabilistic reasoning may be applied to their outputs. The aim of the probabilistic reasoning is to find the most likely configurations of micro-features and this requires calculation of likelihood levels. The calculations combine the neural outputs with statistical knowledge of constellation geometry. In order to reduce the computational cost of the search process, to become insignificant in comparison with the neural processing, a fast algorithm, called spiral scanning, was devised by this author. Spiral scanning is presented and discussed in this chapter.

The component parts of the HPFL, shown schematically in Figure 1, are described in sections 2 and 3. Emphasis is placed on the learning algorithm and results of training. Section 4 describes results of real-world tests. These tests have provided essential feedback to algorithm development, particularly in regard to the effects of lighting conditions, and the direction of current research is discussed briefly.

2 GENERATION OF FEATURE MAPS

2.1 Obtaining candidate feature points in the low resolution image

The feature detectors required in the low resolution stage of the HPFL are required to perform a subtly different function to a straightforward classifier, or those required in the high resolution stage. The purpose of the low resolution feature detectors is to locate candidate feature points, therefore the detection of a point which is not a feature point (a false positive) is much less serious than the failure to detect a point which is a feature point (a false negative). False positives can be pruned out by various methods at a later stage, but false negatives are less easily recovered from. This conditions the way the detectors are trained, as will be described.

MLPs with a 5x5 input window, two or three hidden 2nd degree neurons and a single output neuron have been used as detectors to perform candidate feature point classification for the HPFL. Each detector is scanned across the 16x16 pixel image, and its thresholded output is used to create a 16x16 binary image known as a feature map. Unlike the earlier low resolution experiments, separate detectors are used for left and right eyes. In total there are therefore four feature maps, one each for the left

eye, right eye, mouth and centre of face. After further processing (to be described later) these are passed to the high resolution stage, where they are used to define search regions for the feature detectors in the high resolution image.

The use of a single layer perceptron to detect eye positions in 16x16 pixel images showed some promise [6]. Its performance is perhaps surprising in view of the fact that its decision surface is a single hyperplane in 25 dimensional pattern space. In order to produce more complex decision surfaces, an MLP with a number of hidden nodes should be used. However there is a trade-off between number of hidden nodes and ability to generalise. In order to achieve a balance between ability to generalise, and ability to form more complex decision surfaces, "2nd degree" neurons have been used.

The output of a 2nd degree neuron is given by

$$y_j = f(\theta_j + \sum_i \mathbf{w}_{ij} \cdot \mathbf{x}_i + \sum_i \mathbf{v}_{ij} \cdot \mathbf{x}_i^2)$$

where θ_j is a bias term, \mathbf{w}_{ij} are the weights associated with the original inputs and \mathbf{v}_{ij} are the weights associated with squared inputs. The simple first order neuron function can be obtained by setting these weights to zero. The 2nd degree neuron is a small step in the direction of Higher Order Neural Nets (HONNs) [10], and can form a much richer set of decision surfaces than the linear neuron whereas it requires a 2 layer MLP with a minimum of $N + 1$ hidden linear neurons to form a closed decision surface in N dimensional pattern space.

Because the squared terms can be generated by pre-processing the input pattern vector, nets with 2nd degree neurons in the first layer can be trained using the conventional backpropagation algorithm or variations of it.

2.2 MLP training

The low resolution MLPs are trained using a customised version of the backpropagation algorithm [4], [9]. The backpropagation algorithm is a simple yet highly effective method for training neural networks containing neurons with non-linear neurons [11] and it has done much to popularise connectionism. However, as is the case in many human endeavours, blind application of a method leads to sub-optimum and mediocre results. Knowledge of a class of problem needs to be exploited and a general purpose

method suitably adapted. The training algorithm described below was designed for training MLPs to detect features embedded in a large mass of other data.

There are two broad aspects to the learning: the selection of training parameters and the type of algorithm used in a single training session. A 'single training session' is defined here as a sequence of epochs resulting in a particular set of weight values. Prior to the first epoch an MLP is randomly initialised and during each epoch all training data will have been presented to the MLP and weights adjusted. At the end of a single training session a trained MLP is produced. The quality of the MLP is affected by the chosen parameters but it is difficult, perhaps impossible, to predict the effect of these parameters. Therefore it is necessary to explore parameter space by running a series of individual training sessions and selecting the best MLP. This is referred here as batch training. Batch training raises interesting issues, such as computational complexity and ways of automating the search process.

Algorithm for a single training session

(1) Classification of pattern vectors

As the MLP is scanned across the source image there is one occasion when the scanning window is closest to the desired feature. When this occurs the expected output of the MLP is a high value (1.0), and the pattern vector on its input will be referred to as a *feature vector*. In all other positions the MLP output is expected to be low (0.0). The pattern vectors that form the input to the MLP on these occasions will be referred to as *background vectors*.

(2) Presentation ratio

The MLPs were trained on 30 images selected from a set of 60 head-and-shoulders images. For each low resolution image there are 144 MLP input window positions, corresponding to 143 background vectors and 1 feature vector. This is calculated from the expression for the total number of scannable positions, W , in an image.

$$W = (Width_{image} - Width_{window} + 1) \cdot (Height_{image} - Height_{window} + 1)$$

If each vector is presented only once during a training epoch then the contribution to weight updating made by the feature vector is swamped by the effects of the background vectors; failure to detect the feature vector means that only 1 in 144 pattern vectors are misclassified, a misleading success rate of 99.3%. This problem is avoided by presenting the two classes of pattern vector in a 1:1 ratio; each time a background

vector is presented to the MLP it is followed by the feature vector from the same image.

(3) Selective training

Because of the need to ensure that all feature vectors are detected, whilst minimising the number of background vectors that result in false positives, a selective training procedure was used. (Selective training is a useful general technique recommended by Kelly [12].) In this procedure a pattern vector is used only if the MLP's current response to it is considered unacceptable. In the selective training procedure the feature vectors are presented to the net at the end of a training epoch. If any feature is misclassified, then training enters selective mode, otherwise it enters the non-selective mode. In selective mode only feature vectors are presented. Training remains in selective mode until all features are correctly classified. In either mode backpropagation is only applied for those that are misclassified. Misclassification is deemed to have occurred if the net gives an output < 0.5 for an expected output of 1.0, or if the output is > 0.5 for an expected value of 0.0.

(4) Stopping criteria

It is important to know when performance was optimised during training. Figures 2-5 show examples of how the MLPs behave during training. The database of head-and-shoulder images, made available for the training phase, is partitioned into two parts: 'training' data (Tr) and 'test' data (Te). Two performance measures are used to assess the ability of an MLP to identify a feature in a database, T : average size of search regions, A^T , and feature retention rate, Ret^T . Average size of search regions, A^T , is the percentage of pixels in an image which belong to the search region averaged over the database, T , and feature retention rate is the percentage of feature points in the database which fall inside their respective search regions. Feature rejection rate, Rej^T , may be similarly defined.

More explicitly, the two types of performance measure are defined as follows:-

$$\begin{aligned}
 A^T &= 100. \frac{\text{Average search region size}}{\text{Image area}} \\
 &= 100. \frac{\sum_{m=1}^{m=M^T} \sum_y \sum_x \text{BelongsToSearchRegion}(x, y, m)}{M^T \cdot \text{Width}_{image} \cdot \text{Height}_{image}} \%
 \end{aligned}$$

$$Ret^T = 100. \text{Fraction of feature points retained}$$

Figure 2 Performance versus training time for the left eye's training set (3 hidden neurons, seed=10, learning-rate=0.025)

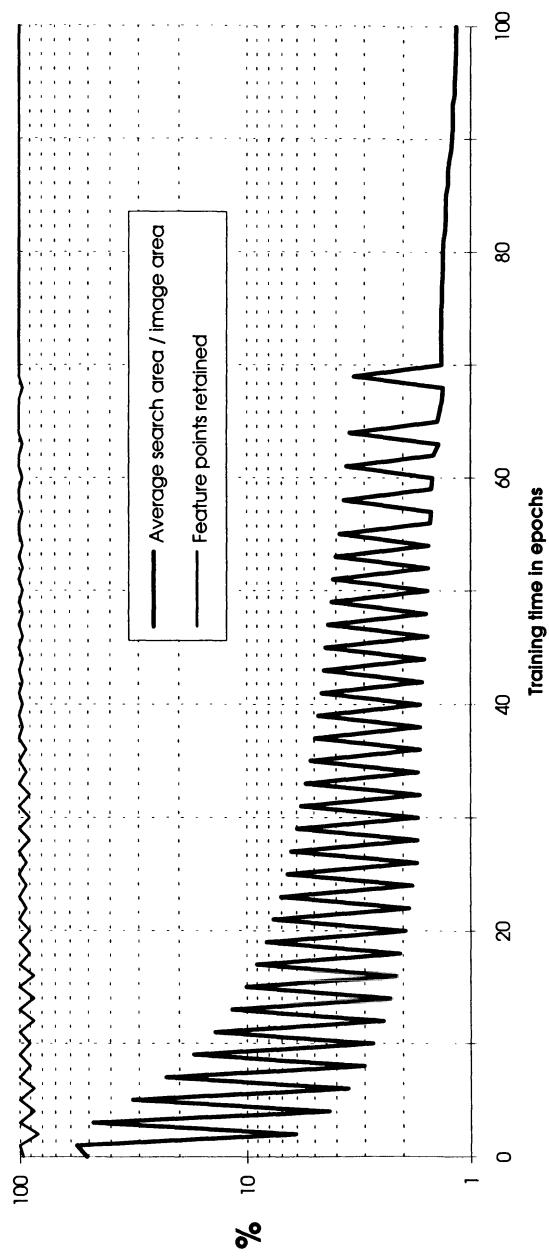


Figure 2 graph1

Figure 3 Performance versus training time for the left eye's test set (3 hidden neurons, seed=10, learning-rate=0.025)

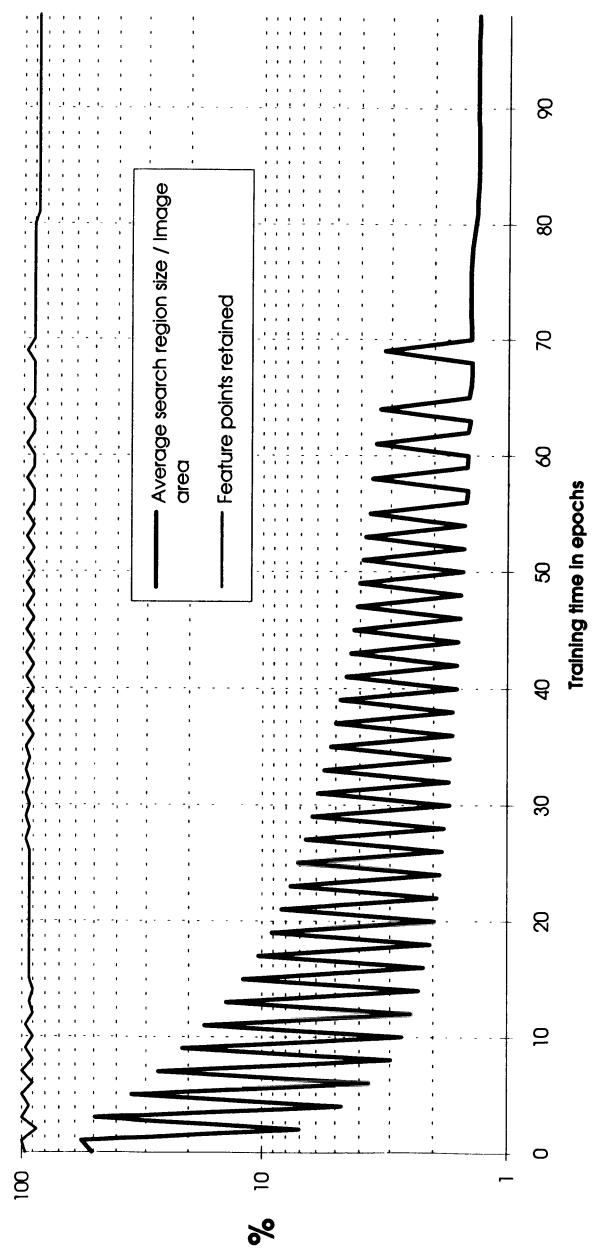


Figure 3 graph2

Figure 4 Performance versus training time for the right eye's training set (3 hidden neurons, seed=100, learning-rate=0.025)

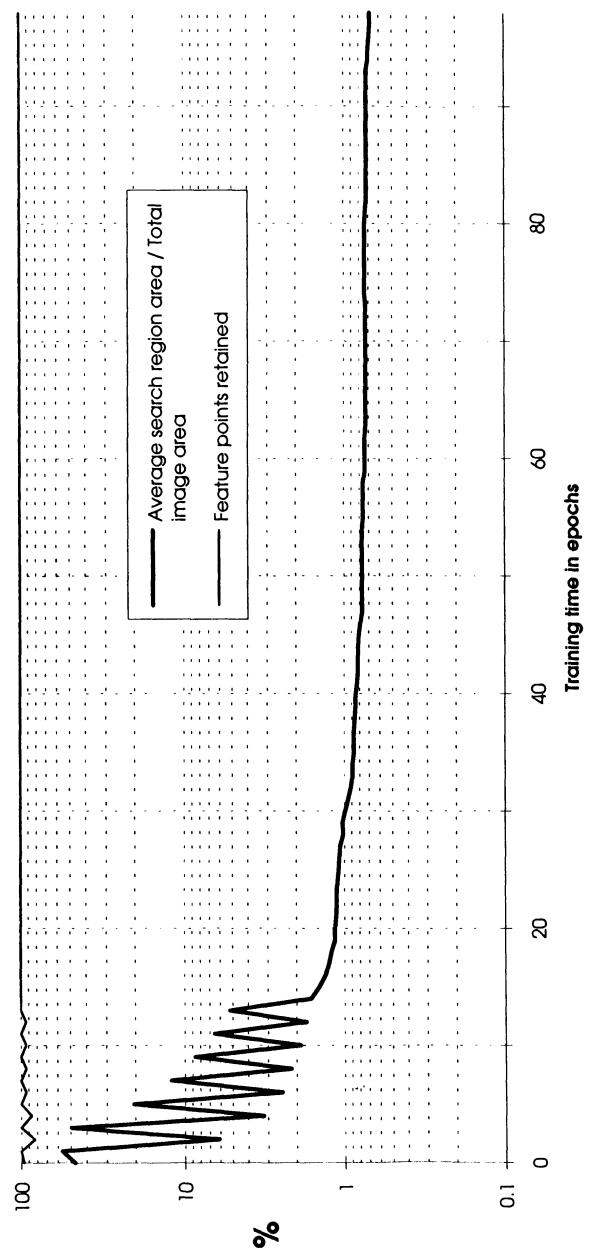


Figure 4 graph3

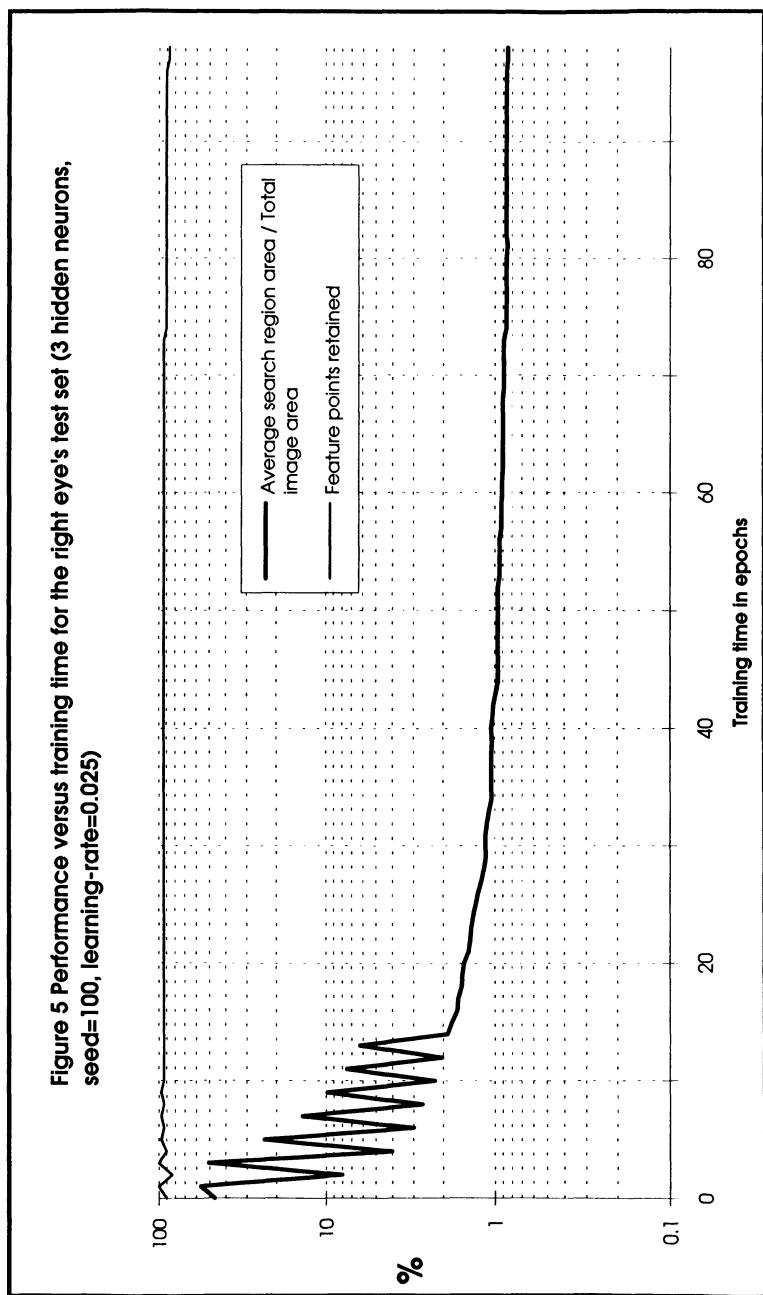


Figure 5 graph4

$$\begin{aligned}
 &= 100 \cdot \frac{\sum_{m=1}^{m=M^T} \text{FeaturePointInsideSearchRegion}(m)}{M^T} \% \\
 \text{Rej}^T &= 100 - \text{Ret}^T \%
 \end{aligned}$$

where the predicates *BelongsToSearchRegion* and *FeaturePointInsideSearchRegion* return either 0 or 1 according to their truth value. *BelongsToSearchRegion*(x, y, m) is true if the pixel (x, y) in the relevant feature map for the m^{th} image belongs to the search region and *FeaturePointInsideSearchRegion*(m) is true if the true feature point for the m^{th} image falls inside the relevant search region. The number of images in the T^{th} database is denoted by M^T .

The training data is presented to the MLPs in order to calculate errors for backpropagation and at the end of each epoch in order to provide mode control. Figures 2 and 4 show learning curves for training data. In addition the test data is presented at the end of each epoch in order to calculate more objective performance figures. This is because the weights are never affected by these latter calculations, except in order to deduce optimum stopping time. Figures 3 and 5 show the learning curves for test data and which were generated at the same time as Figures 2 and 4 respectively. The curves for the test data reveal over-training which means they give a better indication of true performance and the optimum stopping time. The weights are never affected by these latter calculations.

Figures 2 to 5 reveal three distinct phases: oscillatory, convergent and over-fitting. The oscillatory phase occurs when the mode alternates between selective and non-selective. Suddenly the selective mode is no longer required and the weights converge towards stable and satisfactory values but eventually the MLP overfits the training data. Let e_{stable} be the epoch when the oscillatory phase stops, e_{stop} be the optimum stopping time and e_{max} be the maximum number of epochs allowed in a single training session. Optimum training time is such that the rejection rate does not increase after the stabilisation phase.

$$\text{if}(\text{Rej}^{Te}(e) > \text{Rej}_{e_{stable}}^{Te}) \text{ then } (e_{stop} = e - 1 \& \text{stop})$$

The algorithm for a single training session, using the above features, is described below. Define a data base, T , to be a set of low resolution images plus manually measured feature points. A 5x5 window may have one of a number of positions. Let a positioned window have index w and let there be W positions. A feature point's position is also the position of the window containing the feature vector. Therefore feature points are

referred to in terms of window indices. Let there be F types of measured features and let $w_{m,f}^T$ be the index for the type- f feature in the m^{th} image in database T .

$$T = \{(\mathbf{I}_m^T, \mathbf{w}_{m,1}^T, \dots, \mathbf{w}_{m,f}^T, \dots, \mathbf{w}_{m,F}^T) | 1 \leq m \leq M^T\}$$

The following functions are used in the algorithm:-

- $mlp = \text{RandomMlp}(\mathbf{H}, s)$ creates an MLP, mlp , with a single hidden layer of H neurons and randomises the weights using a random number and seed s .
- $\vec{x} = \text{ExtractPatternVector}(\mathbf{I}, w)$ constructs a pattern vector, \vec{x} , from the contents of a window inside image I and which has index w .
- $a = \text{ActivateMlp}(\vec{x}, mlp)$ inputs a pattern vector, \vec{x} , to an MLP and returns the output activation level, a .
- $\text{BackpropagateMlp}(\vec{x}, mlp, t, \eta)$ up-dates the weights of an MLP by applying an incremental cycle of backpropagation, where t is the desired output for input \vec{x} and η is the learning-rate.
- $Rej^{Te} = \text{RejectionRate}(T, f, mlp, \theta)$ returns the percentage of type- f feature points correctly classified by an MLP. A correct classification of a feature point is deemed to have occurred when the response of an MLP to the feature vector is greater than the specified threshold level, θ .

The mode of operation, selective or not selective, is indicated by a flag, $\phi_{selective}$. This is the pseudo-code for training an MLP to detect the type- f feature.

```

 $mlp = \text{RandomMlp}(\mathbf{H}, s);$ 
 $\phi_{selective} = \text{OFF};$ 

 $e = 1 \text{ to } e_{\max}$ 
 $.\quad m = 1 \text{ to } M^{\text{Tr}}$ 
 $.\quad \quad \quad \text{if}(\phi_{selective} = \text{OFF})$ 
 $.\quad \quad \quad \quad w = 1 \text{ to } W$ 
 $.\quad \quad \quad \quad \quad \vec{x} = \text{ExtractPatternVector}(\mathbf{I}_m^{\text{Tr}}, w);$ 
 $.\quad \quad \quad \quad \quad a = \text{ActivateMlp}(\vec{x}, mlp);$ 
 $.\quad \quad \quad \quad \quad \text{if}(w = w_{m,f}^{\text{Tr}} \& a < \theta) \text{BackpropagateMlp}(\vec{x}, mlp, 1, \eta);$ 

```

```

else if( $w \neq w_{m,f}^{Tr}$  &  $a > \theta$ ) BackpropagateMlp( $\tilde{x}$ , mlp, 0,  $\eta$ );
 $\tilde{x} = ExtractPatternVector(I_m^{Tr}, w_{m,f}^{Tr});$ 
if( $a < \theta$ ) BackpropagateMlp( $\tilde{x}$ , mlp, 1,  $\eta$ );
else  $\tilde{x} = ExtractPatternVector(I_m^{Tr}, w_{m,f}^{Tr});$ 
a = ActivateMlp( $\tilde{x}$ , mlp);
if( $w = w_{m,f}^{Tr}$  &  $a < \theta$ ) BackpropagateMlp( $\tilde{x}$ , mlp, 1,  $\eta$ );
else if( $w \neq w_{m,f}^{Tr}$  &  $a > \theta$ ) BackpropagateMlp( $\tilde{x}$ , mlp, 0,  $\eta$ );
 $\tilde{x} = ExtractPatternVector(I_m^{Tr}, w_{m,f}^{Tr});$ 
if( $a < \theta$ ) BackpropagateMlp( $\tilde{x}$ , mlp, 1,  $\eta$ );
RejTe = RejectionRate(Te, mlp,  $\theta$ );
if(RejTe > 0)  $\phi_{selective}$  = ON else  $\phi_{selective}$  = OFF;

```

Batch training

MLPs were constrained to have a single hidden layer, the window size was fixed at 5x5 and maximum training time was 100 epochs. The parameters which could be varied from session to session were:

- Number of hidden neurons (Size)
 - The seed for the random number generator (Seed)
 - Learning rate (η)

A low learning rate, $\eta = 0.025$, was found to be the best amongst those tried out. However, many more parameter sets could have been tried out. Table 1 gives details for the best MLPs when trained on a sixty image database. The database was made at BT Laboratories [1] and training was executed on a 100 Mips Apollo DN10000.

Table 1 The best MLPs obtained during batch training ($\eta = 0.025$)

Feature	Epochs	Size	Seed	Features retained/%	Avg. search region size/%
Left eye	70	3	10	90	1.42
Right eye	73	3	100	93.3	0.89
Mouth	41	2	50	90	2.33
Centre of face	38	2	10	93.3	0.89

It is interesting to compare these figures with those from an earlier, less rigorous, phase of training. The results for the test set are shown in Table 2. The number of rejections (failure to detect a feature vector) must be read in conjunction with the figure

for mean search area, which is a measure of the number of false positives. For good net performance both of these figures should be low; it is perfectly possible to have no rejections with a search area of 100%, but this is clearly undesirable.

Table 2 Results of non-batch training ($\eta = 0.025$)

Feature	Epochs	Size	Seed	Features retained/%	Avg. search region size/%
Left eye	50	2	0	96.7	4.3
Right eye	50	2	0	96.7	1.8
Mouth	50	2	0	90.0	3.3
Centre of face	50	2	0	100.0	2.0

Batch training is computationally expensive and lack of automation made it fairly labour intensive. More recent work has involved a 600 image database and batch training requires days of cpu time. Investigation into fast batch training is therefore required. It is clearly important to have fast individual training sessions and fortunately selective training speeds up training as well as improving the quality of the MLPs.

2.3 Post-processing of feature maps

As stated earlier, for the low resolution stage to perform well, it is necessary to eliminate false negatives whilst keeping the number of false positives low. The function of the low resolution stage's post-processor is to reduce the number of false positives, whilst retaining the true feature points. It has three consecutive stages; spatial pruning, temporal pruning and pixel expansion [4], [9].

Spatial Pruning

Spatial pruning is an intra-frame process that exploits geometric constraints imposed by facial structure on eye and mouth feature locations.

Consider a triple combination of candidate feature points, one point being taken from each of the feature maps generated from a single frame. The pixels have coordinates \bar{r}_L , \bar{r}_R , \bar{r}_M for candidate left eye, right eye and mouth positions respectively, as shown in Figure 6. Some combinations are clearly not feasible, because they would imply, for instance, that the viewpoint is behind the head. This would happen if the candidate left eye position is to the right of the candidate right eye position. Other combinations might imply an unnaturally distended face. A set of geometric feasibility criteria can be established, and each possible triple that can be generated from the feature map can

be tested to establish whether it satisfies these criteria. If not, that triple is rejected; this is the basis of the spatial pruning algorithm.

The parameters used in the HPFL low resolution stage post-processor to establish a set of criteria are shown in Figure 6. L is the spacing between the candidate eye points and θ is the angle that L makes with the horizontal. (U, V) are the coordinates of the candidate mouth point with respect to a frame of reference that has as its principal axes the line L , and an orthogonal line intersecting it at its midpoint.

The following inequalities are tested:-

$$L_{min} < L < L_{max}$$

This implies that all images are at approximately the same scale, and restricts head rotation.

$$|\theta| < \theta_{max}$$

This restricts the range of orientations in the image plane.

$$|U/L| < (U/L)_{max}$$

This restricts the mouth position to lie approximately on the perpendicular axis through the midpoint between the eyes.

$$(V/L)_{min} \leq (V/L) \leq (V/L)_{max}$$

This rejects unnaturally compressed or distended face shapes.

The maximum and minimum allowed values are calculated from the training set of images, and relaxed to allow for greater variability in test images. Pixels in the feature map that do not belong to at least one triple that satisfies the above criteria are considered to be false positives, and are deleted.

The spatial pruning algorithm was applied to the images in the training and test sets. If all triples in a particular image failed to meet the pruning criteria then pruning was

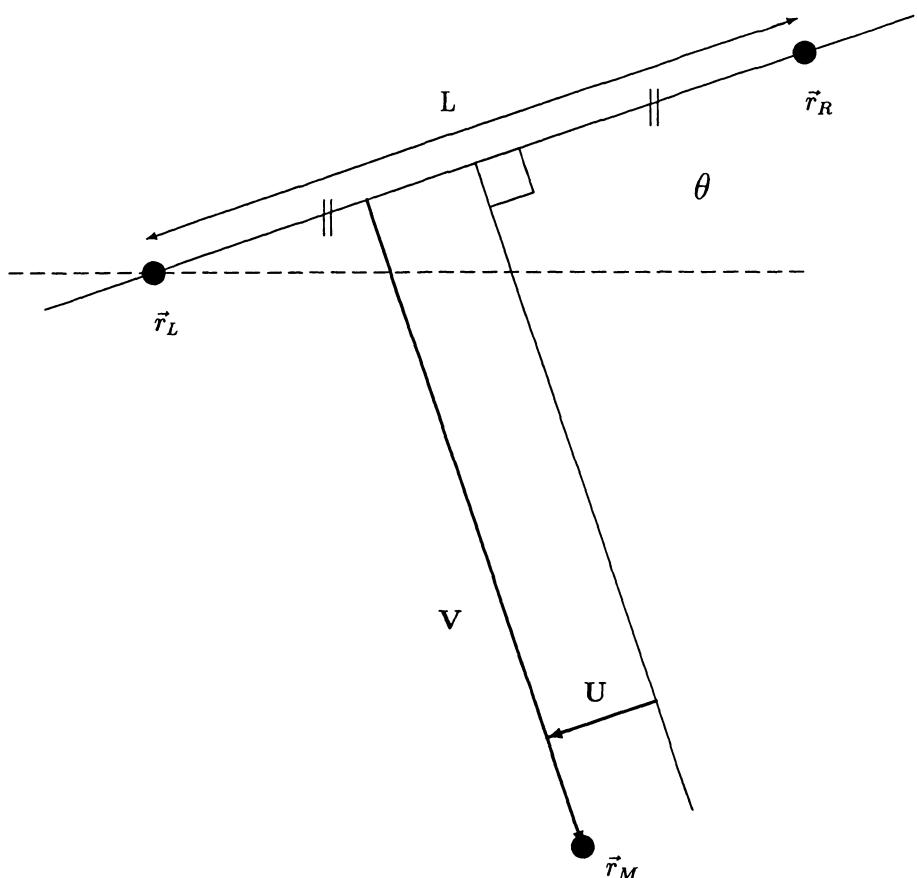


Figure 6 Notation for the Geometric Constraint Variables.

de-activated for that image. When pruning was not deactivated, the retained pixels in all cases included all the candidate feature points corresponding to correct location of the feature. The average reduction in the area of the search regions over each data set was over 40%.

A more robust version of the above spatial pruning method is now used and the modifications are described here. A fourth feature, the centre of the face, is also used. During training this is defined to be the centroid of the eye and mouth feature points. This feature is more reliably detected than the others and it can be used to impose greater geometric constraints during spatial pruning. Also, instead of thresholding each activation level independently, all four activation levels in each combination are considered together and it is their overall strength which is important. Their combined strength value is defined to be their product (or, in practise, the sum of their logarithms) and it is this value which is compared against a threshold. This reduces the chances of rejecting true feature points, even if their activation levels are slightly weak (and below threshold) without having to reduce the threshold level. The threshold level for the combined strength value is $F \cdot \log \theta$ where F is the number of features and θ is the threshold level specified for a single activation level. Thirdly, the representation of the geometric constraints is changed from a set of inequalities to a set of look-up tables. The position vector of a candidate right eye, with respect to the candidate left eye, is an integer pair. If this pair is valid then the maximum and minimum values of (U, V) for that pair are stored in look-up tables. This approach is more specific, and therefore more accurate, than the original approach. Fourthly, dynamic thresholding is used. If the spatial pruner fails to generate search regions then the decision is halved repeatedly until search regions are generated, unless the threshold goes down to an unacceptably small value. Finally, spatial pruning, is implemented by constructing a combinatorial tree during initialisation and scanning the tree during post-processing. This is a highly computationally efficient approach. For example, the tree is constructed so that its nodes only represent geometrically valid entries.

Spatial pruning is a rule-based operation. However, it has also been implemented as a Boltzmann machine [13].

Temporal Pruning

The occurrence of images containing no valid triples indicates the presence of one or more false rejections. The spatial pruning algorithm can therefore be used as a method of detecting false negatives. When false negatives are detected in one frame of a sequence, data from previous frames can be used to estimate the position of rejected feature points, and to allow some 'spatio-temporal' pruning.

The above scheme has not yet been implemented in the HPFL. Currently the only form of temporal pruning implemented is an inter-frame process that imposes motion constraints on candidate feature points. By comparing locations of candidate feature points in adjacent frames, false positives can be rejected. Candidate feature points are constrained to move by no more than one pixel position per frame in any direction. For a 25 frame/sec sequence this corresponds to about 1.5 image widths per second, which is not very restrictive.

Temporal pruning is applied after spatial pruning. Corresponding feature maps are compared for adjacent frames of a sequence. All candidate feature pixels in the current frame that are more than one pixel away from a candidate feature pixel in the previous frame are deleted. This has the effect of reducing the areas of the search regions by removing spurious false positives which occur randomly.

However, temporal pruning can only be used in off-line simulations, or if the frame-rate is high enough in a real-time application. In the experimental work reported in Section 4 this was not possible.

Pixel Expansion

In the final post-processing stage, pixel expansion, the remaining candidate feature points are expanded from single pixels to an array of 3x3 pixels. Although this actually increases the areas of the search regions, it is necessary because the generation of a coarsely sub-sampled image can give rise to sampling problems, where the desired feature in the high resolution image lies across the boundary of two or more low resolution pixels. In such cases a false negative may occur at the feature point, with a false positive in an adjacent pixel location. The false positive survives the spatial and temporal pruning processes, being close to the location of the feature point.

Pixel expansion ensures that the true feature point is brought within the search region, at the cost of increasing its area. However, in order to reduce this area, it is possible to apply additional spatial and temporal pruning. Results for the test set, after all stages of post-processing, are given in Table 3; these figures are for the same nets as Table 2.

Table 3 Performance after post-processing

Feature	Epochs	Size	Seed	Features retained/%	Avg. search region size/%
Left eye	50	2	0	100	3.65
Right eye	50	2	0	100	3.58
Mouth	50	2	0	96.7	4.11

2.4 Testing the low resolution stage

The low resolution stage of the HPFL has been tested on a number of sequences in off-line simulations on a workstation. Subjects in the sequences did not appear in the 60 still images used to train the feature detectors. In a typical example the number of candidate triples in a frame was reduced by spatial pruning from 56 to 12. Temporal pruning reduced the number still further, to 6. Test results are discussed further in [6].

3 FEATURE LOCATION IN THE HIGH RESOLUTION IMAGE

3.1 Micro-feature location

After pixel expansion, the feature maps are passed to the supervisor of the high resolution stage of HPFL (see Figure 1). Each pixel in a feature map corresponds to a 16x16 block in the high resolution image. The high resolution compound feature detectors are constrained to search for the centres within the region defined for that feature.

It has been found that high resolution MLP feature detectors generate spurious responses thereby degrading positional accuracy. To overcome this problem each feature in the image is considered to be composed of 'micro-features' [7], [9]. A typical set of manually measured micro-features is shown in Figure 7.

For each micro-feature an MLP micro-feature detector is trained and the combined outputs of the micro-feature detectors are post-processed in such a way as to increase the overall detection reliability. Five micro-features are used for each of the eyes and up to four micro-features for the mouth, each with an input image window of size 11-by-9 pixels. Micro-features are represented as points and a compound feature is a constellation of M micro-features surrounding a central datum point.

The centres of each of the micro-feature windows are scanned over the search region generated by the low resolution stage for the corresponding feature. The final positions for the micro-features are chosen to maximise the *a posteriori* probability (MAP) that the composite feature has been detected given the image data and prior knowledge about the structure of micro-features within a composite feature.

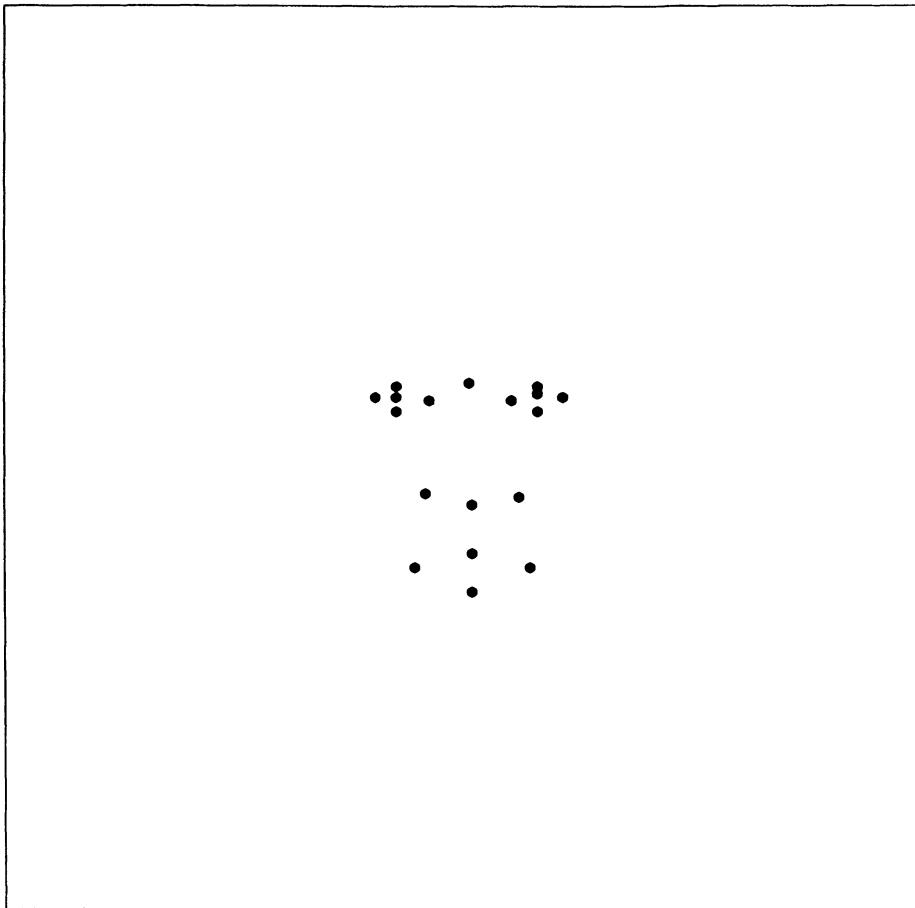


Figure 7 An example of a set of manually measured micro-features for eyes, nose and mouth inside a 256x256 image.

Let the search region for a datum point, \vec{r}_0 , be denoted by R_0 and let the search region for the m^{th} micro-feature be denoted by R_m which is obtained by a morphological operation on R_0 . In its simplest form the operation is a dilation. R_0 is referred to as a primary search region and R_m is a secondary search region.

3.2 MLP training

It has been shown that an appropriately trained MLP approximates an optimal Bayesian classifier [14], [15]. That is, if $a_m(\vec{r})$ is the MLP output for the m^{th} micro-feature at position \vec{r} , then $a_m(\vec{r})$ is an approximation to the posterior probability that the m^{th} micro-feature is located at position \vec{r} given the window of data centred at \vec{r} . We can thus use the MLP as a component in a probabilistic framework to locate the composite feature. The adopted training method, therefore, is to use pure backpropagation and to select pattern vectors from a natural mixture. For these MLPs, rather than needing to be on the right side of a decision threshold, it is necessary to generate probability values.

3.3 MAP processing

The Cartesian product of the search regions for micro-features belonging to a single compound feature defines a search space of feasible constellations:

$$(\vec{r}_0, \dots, \vec{r}_m, \dots, \vec{r}_M) \in \prod_{m=0}^M R_m$$

The problem is to choose *a posteriori* the most likely constellation. Two main sub-problems emerge: how to reliably calculate likelihood levels and how to efficiently search for the most likely constellation. Waite solved the first problem using a Bayesian model [7] and Vincent solved the latter by discovering a stopping criterion.

Given that an MLP has been trained to approximate a Bayesian classifier, we can thus use the MLP as a component in a probabilistic framework to locate the composite feature. One of the micro-features within the composite feature is chosen as a datum and the positions of the remaining micro-features are measured relative to this. Prior knowledge about the spatial configuration of micro-features within a composite feature is obtained by modelling the relative position of each micro-feature, with respect to the datum, by a Gaussian probability distribution. The mean and variance of the distribution were obtained by measurements from a sample of images. If we let the probability distribution of the position, \vec{r}_m , of the m^{th} micro-feature relative to the datum position, \vec{r}_0 , be g_m , then it may be shown that the configuration of micro-feature positions which maximises the probability of detecting a composite feature is that which maximises:

$$z(\vec{r}_0, \dots, \vec{r}_m, \dots, \vec{r}_M) = a_0(\vec{r}_0) \prod_{m=1}^M a_m[\vec{r}_m] g_m(\vec{r}_m - \vec{r}_0)$$

It should be emphasised that the above is derived from a model which makes a number of simplifying assumptions [7] and other formulations for fusing the neural outputs are conceivable.

Because of the assumed independence between relative micro-feature positions, the maximisation is an $O(N^2)$ process, where N is the number of positions within an MLP search region. This considerably reduces computational complexity. There exists a maximum for each pair, (\vec{r}_0, m) :

$$z_{max}(\vec{r}_0, m) = \max_{\vec{r}_m \in R_m} a_m[\vec{r}_m] g_m(\vec{r}_m - \vec{r}_0)$$

and these maxima are used to find an overall maximum:-

$$z_{max} = \max_{\vec{r}_0 \in R_0} a_0(\vec{r}_0) \prod_{m=1}^M z_{max}(\vec{r}_0, m)$$

Computational cost is a function of the sizes of the search regions:-

$$C = \|R_0\| \sum_{m=1}^M \|R_m\|$$

However, this is still a brute force approach and very expensive. Considerable reduction, without sacrificing exhaustiveness, is achieved by ordering the search in a particular way. This is explained as follows. Define $\vec{\Delta r}_m$ to be the position vector of the m^{th} micro-feature relative to a datum point, \vec{r}_0 ,

$$\vec{\Delta r}_m = \vec{r}_m - \vec{r}_0$$

Then define a scanning pattern for the m^{th} micro-feature as a sequence of relative vectors, such that, when added to any candidate datum point, the vectors fill the m^{th} search region.

$$S_m = (\vec{\Delta r}_m[1], \dots, \vec{\Delta r}_m[j], \dots)$$

Let the ordering be such that the Gaussian function decreases monotonically along the scanning path.

$$g_m(\vec{\Delta r}_m[i]) \geq g_m(\vec{\Delta r}_m[j]), \forall i < j$$

The maximum for the m^{th} micro-feature given a candidate datum point, \vec{r}_0 , is up-dated by scanning along the path.

$$z_{max}(j, \vec{r}_0, m) = \max_{i \leq j} a_m(\vec{r}_0 + \vec{\Delta r}_m[i]) g_m(\vec{\Delta r}_m[i])$$

The monotonic ordering leads to a stopping criterion. This is proved below. Define A_m to be the maximum activation level in the m^{th} neural map.

$$A_m = \max_{\vec{r}_m \in R_m} a_m(\vec{r}_m)$$

Given A_m it is possible to calculate an upper bound for the maximum value along the rest of the scanning path.

$$\max_{k > j} a_m(\vec{r}_0 + \vec{\Delta r}_m[k]) g_m(\vec{\Delta r}_m[k]) \leq A_m g_m(\vec{\Delta r}_m[j + 1])$$

If this upper bound is no greater than the current maximum then it is pointless to continue scanning.

$$z_{max}(j, \vec{r}_0, m) \geq A_m g_m(\vec{\Delta r}_m[j + 1]) \Rightarrow z_{max}(j, \vec{r}_0, m) = z_{max}(\vec{r}_0, m)$$

A gaussian is a unimodal convex function. Therefore its scanning pattern forms a spiral path and so this technique is called spiral scanning. Experiments showed that spiral scanning speeded up the search typically by a factor of around 300 making it insignificant in comparison with the neural processing.

3.4 Testing the High Resolution Stage

In simulations it was found that, after training on sixteen images, MAP likelihood feature constellation location gave a mean error of 0.73 pixel widths and a maximum error of 2 pixel widths for a test set of 44 faces. This is a significant improvement over simply choosing the maximum MLP outputs. Further details have been reported previously [7], [9]. It should be noted however, that these results are meaningful only in the range of scales and orientations defined implicitly by the training set of images.

4 EXPERIMENTS WITH FRESHLY GRABBED SEQUENCES

A live demonstration of HPFL is being developed on a pipeline image processing system from Datacube, Inc. A person sits in front of a camera and the resultant search regions and estimated micro-feature points are displayed superimposed on the image. To-date the demonstration processes frames at a slow rate but future versions will tend towards 25Hz.

Figures 8-16 show examples of feature location. Subjects were illuminated by a bright and uniform light source from the front. Under this type of lighting condition the search generator performs well because the training set was similarly lit. Under less well controlled conditions performance degraded. It was therefore necessary to create a much larger and richer database and to re-train the MLPs in order to achieve the desired robustness. However, discussion of this very recent phase of work is beyond the scope of this chapter.

The high resolution stage was inherently more robust to variations in lighting conditions. However, in the real-world tests, we can claim only partial success in pin-pointing micro-features as indicated by the figures. This is due to sensitivity to scale and rotation as well as poor modelling of micro-feature density by the gaussian approximation. These problems are currently being investigated.

5 CONCLUSIONS

The Hierarchical Perceptron Feature Locator described in this paper is a method of locating features in human head-and-shoulders image sequences that has been tested in the real world. The keys to its robustness are multi-resolution processing, specialised neural training algorithms and sophisticated post-processing of neural outputs.

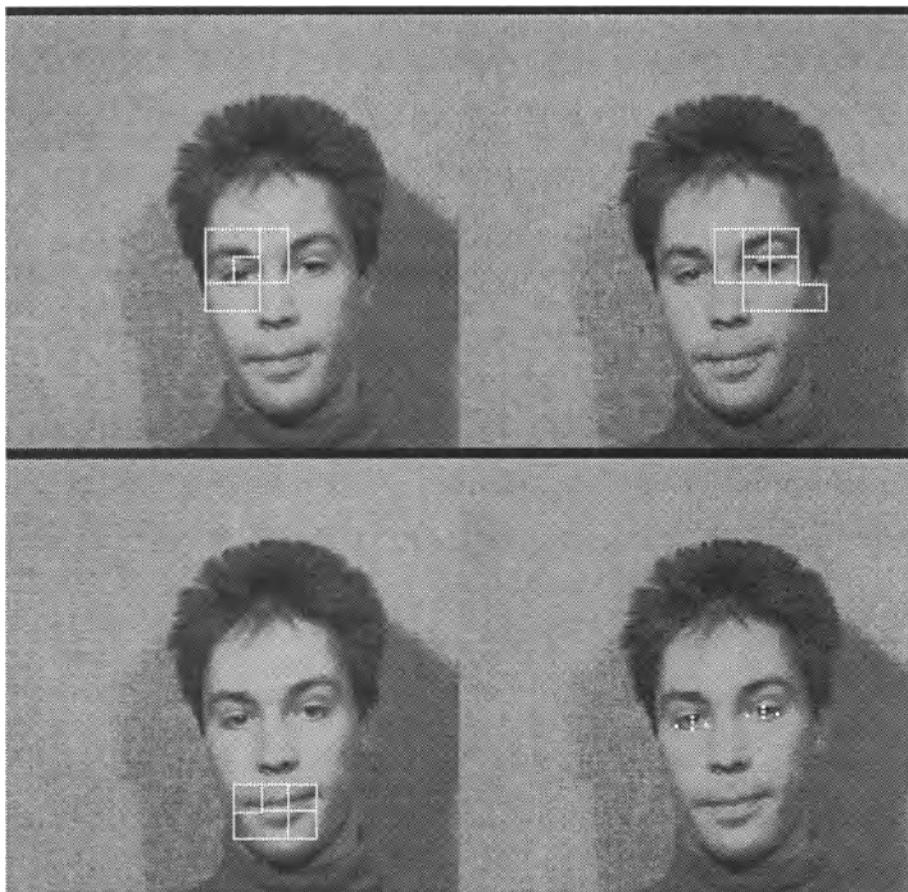


Figure 8 Example of search region generation and micro-feature location. (Example 1)



Figure 9 Example of search region generation and micro-feature location. (Example 2)

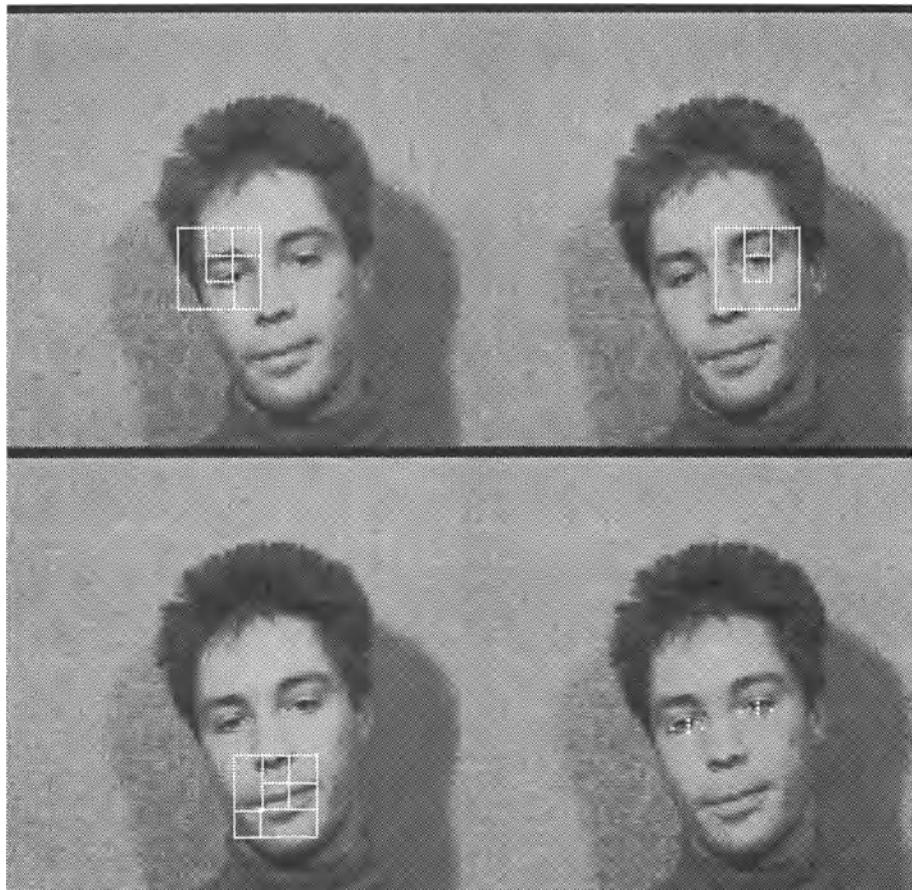


Figure 10 Example of search region generation and micro-feature location. (Example 3)
Positional accuracy is reduced because the face is tilted.

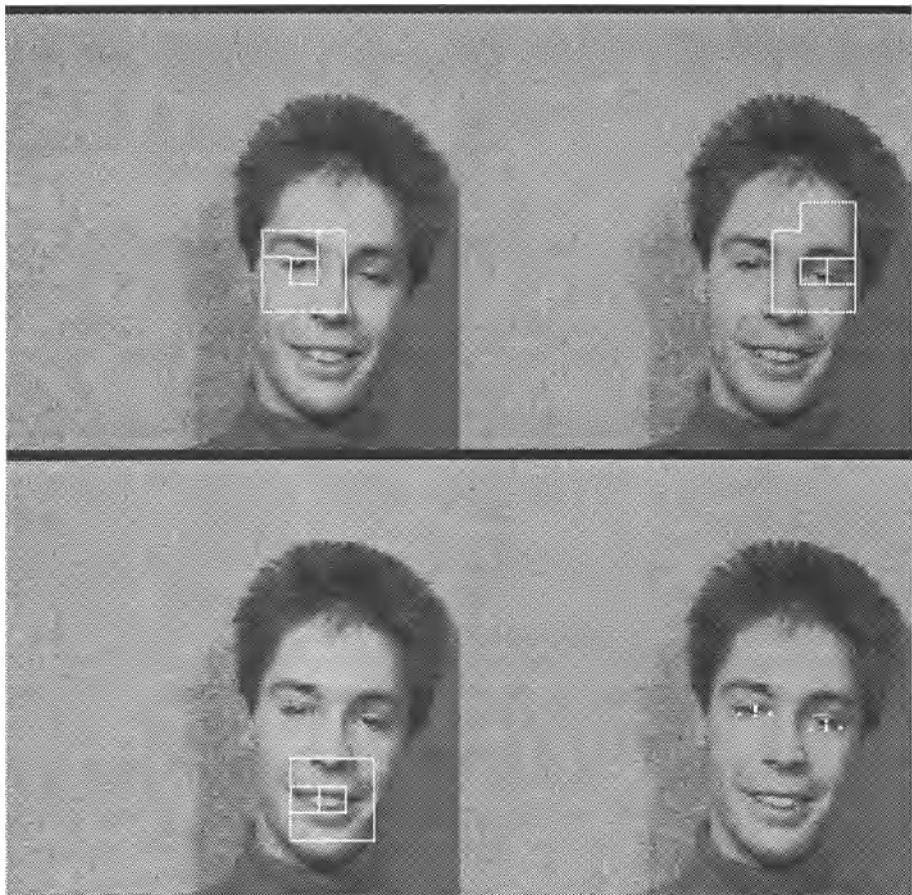


Figure 11 Example of search region generation and micro-feature location. (Example 4)
Positional accuracy is reduced because the face is tilted.

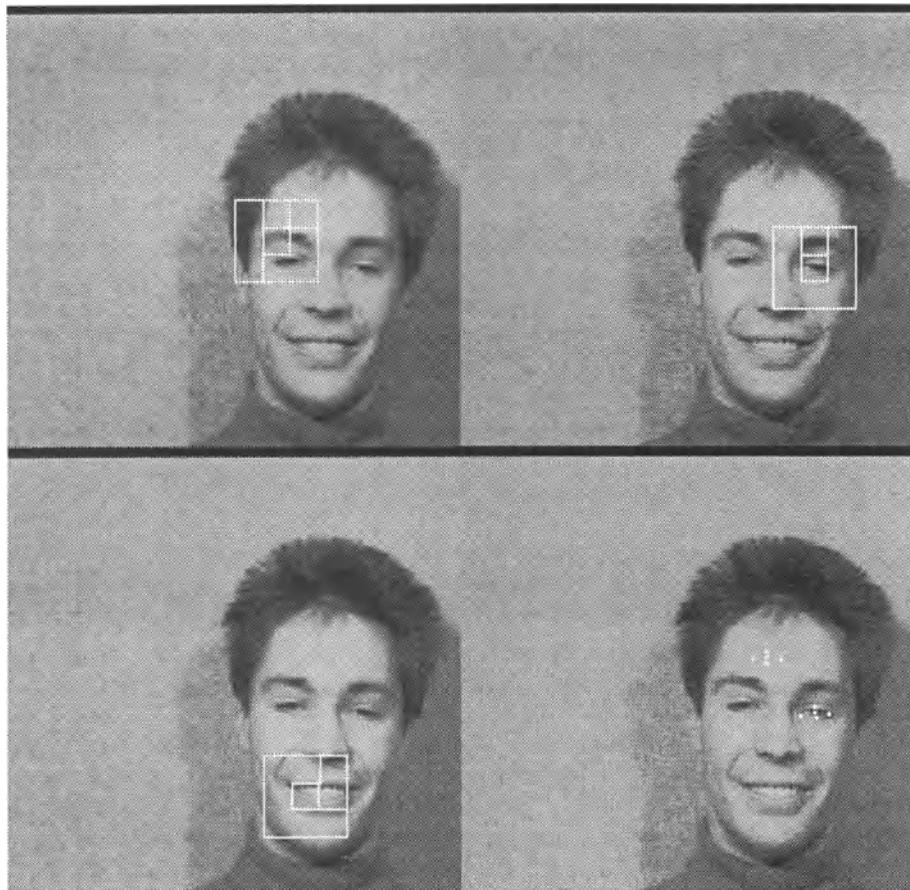


Figure 12 Example of search region generation and micro-feature location. (Example 5) The left-eye detector has locked on to some spurious activation levels because the High-Resolution Stage does not know about global facial structure and possibly because the eyes are nearly closed.

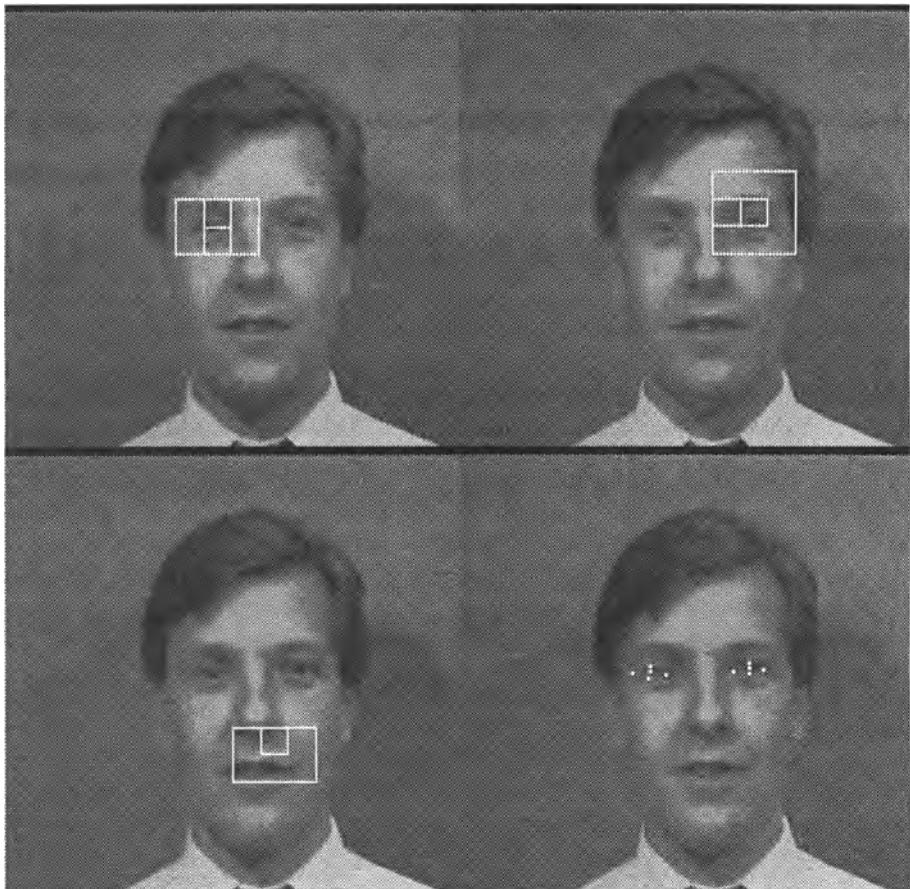


Figure 13 Example of search region generation and micro-feature location. (Example 6)

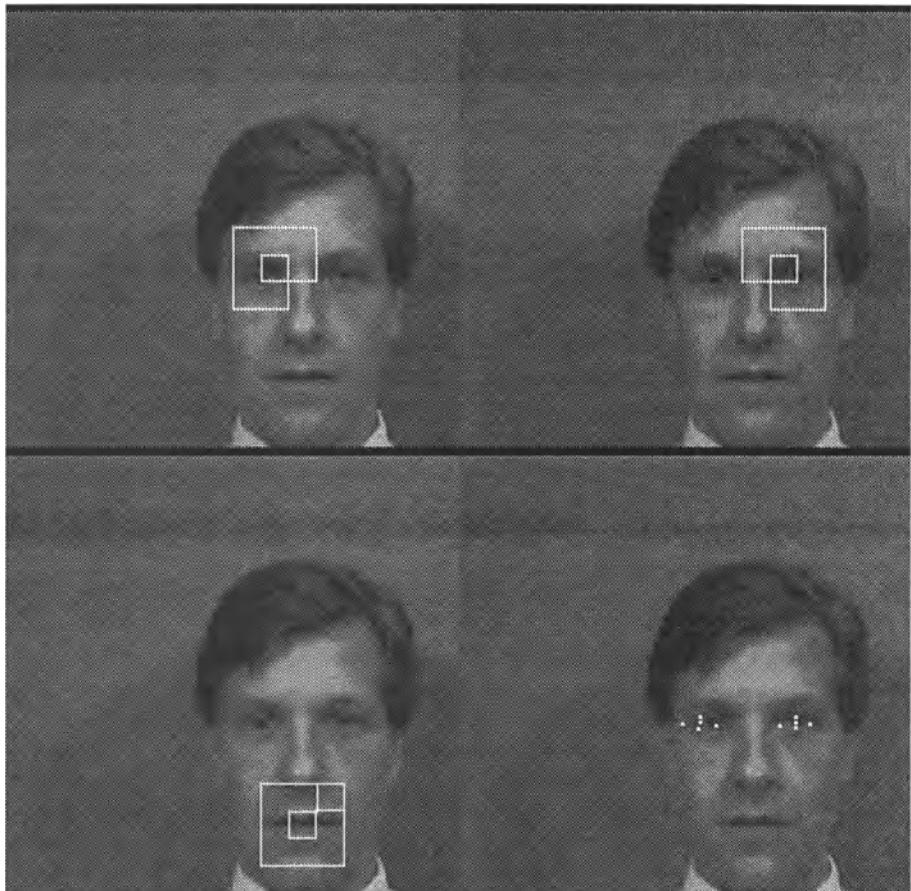


Figure 14 Example of search region generation and micro-feature location. (Example 7)

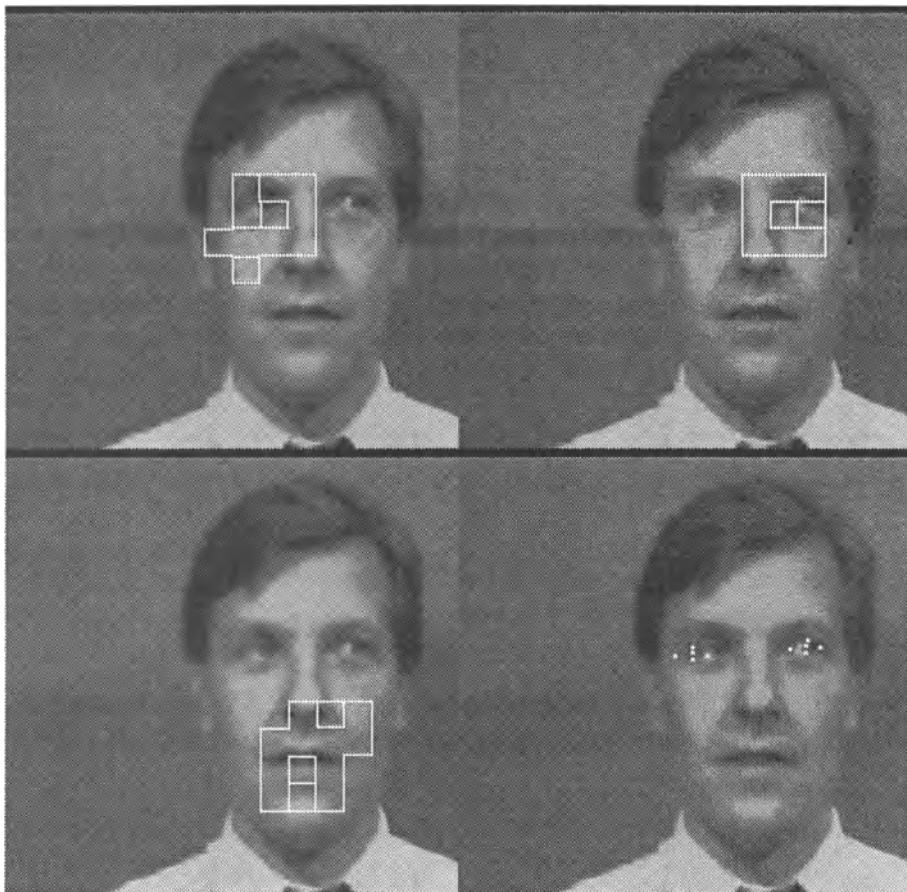


Figure 15 Example of search region generation and micro-feature location. (Example 8)
The left-eye has been badly located and this may be a result of eyeball swivel.

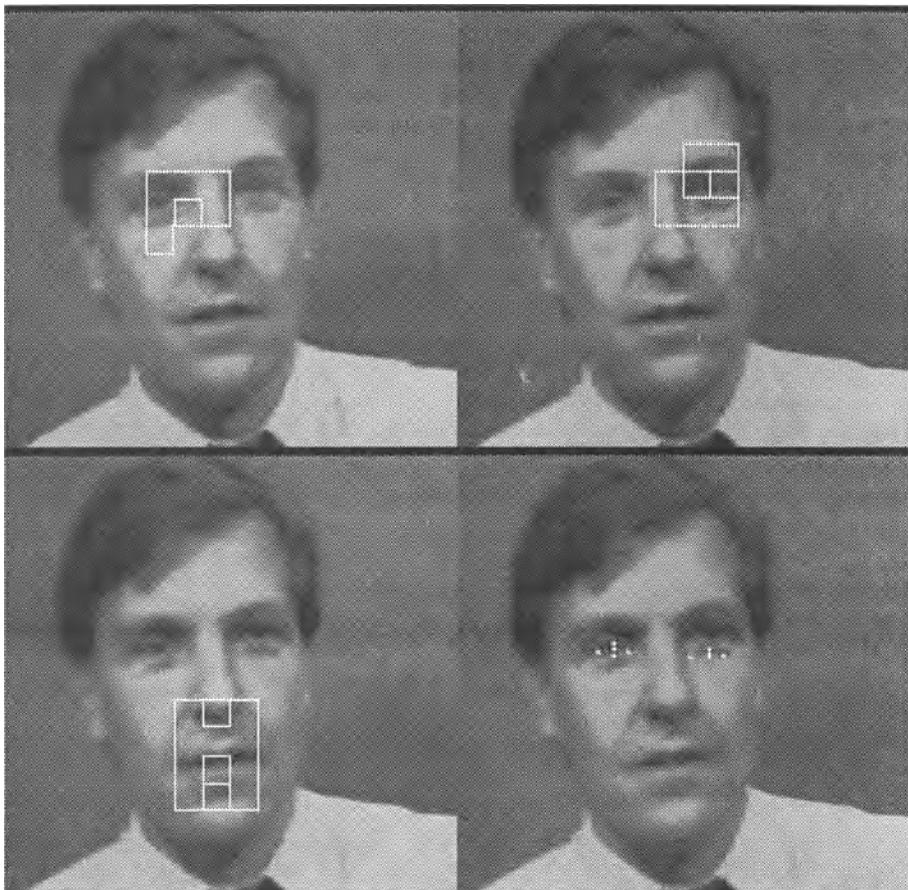


Figure 16 Example of search region generation and micro-feature location. (Example 9)
The eyes have been badly located and this is partly because the High Resolution Stage has not been trained to cope with such a large scale.

There are a number of ways in which the system could be, and is being, improved. For example, at present the High Resolution Stage deals with a narrow range of scales and rotations, and assumes that there is only one head-and-shoulders in the image. Possible extensions include allowing the system to operate over a range of scales (possibly by using a multi-resolution pyramid), and allowing images that have a number of subjects in them. Furthermore, the High Resolution Stage does not yet know about global facial structure and so it cannot check that the chosen constellations are mutually consistent geometrically. This could fairly easily be achieved.

The ultimate aim of the work is to produce a real-time demonstrator system, based on a VLSI implementation of Multi-Layer Perceptrons [16] and to apply it to problems of visual telecommunications.

6 ACKNOWLEDGEMENTS

A number of people have contributed to the work described here. I would like to acknowledge the contribution of Robert Hutchinson and Bill Welsh, who performed some of the early work which lead to the development of the HPFL system, and Jon Waite who devised and tested micro-features. Summer students Adam Buckley and Christoff Pomper wrote much of the software for implementing the high resolution stage, including the tricky spiral scanner. Charley Nightingale, Mike Whybray, David Myers and Paulo Lisboa (University of Liverpool) are appreciated for their valuable comments and advice.

REFERENCES

- [1] Nightingale C, Hutchinson R A, "Artificial neural nets and their application to image processing", British Telecom Technology Journal, Vol. 8, No. 3, pp. 81-93, July 1990.
- [2] Hines E, and Hutchinson R A, "Application of Multi-layer Perceptrons to Facial Feature Location", Proc. 3rd IEE Int. Conf. on Image Processing and its Applications, Warwick UK, July 1989.
- [3] Hutchinson R A and Welsh W J, "Comparison of Neural Networks and Conventional Techniques for Feature Location in Facial Images", Proc. 1st IEE Int. Conf. on Artificial Neural Networks, London, October 1989.

- [4] Vincent J M, "Facial Feature Location in Coarse Resolution Images by Multi-Layered Perceptrons", Proc. Int. Conf. on Artificial Neural Networks (ICANN-91), Vol. I, pp. 821-826, 24-28 June 1991, Elsevier Science.
- [5] Vincent J M, Myers D J, Hutchinson R A, "Image feature location in multi-resolution images using a hierarchy of multi-layered perceptrons", in, Lingard, Myers and Nightingale (Eds): 'Neural networks for images, speech and natural language', Chapman Hall, UK, 1992.
- [6] Vincent J M, Waite J B, Myers D J, "Location of feature points in images using neural networks", BT Technology Journal, Vol 10, No 3, July 1992.
- [7] Waite J B, "Facial Feature Location Using Multilayer Perceptrons and Micro-features", Proc. IEEE Int. Jnt. Conf. on Neural Networks (IJCNN-91), Vol. 1, pp. 292-299, 18-21 Nov. 1991.
- [8] Vincent J M, Waite J B, Myers D J, "Precise location of facial features by a hierarchical assembly of neural nets", Proc. IEE 2nd Int. Conf. on Artificial Neural Networks, ANN-91, 18-20 November 1991, pp. 69-73.
- [9] Vincent J M, Waite J B, Myers D J, "Automatic location of visual features by a system of multilayered perceptrons", IEE Proceedings-F, Vol. 139, No. 6, pp. 405-12, December 92.
- [10] Maxwell T et Al, "Transformation Invariance Using High Order Correlations in Neural Net Architectures" IEEE Int. Conf. on Systems Man and Cybernetics, Atlanta, Georgia. October 1986.
- [11] Rumelhardt D E, and McClelland J L, Parallel Distributed Processing; Volume 1: Foundations", The MIT Press 1987.
- [12] Allred L G, Kelly G E, "Supervised learning techniques for backpropagation networks", Proc. IJCNN, 1990, Vol. 1, pp. 721-8.
- [13] Popescu V, Vincent J M, "Location of facial features using a Boltzmann machine to implement geometrical constraints", Chapter 14 in 'Techniques and applications of neural networks', Taylor M, Lisboa P (Eds), Ellis Horwood Workshop Series, 1993, Ellis Horwood.
- [14] Wan E A, "Neural Network Classification: A Bayesian Interpretation", IEEE Trans. Neural Networks, Vol. 1, No. 4, pp. 303-305, Dec. 1990.
- [15] Ruck D W, et al, "The Multilayer Perceptron as an Approximation to a Bayes Optimum Discriminant Function", IEEE Trans. Neural Networks, Vol. 1, No. 4, pp. 296-298, Dec. 1990.

- [16] Orrey D A, Myers D J, Vincent J M, "A High Performance Digital Processor for Implementing Large Artificial Neural Networks", Proc. Custom Integrated Circuit Conference CICC-91, San Diego, Ca., 12-15 May 1991.

Sex Recognition from Faces Using Neural Networks

B. Golomb, T. Sejnowski

*Howard Hughes
Medical Institute
The Salk Institute
10010 North Torrey Pines Road
La Jolla, CA 92037*

1 INTRODUCTION

Recognizing the sex of conspecifics is important: Expending one's attentions Q or one's seed Q on the wrong parity of partner could put one at a competitive disadvantage. While some animals use pheromones to recognize sex, in humans this task is primarily visual: "Many socially living animals ... recognize each other as members of the same species, as individuals, and as social partners by means of visual signals and communicate their mood and intentions by the same sensory modality. In many primate species the individual structure of the face is the most important visual characteristic of each group member" (Grusser, Selke, & Zynda 1985). A core issue is how sex is recognized from face; yet until recently this received little attention.

Many factors make the face a focus of interest in humans, and thus a good locus for sexually dimorphic features. It is the subject of scrutiny in communication: Facial gestures, phylogenetic precursors to verbal communication (Grusser, et al. 1985), supplement verbal language cues; lip reading augments audial cues of language (and may override them, as in the McGurk effect (McGurk, & MacDonald 1976). The human face is visually accessible since it is less clothed for protection and modesty than other bodily parts, to permit vision and breathing. Autonomic and physical signs appear on the face (such as pupillary constriction/dilation, or flared nostrils with labored breathing or

anger), and can indicate the individual's arousal and potential for threat. Too, emotions are played out in facial expressions: These differ from expressions used in language, and may dictate whether approach or avoidance is the best course. (Ekman and Friesen have identified six "universal" expressions of emotion which occur cross-culturally (Ekman 1973a; Ekman 1973b; Ekman 1977; Ekman 1989; Ekman, & Friesen 1969; Ekman, Friesen, & Simons 1985) and have distinctive autonomic nervous system signatures (Ekman, Levenson, & Friesen 1983; Schwartz, Weinberger, & Singer 1981; Sternbach 1962), making them effective indicators of the bearer's state; several of these, such as anger and fear, have decided significance to the onlooker.)

In short, we have reason to look at the face; and we have need of a visual indication of sex; so it is sensible that some visual cues of sex should abide on the face.

Primate physiologists have devoted much attention to faces. Neurons that respond selectively to faces have been identified in the amygdala (Rolls 1981; Rolls 1984; Sanghera, Rolls, & Roper-Hall 1979); the anterior inferior temporal cortex (Gross, Rocha-Miranda, & Bender 1972; Rolls, Judge, & Sanghera 1977); a polysensory area of the dorsal (anterior) bank of the superior temporal sulcus (STS) (Bruce, Desimone, & Gross 1981), where response latencies are 200-300 msec after presentation of the face; in the fundus and anterior portion of the superior temporal sulcus, where response latencies are 70-150 msec (Baylis, Rolls, & Leonard 1985; Perrett, Rolls, & Caan 1982; Perrett, et al. 1984); in parietal cortex (Leinonen, & Nyman 1979); and in frontal cortex (Pigarev, Rizzolatti, & Scandolara 1979). The superior temporal sulcus work is particularly interesting, as up to 20% (Bruce, et al. 1981; Perrett, et al. 1982), though as low as 3% (Perrett, et al. 1984) of neurons tested are accounted "face cells" in that their response to the best tried face was at least twice (and often more than ten times) as great than the greatest response to any nonface stimulus tested (Perrett, et al. 1982) (including gratings and random 3-D objects for instance). Some such neurons appear to identify specific faces independent of expression (Perrett, et al. 1982), while others respond to features or expression independent of face (Perrett, et al. 1984).

There is great variability in how selective these cells are to a specific face: Some are active for most faces, and others highly selective for but one or a few (Perrett, et al. 1982; Rolls 1992). Consequently it would not be implausible for some STS cells to respond preferentially according to sex of faces; or this could occur in the amygdala, where damage leads to disruption

of emotional and social responses to faces (Rolls 1984; Young, & Yamane 1992). No attempt has been made to test for face cells selective for the sex of a primate (human or monkey), perhaps due to lack of conviction that facial sex differences exist in rhesus monkeys.

Monkeys with bilateral temporal lobe lesions (affecting, for instance, the STS) may develop "Kluver-Bucy" syndrome (Horel, Keating, & Misantone 1972; Jones, & Mishkin 1972; Kluver, & Bucy 1939; Trezean, Dalle, & Ore 1955), which includes inappropriate reaction to faces, and sexual indiscriminanza. However, this is not likely to ensue from selective inability to recognize sex, since these monkeys' pathological sexual eclecticism extends not only to individuals of the wrong sex, but of the wrong species Q and kingdom.

There may be a human analog of Kluver-Bucy (Trezean, et al. 1955); however the best-described human lapse in responding to faces is "prospagnosia," or inability to recognize faces (Benton, & van Allen 1972; Bodamer 1947; Bornstein 1963; Bornstein, Sroka, & Munitz 1969; Christen, Landis, & Regard 1985; Damasio, Damasio, & van Hoesen 1982; Gloning, Gloning, Jellinger, & Quatember 1970; Hecaen, & Angelergues 1962; Meadows 1974; Whiteley, & Warrington 1977). This attends bilateral mesial occipitotemporal lesions (lesions of the lingual and fusiform gyri) (Damasio, et al. 1982; Jeeves 1984), often from vascular lesions of the posterior cerebral artery. The affected neurons may be the analog of the "face cells" studied in monkeys. The afflicted are able to tell there is a face, but cannot identify whose it is. Sex recognition is typically preserved (Tranel, Damasio, & Damasio 1988).

The lesion technique provides evidence that a region of the brain may be involved in some function, but cannot by itself tell us what that function is. New techniques for functionally mapping the activity of normal brains have recently been developed. One of these methods, positron-emission tomography (PET), has been used to identify brain regions that are involved in face processing and, in particular, to localize brain areas that are active during recognition of sex from faces (Sergent, Ohta, & MacDonald 1992).

Several mechanisms may explain preserved ability to discern sex, in the "face" of lost ability to identify the bearer of the face, in humans. Different cortical areas may subserve recognition of sex versus facial identity; indeed, more rostral areas of temporal cortex, in addition to subcalcarine cortex, are

believed necessary for identity but not sex recognition in humans; damage to these rostral areas would engender loss of identity recognition, with preservation of sex recognition. Loss of ability to discriminate sex appears to require damage to both left and right subcalcarine "early" visual association cortices (Tranel, et al. 1988).

Preserved ability to infer sex from ancillary features may help: Characteristic hairstyles, clothing, jewelry or makeup are associated with males or females, but seldom uniquely identify the bearer. In addition, sex discrimination (but not identity discrimination) is vastly overlearned, in legion contexts, which could promote a robust neuronal representation less sensitive to loss of a fraction of involved neurons. Each face to which we are exposed represents a "training instance" of either maleness or femaleness, with associated features of voice, mannerisms, clothing and body habitus to disambiguate the answer (provide a "teacher"); yet we are exposed to comparatively few instances of any individual, and don't always have a secondary tag to tell us if we are right or wrong about who it is, supposing we have any idea at all.

Quite young infants are able to tell males from females (by whatever constellation of cues, probably including pitch of voice), allowing new encounters to serve as training examples in which novel faces are linked to the appropriate sex. Thus the process of sex recognition training may commence quite early. Finally, computer analysis has revealed that the features which allow sex discrimination may occur in the first few principal components of the faces, involving lower spatial frequency cues, while those for identity recognition reside in later principal components (O'Toole, Abci, Deffenbacher, & Valentin 1993).

In any event, neurologically intact humans do recognize sex from face. But by and large they are unable to say how. Although certain features are nearly pathognomonic for one sex or the other (facial hair for men, makeup or certain hairstyles for women), even in the absence of these cues the determination is made; and even in their presence, other cues may override.

Sex-recognition in faces is thus a prototypical pattern recognition task of the sort at which humans traditionally excel, and by which knowledge-based Artificial Intelligence has traditionally been vexed; in short, an ideal application to demonstrate the capabilities of neural networks. It appears to follow no simple algorithm, and indeed is modifiable according to fashion (makeup, hair etc). While ambiguous cases exist, for which we must appeal to other cues such as physical build (if visible), voice patterns (if audible), and mannerisms, humans are fairly good in most cases at discriminating sex

merely from photos of faces, without resorting to such adscititious cues. The obvious question is, can neural networks do the same? We have developed a neural network system called SexNet that can classify the faces of college-aged students with 89% accuracy. Humans achieved a classification performance of 88% on the same set of faces. Our performance is higher than that reported from other neural network systems that start with feature extraction (Golomb, Lawrence, & Sejnowski 1991). Moreover, even a perceptron, with one layer of variable weights, achieves a respectable accuracy (Gray, Lawrence, Golomb, & Sejnowski 1993). These results demonstrate that good performance at face classification can be achieved with relatively simple preprocessing of images. In this chapter we will summarize these results.

2 METHODS

We started with 512x512 face images of 90 young adult faces (45 male, 45 female) (See Fig. 1) (O'Toole, Millward, & Anderson 1988). Faces had no facial hair, no jewelry, and apparently no makeup. A white cloth was draped about each neck to eliminate possible clothing cues. Most images were head on, but exact angle varied.

Each face image was rotated until eyes were level; scaled and translated to position eyes and mouth similarly in each image; and clipped to present a similar extent of image around eyes and mouth. (This clipping eliminated hair cues on many faces, though residual hair cues remained in some.) Final faces were 30x30 pixels with 12 pixels between the eyes, and 8 pixels from eyes to mouth. The 256 gray-level images were adjusted to the same average brightness. (No attempt was made to equalize higher order statistics.) This served as input to the networks, as shown in Figure 1.

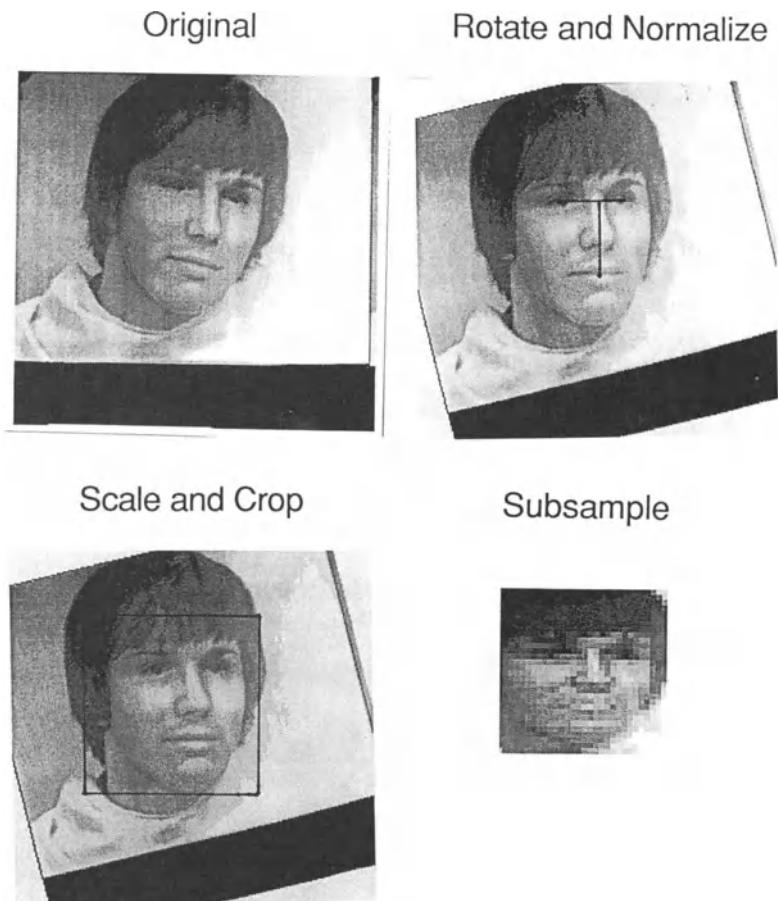


Figure 1

Figure 1: Preprocessing for faces. The original image is shown in the upper left. The center of each eye was located by hand and the line joining the eyes was rotated to the horizontal. The distance between the eyes and the perpendicular distance between the eyes and mouth were normalized (upper right). The image was scaled and cropped (lower left) and subsampled randomly within each subregion to produce a 30x30 image (lower right). The average gray level of the pixels was adjusted to have the same value for all images.

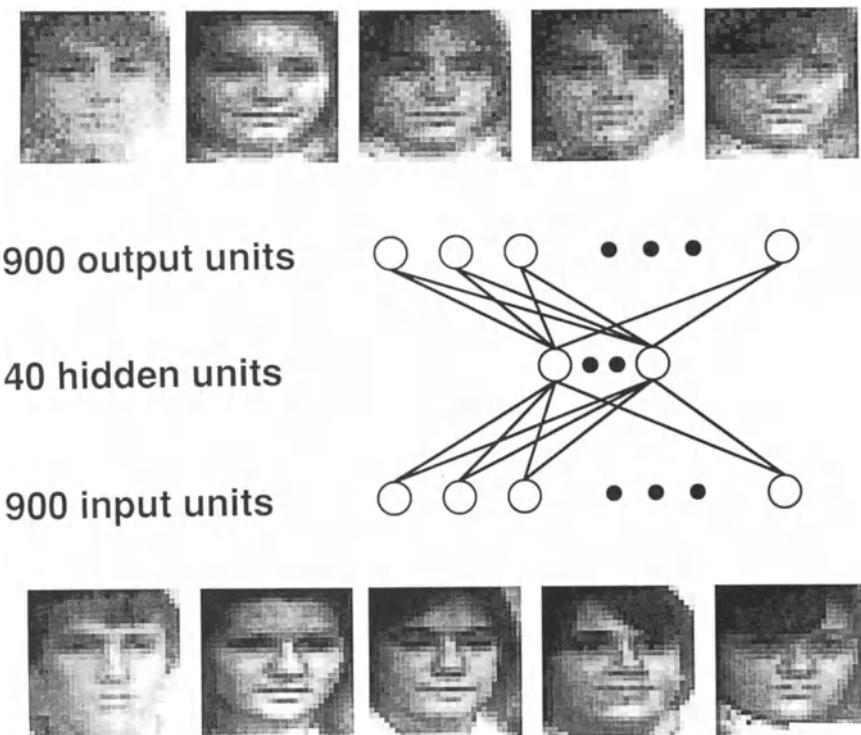


Figure 2

Figure 2: Compression network. The preprocessed images were compressed with a $900 \times 40 \times 900$ encoder to reduce the dimensionality of the input space to the SexNet. Sample input images are shown on the bottom and their corresponding outputs from a trained compression network are shown above. In some networks this compression stage was bypassed and the gray level image was used directly as input to the sex identification network.

Network processing consisted of two phases: image compression and sex discrimination. Both networks were fully connected three layer networks with two biases, trained with conventional backpropagation (Parker 1986; Rumelhart, Hinton, & Williams 1986; Werbos 1974), with a bias, a sigmoidal squashing function, and a learning rate of 0.2, using the Neuristique SN2 simulator written by L. Bottou and Y. LeCun.

Image compression followed the scheme of Cottrell, Munro and Zipser (Cottrell, Munro, & Zipser 1987), in which an input to a back-propagation network is reproduced as output, after being forced through a hidden unit bottleneck (i.e. a smaller number of units), as shown in Figure 2. This process, if successful, gives a new reduced representation of the input (here, the face image) in the activity of the (fewer) hidden units. If the network is able to reproduce the input as output from the information in the hidden units, the new reduced representation in the hidden units must contain the salient information of the input (faces). This scheme, though it technically involves supervised learning, can be viewed as unsupervised since no "teacher", other than the input itself, is required to produce the error signal (Cottrell, & Fleming 1990).

The new reduced representation of each input (i.e. the activities of the hidden units of the compression network for that face) can be used as a substitute, more parsimonious input to a second stage network which derives some secondary information from the input (in our case, the sex of the face), as illustrated in Figure 3. The application of this two-stage network technique to faces was pioneered by Cottrell and Fleming, who used it for face recognition, with a more limited effort at sex recognition (producing 37% errors) (Cottrell, et al. 1990).

In our work, the compression net served to force the 900 unit (30x30) images (900 inputs) through a 40 hidden unit bottleneck, and reconstruct the image at the 900 unit output level. Thus, the input equalled the desired output. The network trained for 2000 runs on each of 90 faces, yielding output faces which were subjectively distinct and discriminable, although not identical to the inputs. This procedure served to forge a representation of each face in the activities of only 40 units, and thus provide a more tractable input (40 units rather than 900) to the sex discrimination network Q a boon for a small training set. The second, sex-discrimination network, had 40 inputs (the activities of the 40 hidden units of the compression net), 0 or 40 hidden units, and one output unit. Training consisted of encouraging, by gradient descent (Rumelhart, et al. 1986); the network to produce a "1" for men, and a "0" for

women. (These numbers are standard network integers, and any similarity to discriminating male and female anatomical parts is purely coincidental.) Values greater than 0.5 were accounted "male", and those less than 0.5 female. Trials were also done with no antecedent compression, and with no hidden units in the SexNet (i.e. feeding the 900 unit input directly to the one unit output).

As the suitable benchmark for sex discrimination by the network is human performance on the same faces, human testing was undertaken. Sex order was chosen to be random for 45 faces (even vs odd sequential digits of pi coded male vs female), and, to equalize the numbers of males and females, the order was repeated with reverse parity for the second 45 faces. 5 humans were tested on the 90 faces, and made two binary decisions for each face: sex and certainty of their answer (sure vs unsure). They had unlimited time, and could scrutinize faces in any manner. The full 512x512 resolution images were used, which included hair but no clothing (see Fig. 1).

For comparison, 9 tests of the SexNet were undertaken, each training on a different 80 faces, leaving a distinct set of 10 untrained faces for testing. We performed tests on two sets of images. The first set included some hair (See Fig 2) but the second set was cropped so that only the central portion of the face was visible.

Sex Network

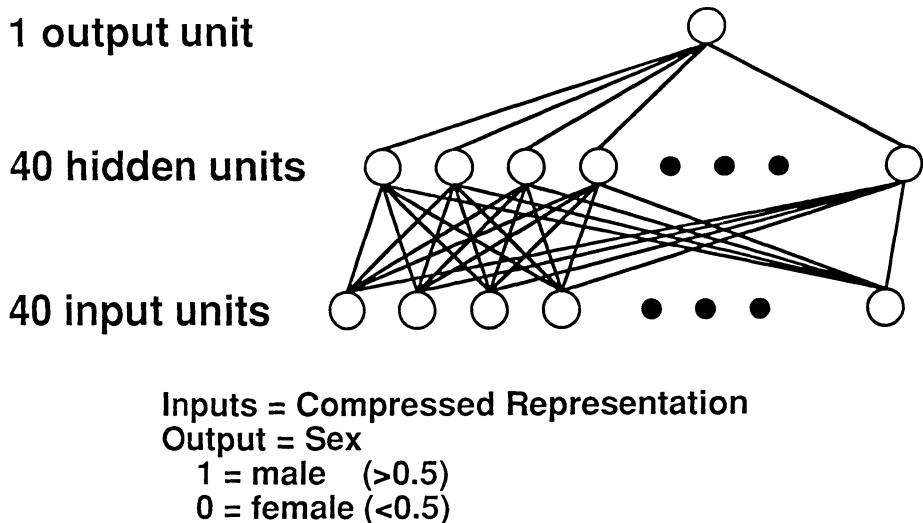


Figure 3: Network for identifying sex from compressed images of faces. the 40 hidden units from a compression network (see Fig. 2) are used as input to a $40 \times 40 \times 1$ identification network. The output was threshholded to classify each image as male (output > 0.5) or female (output < 0.5).

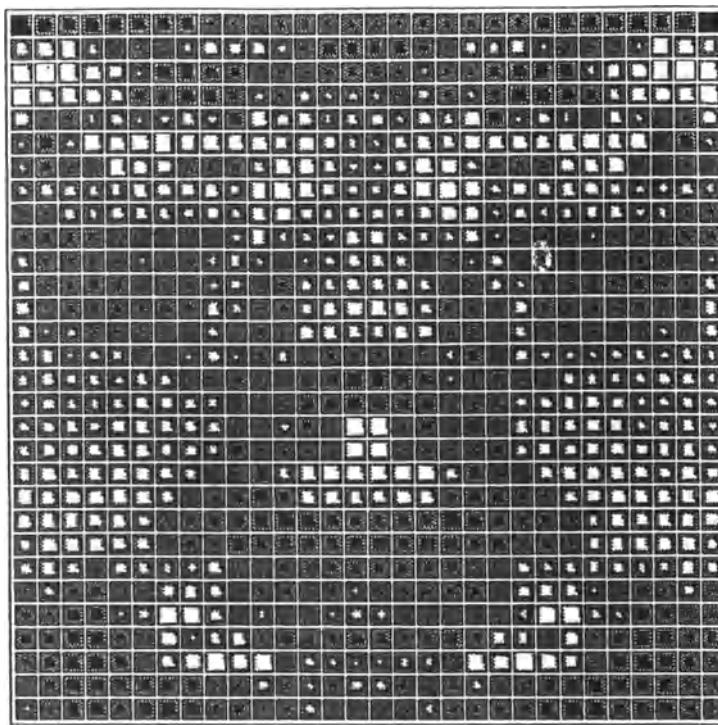


Figure 4: Weights in a perceptron network that was trained to discriminate sex from images of faces. Only the central region of the face was used in training, which excluded information from the outline of the face and hairline. The area of each square in the diagram is proportional to the value of the weight in the network assigned to that region of the face. White weights are excitatory and black weights are inhibitory. A high intensity in a region of the face with a white weight provides evidence for male, and a high intensity region with black weights is evidence for female (Adapted from (Gray, et al. 1993)).

3 RESULTS

Human Performance.

Psychophysical studies of 5 humans on the 90 faces revealed errors on 8, 10, 12, 8, and 14 faces, corresponding to 8.9%, 11.1%, 13.3%, 8.9% and 15.5%, for an average error rate of 11.6%. Humans and the network seemed prone to similar errors. One male face gave particular trouble to the SexNet, being wrongly assigned when a test face, and taking long to train when a training face. This same face was (erroneously) judged "female", "sure" by all 5 observers.

SexNet.

SexNet achieved a performance of 89% on the set of images that included the outline of the face and some hair, which was comparable to the 88% performance achieved by humans given the same information. When the images of the faces were pared down to eliminate the chin and forehead (which also removed all hair cues), sex judgements were much more difficult for human observers; the performance of the network fell from 89% to 81%, and remained superior to human observers.

The hidden units proved unnecessary to the SexNet's performance; a perceptron (no hidden units) was as successful as networks with up to 40 hidden units, suggesting that the task may be linearly separable. In a followup study (Gray, et al. 1993) we trained a perceptron directly from the gray level images and achieved a performance of 79% for the cropped images. This was quite surprising since it had been previously thought that sex discrimination was a higher-order problem. In Figure 4, the area of each square corresponds to the strength of the weight connecting that image pixel to the output unit. The value of the output unit was computed by multiplying the gray level of each pixel with its corresponding weight, summing the products, and passing the sum through a sigmoid. Since "male" corresponded to an output value of 1, each positive term in the sum (from a positive weight) could be considered evidence that the face is male, and each negative term (from a negative weight) evidence that the face is female.

Evaluation of optimal resizing of the faces (prior to their functioning as inputs to the network) showed the choice of 30x30 images to have been near optimal for network performance (reference). In contrast humans need higher resolution for best performance.

Antecedent compression of faces was not required for good performance of the SexNet, which functioned approximately equally well with the raw gray-level images. (However, having once compressed the images, training time was faster with the reduced input due to fewer connections in the network.)

Although compression was not required for good network performance, use of compression permitted examination of the characteristics of hidden units, which in turn provided insight into characteristics used by the network. (Certainly, the characteristics used by that network with those hidden units; and presumably networks without.) If one hidden unit in the compression subnet is "turned on" (activity set to 1), with all others "off" (activity 0), the compression network's output (the decompressed image) reveals a rather vague-appearing nonspecific face, corresponding to the "holon" of Cottrell and Fleming (Cottrell, et al. 1990), or a mixture of the "eigenfaces" of Turk and Pentland (Turk, & Pentland 1991).

Deciphering characteristics of the SexNet was also possible by examining weights in the network: For example, evaluation of the weights of the SexNet showed presence of an elongated philtrum (the space between the nose and mouth, relative to the distance from eyes to mouth which was standardized) was strongly correlated with maleness. This was apparent both in the simplest perceptron networks (See Fig. 4) and in examination of the hidden units from compression inputs with a particularly strong positive connection to maleness, and a strong negative connection to femaleness.

4 DISCUSSION

Comparison with human performance.

The complex visual pattern recognition task of sexing human faces can be adequately performed by a neural network without prior feature selection and with minimal preprocessing Q contrary to the confident predictions of several of our colleagues. Human performance was matched by a using either a 900 x 40 x 900 back-propagation image compression network such as had previously been used by Cottrell and Fleming for face identification, the activities of whose hidden units served as inputs to a 40x40x1 back-propagation SexNet; or equivalent performance was achieved by a perceptron SexNet which translated the 900 unit input (or the 40 unit reduced input) directly to sex with no intermediate hidden units, indicating that the task is

linearly separable. No efforts to optimize the network were needed to match human performance¹.

The SexNet performance was similar to humans' not just by percent errors. Not only did it correctly sex previously unseen faces, but it had difficulties on faces which also posed difficulties for humans. Indeed on one preliminary trial it correctly assigned all ten test faces, but misjudged two of the 80 training faces. These included the problematic male discussed above, to which it assigned the androgynous value of 0.495, and another male on which it performed wretchedly, with a value between 0.2 and 0.3, despite copious training. The SexNet proved correct: The face was an unambiguous female whose sex value had been mistranscribed in the training data. The SexNet correctly sexed the face based on the other faces, in spite of faulty training information. It had evidently done a fine job of abstracting what distinguishes the sexes.

Failure of humans and the network on the same face suggests how one might handle the network's difficulty, in analogy with human strategies. When a face is found by the network to be female (according to weights which correctly gauge sex of "most" training faces) but the person is male (or vice versa), one shouldn't emend male-female categories too drastically, as one may encounter another nearly identical face which is in fact female. The human strategy confronted with a "training face" (one for which sex is known by other criteria) would consist in making a special category for the individual, and having that provide input to overrule the facial information. If another network were trained to identify problematic individuals then they could be dealt with separately. The outliers could then be correctly identified without adverse consequences to generalization. A similar strategy has proved useful in other domains: In Baxt's neural network which identifies whether patients presenting to the emergency room with chest pain are having a myocardial infarction, superior performance was achieved by training a separate network on the difficult-to-learn cases (Baxt 1992).

¹ The SexNet was in fact slightly better at discriminating sex than the humans tested; however, since the network's performance improved with training, the results may be taken to indicate that the humans tested should spend more time engaged in discriminating sex.

Comparison with other network approaches to sex recognition.

Others have used neural networks for sex identification from faces. The task is referred to by other authors as "gender" recognition, but "gender" is properly a grammatical term: specifically, words have gender, people don't. In English, the term "gender" has attained increasing currency as a euphemistic substitute for (male vs female) "sex". However, "To talk of persons or creatures of the masculine or feminine g., meaning of the male or female sex, is either a jocularity (permissible or not according to context) or a blunder."² Admittedly, the usage is otherwise, but we prefer the anatomically correct term. Neural network efforts on sex recognition begin with Cottrell and Fleming, whose principal focus was face recognition. Their network used a 64 x 64 face with 80 hidden units for compression; this was passed to a perceptron with a unit for each name, a unit for faceness, and units for male and female "gender". They suggest that for image compression the relevant ratio is not that of hidden units to input units, but rather of hidden units to number of patterns, which they suggest should be about 1:1. (Ours was 1:>2.). Their training set consisted of 64 faces and 13 non-face stimuli. They note that their network "failed to accurately categorize novel faces according to gender, making 37% errors on novel faces"; however their the small training set is likely to contribute.

Brunelli and Poggio used Hyper Basis Function networks for "gender" classification. Their strategy differed from ours and that of Cottrell in that they first automatically extracted features from images of pre-normalized scale and rotation; these features, rather than the faces, were used as input. This requires that they have advance insight into which features might be important for the discrimination process; this is a task which is bypassed in the strategy we employed, which requires no advance insight into what might prove useful. They also symmetrized the faces by averaging left and right eyebrow and chin information, which may be reasonable for computer sexing, but detracts from any suggestion of physiological verisimilitude. They used two competing network, one for male recognition and one for female recognition; the outcome was determined by which network gave the greatest response. Only three of the sixteen features used as input developed

² Fowler's Modern English Usage, Oxford University Press, 2nd ed. 1985, revised 1983

significant weights, and two of these may be artifacts of modern sex-specific preening practices: eyebrow thickness and distance from eyebrows to eyes are both artificially modified in women, in the direction of increased eye-to-brow distance and decreased width, by the practice of eyebrow plucking or tweezing (which may be even more widespread in Italy, whence the face images derived, than in the U.S.), so that their preeminence among the discriminating features may rank with hair style, makeup, and jewelry as social rather than biological facial cues. The third feature was nose width. Vertical position of nose and mouth were also given small weights. Their strategy of using competing male and female networks allowed natural determination of the networks' "prototype" male and female faces, constrained, however, by the features selected for the network. These prototypes were observed to resemble not average faces, but caricatures of the sexes, exaggerating the distinctive features. Interestingly, the prototype faces extracted from the networks, in addition to differing in eyebrow findings (and face size) differed perhaps most markedly in the size of the philtrum (which had been identified as important to the SexNet)! Indeed, the nose width, the third "important" feature, was not visibly different in the prototypes. However female nose length is represented as much longer than male, which is introspectively unrealistic Q but may have resulted from a drive to display a small difference between eye-nose and eye-mouth distances (small philtrum). This reinforces the concern that the features selected for use in a feature-driven network may not be the most important (philtrum span was not among the input features mentioned, though it can be obtained by differencing two other features). Additionally, the much longer nose length for the female than the male prototype suggests that prototype faces driven by preselected features cannot be taken at "face" value. This feature-based network showed an average performance of 79% for sexing new faces, compared to 89% for the SexNet including an outline of the face and 81% including only the central region of the face.

Other applications of SexNet.

Although the SexNet task has limited utility of itself Q after all, humans sex human faces fine, without recourse to a neural network Q extensions of this work have application. For instance, it is not known whether faces differ for male and female rhesus monkeys. By training a neural network to perform monkey sexing, and comparing the network's performance on untrained monkey faces to their known sex, better than chance performance would imply there are facial sex differences in rhesus monkeys Q answering a

question of some ethological significance. Physiologists might then be inclined to look for sex-specific face cells in monkey cortex.

In a more lighthearted vein, one could use personality indices rather than sex for the second phase of the net, to scientifically test the tenets of anthroposcopy (physiognomy), possibly for the first time.

More importantly a variety of congenital medical disorders (such as Down syndrome) are accompanied by "craniofacial anomalies" (Dyken, & Miller 1980), resulting in distinctive "facies", or facial appearances. Some are subtle and/or rare, and not often recognized by physicians. It may be possible to screen normal from affected infants or children using special purpose neural networks. We hope to extend our work to encompass neural nets for diagnosing William's syndrome, or infantile hypercalcemia, in which childrens' faces are "elfin-like" (Bellugi, Bahrle, Trauner, Jernigan, & Doherty 1990; Trauner, Bellugi, & Chase 1989). Williams' faces compare to normals in a manner which recalls the male / female distinction in that no isolated well described features occur in all of one but none of the other; rather the gestalt distinguishes them. Early diagnosis is important because these children often have associated cardiac defects requiring surgical correction.

We have extended our work to include neural network analysis of facial expressions, and using similar methodology to that described above, have successfully achieved discrimination among eight facial expressions (with examples of each expression differing in intensity) corresponding crudely to smile, frown, brow-raise, purse-lips, pucker-lips, sneer, squint, and neutral (unpublished). The neutral expression was the most difficult to train, a finding which again correlated to that of human observers, who like the network failed to misclassify this expression as any other, but were reluctant to denote it neutral, either.

References

- Baxt, W. G. (1992). Improving the accuracy of an artificial neural network using multipl differently trained newtorks. *Neural Computation*, 4, 772-780.
- Baylis, G. C., Rolls, E. T., & Leonard, C. M. (1985). Selectivity between faces in the responses of as population of neurons in the cortex in the superior temporal sulcus of the monkey. *Brain Research*, 342, 91-102.
- Bellugi, U., Bahrle, A., Trauner, D., Jernigan, T., & Doherty, S. (1990). Neuropsychological, neurological, and neuroanatomical profile of Williams syndrome children. *American Journal of Medical Genetics*, In Press,
- Benton, A. L., & van Allen, M. W. (1972). Prosopagnosia and facial discrimination. *Journal of Neurological Science*, 15, 157-172.
- Bodamer, J. (1947). Die Prosopagnosie. *Archiv fur Psychiatrie und Nervenkrankheiten*, 179, 6-53.
- Bornstein, B. (1963). Prosopagnosia. In L. Halpern (Ed.), *Problems of dynamic neurology* New York: Grune and Stratton.
- Bornstein, B., Sroka, H., & Munitz, H. (1969). Prosopagnosia with animal face agnosia. *Cortex*, 5, 164-169.
- Bruce, C., Desimone, R., & Gross, C. G. (1981). Visual properties of neurons in a polysensory area in superior temporal sulcus of the macaque. *Journal of Neurophysiology*, 46, 369-384.
- Christen, L., Landis, T., & Regard, M. (1985). Left hemispheric functional compensation in prosopagnosia. A tachistoscopic study with unilaterally lesioned patients. *Human Neurobiology*, 3 or 4,
- Cottrell, G., & Fleming, M. (1990). Face recognition using unsupervised feature extraction. Paris, France: Kluwer Academic Publishers, 322-325.
- Cottrell, G., Munro, P., & Zipser, D. (1987). Learning internal representations of gray scale images: An example of extensional programming. Seattle, Wa.:
- Damasio, A. R., Damasio, H., & van Hoesen, G. W. (1982). Prosopagnosia: anatomic basis and neurobehavioral mechanisms. *Neurology*, 32, 331-341.

- Dyken, P. R., & Miller, M. D. (1980). *Facial Features of Neurologic Syndromes*. St. Louis, Missouri: C.V. Mosby Company.
- Ekman, P. (1973a). Cross-cultural studies of facial expression. In P. Ekman (Ed.), *Darwin and facial expression: A century of research in review* (pp. 169-222). New York: Academic Press.
- Ekman, P. (1973b). *Darwin and Facial Expression: A Century of Research in Review*. New York: Academic Press,
- Ekman, P. (1977). Biological and cultural contributions to body and facial movement. In J. Blacking (Ed.), *Anthropology of the Body* (pp. 39-84). London: Academic Press.
- Ekman, P. (1989). The argument and evidence about universals in facial expressions of emotion. In H. W. a. J. Manstead (Ed.), *Handbook of psychophysiology: Emotion and social behavior* (pp. 143-164). London: John Wiley and Sons.
- Ekman, P., & Friesen, W. V. (1969). Nonverbal leakage and clues to deception. *Psychiatry*, 32(1), 88-105.
- Ekman, P., Friesen, W. V., & Simons, R. C. (1985). Is the startle reaction an emotion? *Journal of Personality and Social Psychology*, 49(5), 1416-1426.
- Ekman, P., Levenson, R. W., & Friesen, W. V. (1983). Autonomic nervous system activity distinguishes between emotions. *Science*, 221, 1208-1210.
- Gloning, I., Gloning, K., Jellinger, K., & Quatember, R. (1970). A case of "prosopagnosia" with necropsy findings. *Neuropsychologia*, 8, 199-204.
- Golomb, B. A., Lawrence, D. T., & Sejnowski, T. J. (1991). SEXnet: A neural network identifies sex from human faces. In D. S. Touretzky, & R. Lippmann (Ed.), *Advances in Neural Information Processing Systems*, Vol. 3 San Mateo, California: Morgan Kaufmann.
- Gray, M., Lawrence, D., Golomb, B. A., & Sejnowski, T. J. (1993). A perceptron reveals the face of sex. UCSD Institute for Neural Computation Technical Report INC 93-03.

- Gross, C. G., Rocha-Miranda, C. E., & Bender, D. B. (1972). Visual properties of neurons in inferotemporal cortex of the macaque. *Neurophysiology*, 35, 96-111.
- Grusser, O. J., Selke, T., & Zynda, B. (1985). Age dependent recognition of faces and vases in children and adolescents. *Human Neurobiology*, 4, 33-39.
- Hecaen, H., & Angelergues, R. (1962). Agnosia for faces (prosopagnosia). *Archives of Neurology*, 7, 92-100.
- Horel, J. A., Keating, E. G., & Misantone, L. G. (1972). Kluver-Bucy syndrome produced by destroying neocortex or amygdala. *Brain Research*, 94, 347-359.
- Jeeves, M. A. (1984). The historical roots and recurring issues of neurobiological studies of face perception. *Human Neurobiology*, 3, 191-196.
- Jones, B., & Mishkin, M. (1972). Limbic lesions and the problem of stimulus-reinforcement associations. *Experimental Neurology*, 36, 362-377.
- Kluver, H., & Bucy, P. C. (1939). Preliminary analysis of functions of the temporal lobes in monkeys. *Arch. Neurol. Psychiat.*, 42, 979-1000.
- Leinonen, L., & Nyman, G. (1979). Functional properties of cells in antero-lateral part of area 7 associative face area of awake monkeys. *Experimental Brain Research*, 34, 321-333.
- McGurk, H., & MacDonald, J. (1976). Hearing lips and seeing voices. *Nature*, 264, 746-748.
- Meadows, J. C. (1974). The anatomical basis of prosopagnosia. *J. Neurol. Neurosurg. Psychiat.*, 37, 489-501.
- O'Toole, A. J., Abci, H., Deffenbacher, K. A., & Valentin, D. (1993). Low-dimensional representation of faces in higher dimensions of the face space. *Journal of the Optical Society of America*, A10, 405-411.
- O'Toole, A. J., Millward, R. B., & Anderson, J. A. (1988). A physical system approach to recognition memory for spatially transformed faces. *Neural Networks*, 1, 179-199.

- Parker, D. B. (1986). A comparison of algorithms for neuron-like cells. In J. S. Denker (Ed.), *Neural networks for computing* New York: American Institute of Physics.
- Perrett, D. I., Rolls, E. T., & Caan, W. (1982). Visual neurones responsive to faces in the monkey temporal cortex. *Experimental Brain Research*, 47, 329-342.
- Perrett, D. I., Smith, P. A. J., Potter, D. D., Mistlin, A. J., Head, A. S., Milner, A. D., & Jeeves, M. A. (1984). Neurones responsive to faces in the temporal cortex: studies of functional organization, sensitivity to identity and relation to perception. *Human Neurobiology*, 3, 197-208.
- Pigarev, I. N., Rizzolatti, G., & Scandolara, C. (1979). Neurones responding to visual stimuli in the frontal lobe of macaque monkeys. *Neuroscience Letters*, 12, 207-212.
- Rolls, E. T. (1981). Responses of amygdaloid neurons in the primate. In Y. Ben-Ari (Ed.), *The Amygdaloid Complex* (pp. 383-393). Amsterdam: Elsevier.
- Rolls, E. T. (1984). Neurons in the cortex of the temporal lobe and in the amygdala of the monkey with responses selective for faces. *Human Neurobiology*, 3, 209-222.
- Rolls, E. T. (1992). Neurophysiological mechanisms underlying face processing within and beyond the temporal cortical visual areas. *Philosophical Transactions of the Royal Society of London.*, B335, 11-20.
- Rolls, W. T., Judge, S. J., & Sanghera, M. K. (1977). Activity of neurones in the inferotemporal cortex of the alert monkey. *Brain Research*, 130, 229-238.
- Rumelhart, D. E., Hinton, G., & Williams, R. J. (1986). Learning internal representation by error propagation. In D. E. R. a. J. L. McClelland (Ed.), *Parallel Distributed Processing, Explorations in the microstructure of cognition* (pp. 318-362). Cambridge, Mass.: MIT Press.
- Sanghera, M. K., Rolls, E. T., & Roper-Hall, A. (1979). Visual responses of neurons in the dorsolateral amygdala of the alert monkey. *Experimental Neurology*, 63, 610-626.

- Schwartz, G. E., Weinberger, D. A., & Singer, J. A. (1981). Cardiovascular differentiation of happiness, sadness, anger and fear following imagery and exercise. *Psychosomatic Medicine*, 43, 343-363.
- Sergent, J., Ohta, S., & MacDonald, B. (1992). Functional neuroanatomy of face and object processing. A positron emission tomography study. *Brain*, 115, 15-36.
- Sternbach, R. A. (1962). Assessing differential autonomic patterns in emotions. *Journal of Psychosomatic Research*, 6, 87-91.
- Tranel, D., Damasio, A. R., & Damasio, H. (1988). Intact recognition of facial expression, gender, and age in patients with impaired recognition of face identity. *Neurology*, 38(5), 690-696.
- Trauner, D., Bellugi, U., & Chase, C. (1989). Neurologic features of Williams and Down Syndromes. *Pediatric Neurology*, 5(3), 166-168.
- Trezean, H., Dalle, & Ore, G. (1955). Syndrome of Kluver and Bucy reproduced in man by bilateral removal of the temporal lobes. *Neurology*, 5, 373-380.
- Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 71-86.
- Werbos, P. (1974). *Beyond Regression: New tools for prediction and analysis in the behavioral sciences*. Harvard University,
- Whiteley, A. M., & Warrington, E. K. (1977). Prosopagnosia: a clinical, psychological and anatomical study in three patients. *J. Neurol. Neurosurg. Psychiat.*, 40, 394-430.
- Young, M. P., & Yamane, S. (1992). Sparse population coding of faces in the inferotemporal cortex. *Science*, 256, 1327-1331.

ANN BASED CLASSIFICATION OF ARRHYTHMIAS

M. Jabri, S. Pickard, P. Leong,
Z. Chi, E. Tinker, R. Coggins and B. Flower

*Systems Engineering and Design Automation Laboratory
Sydney University Electrical Engineering
NSW 2006, Australia*

1 INTRODUCTION

Implantable cardioverter-defibrillators (ICDs) represent an important therapy for people susceptible to sudden cardiac death. These devices monitor the heart for abnormal rhythms, and can deliver either pacing or shock therapy to terminate the episode. ICDs sense the electrical activity of the heart through leads attached to the internal surface. Present devices use a single ventricular lead inserted in the right ventricular apex (RVA). Future devices will use an additional atrial lead placed in the high right atrium (HRA). These leads measure electrical potential and recordings made from such leads are called intracardiac electrograms (ICEGs).

Although there are many different types of abnormal heart rhythms (arrhythmias), for ICDs, they are classified into four groups, normal sinus rhythm, supraventricular tachycardia, ventricular tachycardia and ventricular fibrillation. This is, in part, due to the limited number of different therapies available in an ICD: pacing, low energy shocks and high energy shocks. Figure 1 shows an example of a Normal Sinus Rhythm. Note the regularity of the beats. Of interest to us is what is called the QRS complex, which represents the electrical activity in the ventricle during a beat (see the beat in the middle of Figure 1). The R point (also called the R wave) represents the peak, and the distance between two heart beats is usually called the RR interval.

Figure 2 shows an example of a Ventricular Tachycardia (more precisely a Ventricular Tachycardia Slow or VTS). Note that the beats are faster in comparison with an NSR. Ventricular Fibrillation (VF) is shown in Figure 3. Note the chaotic behaviour and the absence of well defined heart beats. For an NSR,

Figure 1 A Normal Sinus Rhythm (NSR) waveform (RVA channel)

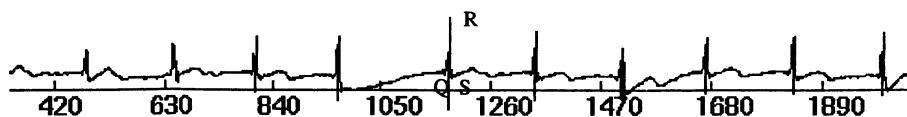


Figure 2 A Ventricular Tachycardia (VT) waveform (RVA channel)

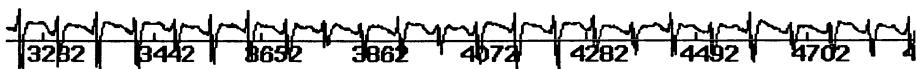


Figure 3 A Ventricular Fibrillation (VF) waveform. (RVA channel)



no therapy is delivered. For a VT an action of “pacing” is performed, and if this is not successful, shock therapy is applied. For VF, a high energy shock is issued. Present devices cannot recognise SVT, and so no therapy is available for it. However, experiments have shown that pacing of the atrium can terminate SVT. Because they are near field signals, ICEGs are different from surface electrograms (ECG). As a result, classification algorithms developed for ECG patterns may not necessarily be valuable for ICEG recognition.

The difficulties in ICEG classification lie in that many arrhythmias have similar features. For instance, many ICDs make use of the heart rate as a fundamental feature in the arrhythmia classification process. However, several arrhythmias that require different types of therapeutic actions have similar heart rates. For example, a Sinus Tachycardia (ST) is an arrhythmia characterised by a heart rate that is higher than that of an NSR and close to a VT. Many classifiers would classify an ST as VT leading to a therapy of pacing, whereas an ST is supposed to be grouped under an NSR type of therapy. Another example is a fast VT which may be associated with heart rates that are indicative of VF. In this case the defibrillator would apply a VF type of therapy when only a VT type therapy is required.

The overlap of the classes when only heart rate is used as the main classification feature highlights the necessity of the consideration of further features with higher discrimination capabilities. Features that are commonly used besides the heart rate are:

1. average heart rate (over a period of time)
2. arrhythmia onset
3. arrhythmia probability density functions

Because of the limited power budget and area, arrhythmia classifiers in ICDs are kept extremely simple with respect to what could be achieved with more relaxed implementation constraints.

Artificial neural network techniques offer the potential of higher classification performance at an acceptable and competitive implementation area and power supply requirements. To achieve minimum power consumption, VLSI micropower implementation techniques are used.

We have described in earlier reports the potential of ANN techniques in ICEG classification [1], [7], [4]. Here we present a more conclusive study that is based

on a much larger patient database. Our research shows that although ANN techniques may contribute to many classification aspects in implantable heart defibrillators, the most spectacular potential is achieved in implementing the classifiers using analogue subthreshold techniques. The resulting devices can be used in either single or dual lead ICDs.

In the next section we describe the database used in our studies. In Section 3 ANN based classifiers for single chamber ICDs are considered. Then in Section 4 we treat the dual chamber case. We present microelectronic implementations in Section 5. Finally, the work and the contribution of ANN to ICDs are discussed in Section 6.

2 DATA, PREPROCESSING AND FEATURE EXTRACTION

1 Database

Data used in our experiments were collected from electrophysiological studies (EPS) at hospitals in Australia and the UK. This technique involves introducing temporary probes into the internal surface of the heart, and inducing arrhythmias by stimulating the heart through them. The same probes can be used to monitor the electrical activity of the heart, and the arrhythmias are diagnosed from this activity. Altogether, and depending on whether single or dual chamber is considered, data from over 150 patients are available. Cardiologists from our commercial collaborator have labelled these data. All tests were performed on testing sets that are not used in the classifier's creation process. Arrhythmias recorded during EPS are produced by stimulation. As a result, no natural transitions from normal rhythms to arrhythmias are captured and so it is not possible to use arrhythmia onset in the classification process.

2 Preprocessing

Obviously, to minimise power and implementation area, we wish to perform as little as possible preprocessing on our raw data. That is, the data that has been collected in EPS studies should preferably not require any further filtering or preprocessing. In reality, the data representing the electrical activities and sensed at the heart tissue is analogue and will undergo some filtering as it is performed by most ICDs. The most common is to band filter the data with a

3Hz-45Hz filter. The data used in our experiments was band filtered to 3Hz-125Hz.

3 Feature Extraction

The feature extraction that we perform is centered around the ability to detect the R waves (also called QRS detection) in an ICEG. Once waves are identified, the RR intervals can be determined. All current ICD devices have QRS detectors and the one used in our research compares the slope of the ICEG with an adaptive threshold to identify the R wave. Such detectors do not always detect the peak accurately, but a point that is very close to it. This has implications with regard to morphology based classification and analysis that aim to recognise the shape of the waveform made of samples captured at the RVA lead. The errors introduced by bad R point detection may lead to failures in the morphology classification. Time shift invariance can be achieved by further preprocessing at a higher power and area costs [5].

Probability Density Functions (PDF) of the RR is a measure of confidence in separating VT and VF. Given the means M_{vt} and M_{vf} of the RR for a VT and VF respectively across a database, and SD_{vt} and SD_{vf} being the associated standard deviations, a PDF measure is computed as:

$$\text{PDF} = \frac{\sum_{i=1}^N (RR_i - M_{vf})^2}{(SD_{vf})^2} - \frac{\sum_{i=1}^N (RR_i - M_{vt})^2}{(SD_{vt})^2} \quad (1)$$

Where N defines a window of consecutive RRs over which the PDF is being considered.

4 Morphology

The morphology of the QRS complex can provide additional clues towards the identification of the arrhythmia. This information is particularly important when the heart rate (RR intervals) by itself is not sufficient to identify the arrhythmia. For instance, Sinus Tachycardia is an arrhythmia that is treated as an NSR but may be associated with a heart rate that is close to a slow VT. Because some VTs feature a widening of the QRS complex, morphology analysis will help in producing the correct classification.

Morphology analysis is a common technique in ECG recognition especially where no strict power or area constraints exist. Many algorithms have been developed for this purpose including correlation waveform analysis (CWA) and

bin analysis. The CWA technique corresponds to standard template matching and a correlation coefficient is used to decide whether a morphology match exist between the present QRS and a reference set. The bin area method is similar to CWA but is less computationally expensive.

5 Patient Dependent and Patient Independent Classifiers

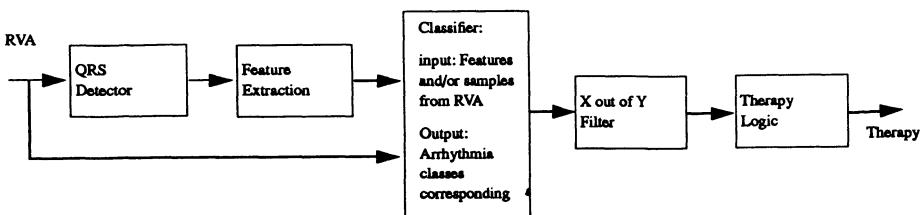
Patient independent classifiers are, of course, the desired solution to the ICEG classification problem since they require less tuning for a particular patient. However, patient dependent classifiers in which certain parameters are tuned for a particular patient, will outperform patient independent classifiers in nearly all cases. Note that morphology analysis is difficult to achieve in a patient independent fashion and variations across patients are significant. In fact, the morphology of the signal of a particular patient may change over time due to the growth of tissue around the leads or the effects of drugs.

3 SINGLE CHAMBER CLASSIFICATION

A block diagram of a classifier in a single chamber ICD is depicted in Figure 4. The QRS detector detects the R waves. The X out of Y filter is used to smooth out all spurious classifications producing therefore a decision that is based on a “majority” (at least X) vote over a number (Y) of classifications. The Therapy Logic block assigns the therapy that corresponds to the recognised class. Note the RVA input to the classifier is only required if a morphology analysis is to be performed. Of interest to us here is the design of a reliable classifier that can be implemented in as small area as possible and that would consume as little power as possible.

We have evaluated several artificial neural computing techniques to implement such classifiers. For single chamber, not all arrhythmias could be reliably classified (not even by human experts). This is because data from the RVA lead represents only the ventricular electrical activity, and for many arrhythmias, this is insufficient information. Cardiologists normally require at least two leads and a surface ECG, and patient history to classify arrhythmia. We will see later in Section 4 how classification can be improved by introducing information from an atrial lead.

Figure 4 Classifier in a single chamber ICD



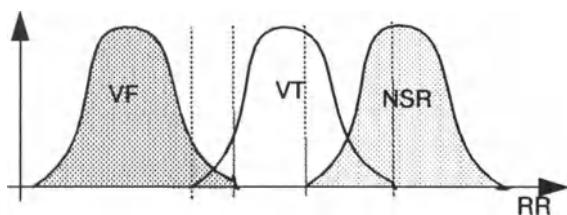
1 Rate Based Classification

As a comparison to the ANN classifiers, we first present a common classification technique currently used in implantable classification systems. This method simply examines the RR timing interval alone to decide if the heart rhythm is a VF, VT, or NSR. Typically, the VT type arrhythmia has a faster heart rate than the Normal Sinus Rhythm, and the VF arrhythmia has a still faster rate. Figure 5 shows a typical probability distribution of the three arrhythmia classes over the possible RR interval range.

An RR rate only based classifier only needs to compare a given RR interval with two predetermined thresholds to decide which class it belongs. The predetermined thresholds are calculated using a set of training examples of RR intervals. Our example used 3,814 examples from 51 patients to find the optimum thresholds to achieve the best classification results on the training set. These thresholds were then applied to our entire patient database of 153 patients or 12,596 examples. The classifier performs with a best overall classification performance of 89.9%.

This classifier fails where the VT arrhythmia rate overlaps with that of the VF

Figure 5 NSR, VT, and VF Arrhythmia Distribution based on RR Timing Feature



Class	NSR	VT	VF	Accuracy (%)
NSR	8420	27	16	99.5
VT	957	680	55	40.2
VF	17	198	2226	91.2

Table 1 Performance of Rate Based Only Classification

Class	NSR	VT	VF	Accuracy(%)
NSR	16848	46	32	99.5
VT	2158	1116	110	33.0
VF	40	390	4452	91.2

Table 2 Performance of MLP trained on Rate Based Feature Only

and NSR classes. This overlap is more prevalent between NSR and VT classes.

2 Multilayer Perceptrons

Training A Multilayer Perceptron for Rate Based Classification

This experiment trained a 1-6-3 MLP with only the RR feature as input by presenting it with 3,814 examples of VF, VT and NSR classes from 51 patients. The network was trained using conjugate gradient decent to a performance of 91.1%. It was then tested using a patient database of 153 patients or 12,596 examples and generalised to an overall performance of 89.0%.

Timing Only Based Classification

In another experiment, a 14-10-3 MLP classifier has been used to classify NSR, VT and VF using the five RR intervals, the last five probability density functions (PDFs), average RR, standard deviation, and average difference of RR intervals as input features. The training set includes 4161 out of a total of 16659 observations obtained from 147 patients. The remaining 12498 observations are included in the testing set. Table 3 shows the result of one experiment. The MLP classifier achieves a reasonable good performance over 6 subclasses (average 85.1%). The result also confirms the overlap amongst ST, VTF and VT. The classification can be improved to some extent using a postprocessing scheme like X out of Y filtering.

3 Multi-Module Architecture

The multi-module approach uses two or more separate MLP networks and combines their outputs with another MLP to generate the final classification. In

Subclass	Class	NSR	VT	VF	Accuracy(%)
NSR	NSR	7133	218	4	97.0
ST	NSR	1613	585	3	73.3
VTS	VT	6	48	0	88.9
VT	VT	89	562	95	75.3
VT 1:1	VT	81	463	93	72.7
VF	VF	0	0	634	100
VTF	VF	0	100	771	88.5

Table 3 Performance of the MLP classifier for single chamber classification

our example, we use three separate MLP's (see Figure 6). The *Timing* MLP is trained with only the AVR (average RR interval) feature. The *Morphology* MLP is trained to classify arrhythmia based on only 6 RVA samples centred around the QRS point as input. Finally, the *Combine* MLP is trained to combine the outputs of the *Timing* and *Morphology* MLP into one classification.

This architecture was trained on 225 training vectors from 51 patients and was tested on all patient data at 81.5% correct classifications. This does not perform as well as other designs that we have examined when trained on a patient independent level. This is because morphology is a patient dependent feature. The usefulness of this architecture is in it's modularity that allows the *Morphology* net to be retrained patient dependently while the *Timing* net is left alone.

The performance of this design was tested in a patient specific manner by retraining only the *Morphology* and *Combine* networks with a small training set of no more then 6 examples of each arrhythmia from a given patient, and then all patient vectors were tested. The histograms in Figure 7 show the performance of this architecture when tested on each of 153 different patients. Results for patient independent training only, and patient dependent training of the *Morphology* and *Combine* modules, are shown.

Notice that while the overall patient independent performance was 81.5%, many patients performed below 50% individually. After patient dependent training, all patients were classified over 50%, and most above 90%!

Figure 6 Multi-Module Architecture for ICEG Classification

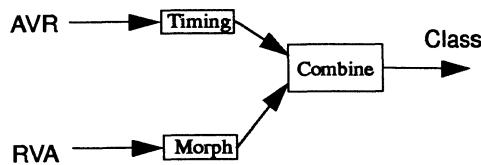
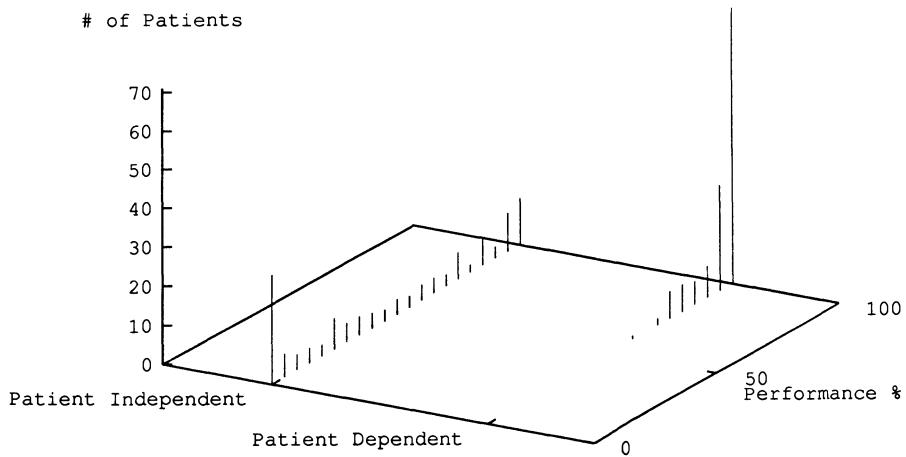


Figure 7 Histogram for Patient Dependent Multi-Module Testing.



Subclass	Class	NSR	VT	VF	Accuracy(%)
NSR	NSR	7155	183	17	97.3
ST	NSR	1743	452	6	79.2
VTS	VT	11	43	0	79.6
VT	VT	69	621	56	83.2
VT 1:1	VT	52	495	90	77.7
VF	VF	0	16	618	97.5
VTF	VF	0	150	721	82.8

Table 4 Performance of the decision tree classifier for single chamber classification

4 Induction Of Decision Trees

Building knowledge-based systems by inductive inference from examples has been proved successful in several practical applications. ID3 is a system to create decision trees using an information-based method. In our study, the C4 package (an ID3 derivative) developed by Ross Quinlan (Quinlan, 86) has been used to perform arrhythmia classification. Table 2 shows experiment results when same training set and testing set as in Section 3.1 were used. Experiment results shows that a decision tree classifier has similar generalisation capability as an MLP classifier. The design tree classifier achieves 91.2% (average 85.3%). If input features are corrupted by noise, a decision tree classifier can not achieve as good performance as an MLP classifier [2]. The reason is that an MLP classifier can capture statistical information of training patterns.

The implementation of decision trees in VLSI is not a difficult procedure, however, the branching thresholds are difficult to implement in digital technology (for large trees) and even more difficult to implement in micropower analogue VLSI. The latter implementation technique would be possible if the induction process can take advantage of the hardware characteristics in a similar way that “in-loop” training of subthreshold VLSI MLP achieves the same objective.

5 Hybrid Rate Based and Morphological Classifier

Some of the best classification results with single chamber classifiers can be obtained with this last technique that combines a rate based decision with a neural network morphology classifier. The rate only based classifier discussed

Class	NSR	VT	VF	Accuracy (%)
NSR	8401	46	16	99.3
VT	637	1000	55	59.1
VF	12	203	2226	91.2

Table 5 Performance for Hybrid Single Channel Classifier

in section 3.1 performs well except for the cases where there is overlap between the timing of the VT and VF classes and particularly the VT and NSR classes as depicted in Figure 5. The hybrid technique uses a network trained on morphology samples to classify the observations where the timing information is not conclusive as to either a VT or NSR classification.

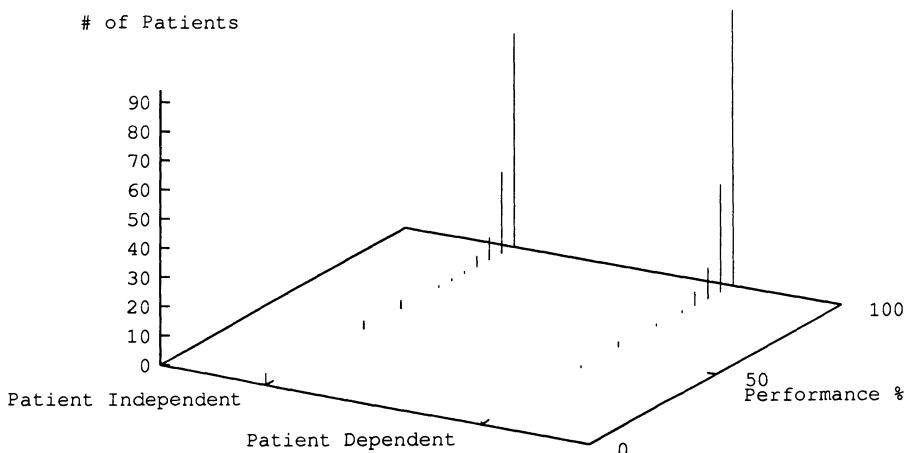
In our example, the rate thresholds were computed by optimisation for a training set of 3,814 examples from 51 patients. A 10-6-2 network was trained on 10 morphology samples taken around the R wave of the QRS wave and sampled at half the normal sampling rate. The training set was 1,207 examples of VT and NSR classes from the 51 training patients. The test set of 12,596 observations from 153 patients was classified with an over all performance of 92.3%.

The system above was trained in a patient independent way to make a general classifier that can work across several patients. However, experiments show that classification performance can be further increased if the morphology network alone is trained specific to individual patients.

To train this hybrid classifier patient individually, the morphology net was trained on a small training set of no more then six VT and six NSR examples from a patient. Then the classifier was tested on all the patient's vectors.

The histogram in Figure 8 shows the classifier's performance on individual patients when trained patient independently only, and when the morphology net was trained patient dependently.

Figure 8 Histogram for Patient Dependent Hybrid Single Channel Classifier Testing



4 DUAL CHAMBER BASED CLASSIFICATION

1 Hybrid Decision Tree/MLP

The hybrid decision tree/multilayer perceptron “mimics” the classification process as performed by cardiologists. The architecture of the classifier is shown in Figure 9. The decision tree is used to produce classification based on:

1. The rate aspects of the ventricular and atrial channels,
2. The relative timing between the atrial and ventricular beats.

In parallel with the decision tree, a morphology based classifier is used to perform template matching. This a simple MLP with inputs that are signal samples (sampled at half the speed of the normal sampling rate of the signal), and is used to distinguish between arrhythmias that cannot be identified based on timing alone. The outputs of the timing and morphology classifiers are fed

Subclass	Class	NSR	SVT	SVT	VF
NSR	NSR	5605	4	2	0
ST	NSR	1535	24	2	1
SVT	SVT	0	1022	0	0
AT	SVT	0	52	0	0
AF	SVT	0	165	0	0
VT	VT	0	0	322	0
VT 1:1	VT	2	0	1253	0
VF	VF	0	0	2	196
VTF	VF	0	2	0	116

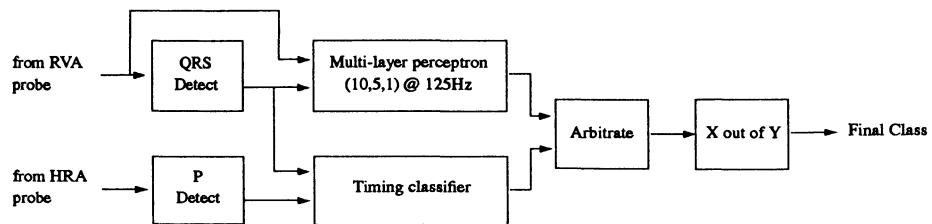
Table 6 Performance of the hybrid decision tree/MLP classifier for dual chamber classification (floating point software neural network)

into an arbitrator that combines them to produce a single classification. The output of the arbitrator is then passed through an "X out of Y" filter to remove spurious classifications due to noise, missed QRS detections and ectopic beats. Further details on the implementation and the operation of the hybrid classifier can be found in [7]. This classifier achieves a high classification performance over a wide range of arrhythmias. Table 6 shows the results of applying this technique to a database of 12483 QRS complexes recorded from 67 patients. A classification performance of 99.6% was achieved.

5 MICROELECTRONIC IMPLEMENTATIONS

In all our classifier architecture investigations, microelectronic implementation considerations were a constant constraint. Many other architectures that can achieve competitive performance were not discussed here because of their unsuitability for low power/small area implementation. The main challenge in a low power/small area VLSI implementation of classifiers similar to those discussed above, is how to implement in very low power an MLP architecture that can reliably learn and achieve a performance comparable to that of the functional simulations. Several design strategies can achieve the low power and small area objectives. Both digital and analogue implementation techniques are being investigated and we report here on our analogue implementation efforts only. Our analogue implementations make use of the subthreshold region of

Figure 9 Architecture of the hybrid decision tree/neural network classifier



MOS transistor conduction to maintain a very low power dissipation.

1 The Kakadu Chip

The Kakadu chip makes use of multiplying digital to analogue converters to implement the synapses [6]. This technique has the advantage of allowing weights to be stored in digital register while keeping the signal processing operations in the analogue domain. All neurons were implemented as external resistors since these allow a controllable neuron transfer function to be used and they also provide convenient test points for the chip. Figure 10(b) shows the schematics for the synapse.

Unlike a conventional neural network, Kakadu implements linear neurons and nonlinear synapses. This method was used since it allowed us to use the nonlinearity of the analogue multiplier to an advantage. Kakadu's transfer function can be expressed as:

$$u_i = \sum_{j=1}^{N_l} w_{ij} \tanh\left(\frac{\kappa a_j}{2}\right) \quad (2)$$

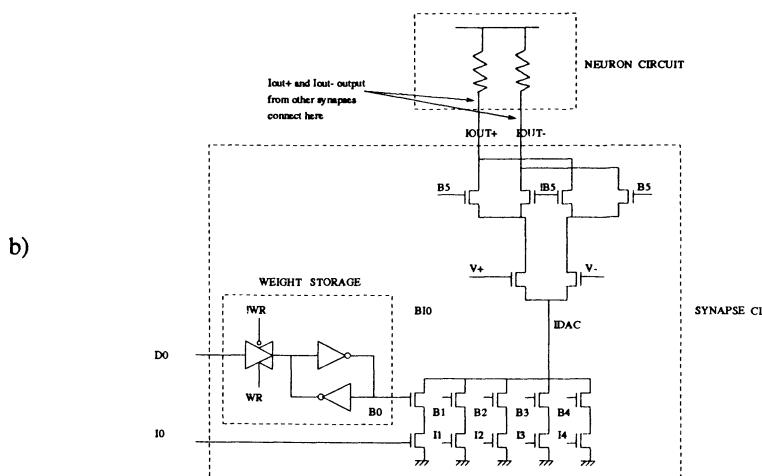
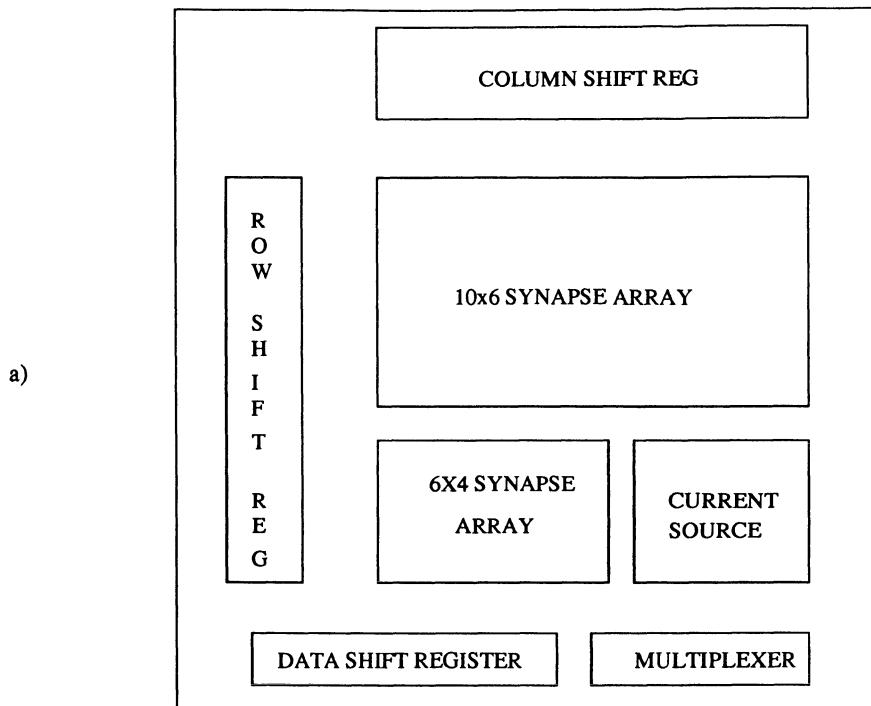
$$a_i = \alpha u_i \quad (3)$$

where the w_{ij} represent the synapse connections to neuron i , u_i is the summed output of the synapse, a_i is the neuron output, α is the neuron gain (the resistance), κ is a constant, l denotes the layer number, L is the total number of layers, N_l is the number of neuron units in the l th layer and i is the neuron number.

The Kakadu chip implements a 10-6-4 feedforward network (84 synapses), see Figure 10(a), and has been successfully tested on several classification problems including ICEG morphology classification, XOR, 3 bit parity, 4 bit parity and simple character recognition.

Kakadu was used in the MLP part of the Hybrid Decision Tree/MLP classifier described earlier. When used to classify the same database as that used to obtain Table 1, the classification performance achieved was 99.3%, with only a slight degradation being experienced (see Table 7). While performing this arrhythmia classification task, Kakadu has a power consumption of less than 25 microwatts. It has a propagation delay of approximately 30 microseconds

Figure 10 The Kakadu chip (a) Floorplan, (b) schematic of synapse



Subclass	Class	NSR	SVT	VT	VF
NSR	NSR	5406	4	20	0
ST	NSR	1535	24	2	1
SVT	SVT	0	1022	0	0
AT	SVT	0	52	0	0
AF	SVT	0	165	0	0
VT	VT	0	0	322	0
VT 1:1	VT	0	16	974	0
VF	VF	0	0	2	196
VTF	VF	0	2	0	116

Table 7 Performance of the hybrid decision tree/MLP classifier for dual chamber classification (Kakadu Chip)

and QRS complexes occur at a frequency of approximately 1 Hz. If we allow 1000 microseconds for the propagation, and provide a zero bias to Kakadu when it is not being used, a continuous power consumption of less than 25 nanowatts can be achieved.

6 CONCLUSIONS

We have presented architectures for single and dual chamber arrhythmia classifiers. In both cases a good classification performance was achieved. In particular, for dual chamber classification, the complexity of the problem calls on more structured classifier architectures. A microelectronic low power implementation was presented that can perform classification with an average power consumption of 25 nanowatts. Progress so far indicates that micropower VLSI ANNs offer a technology that will enable the use of such classification strategies to improve the performance of existing ICD devices.

7 ACKNOWLEDGEMENT

This work was supported by the Australian Department of Industry Technology & Commerce and Telecommunications Pacing Systems, Sydney, Australia.

REFERENCES

- [1] Z. Chi and M. Jabri (1991), "Identification of Supraventricular and Ven-

- tricular Arrhythmias Using a Combination of Three Neural Networks". Proceedings of the Computers in Cardiology Conference, Venice, Italy, September 1991
- [2] Z. Chi and M. Jabri (1991), "A Comparison of MLP and ID3-derived Approaches for ECG Classification", Proceedings of the Second Australian Conference on Neural Networks, Sydney, Australia.
 - [3] M. Jabri and B. Flower (1991), "Weight Perturbations: An optimal architecture and learning technique for analogue VLSI feed-forward and recurrent multi-layer networks", Neural Computation, Vol. 3, No. 4, MIT Press.
 - [4] M. Jabri, S. Pickard, P. Leong, Z. Chi, B. Flower and Y. Xie, "ANN based classification for heart defibrillators", NIPS 4, pp 637-644, Morgan Kaufmann publishers, 1992.
 - [5] S. Lee (1990), "Using a translation-invariant neural network to diagnose heart arrhythmia", NIPS 2, pp. 240-247, Morgan Kaufmann Publishers, 1990.
 - [6] P.H.W. Leong and M. Jabri (1993), "A VLSI Arrhythmia Classifier", Proceedings of the Fourth Australian Conference on Neural Networks, pp. 41-44, Melbourne, Australia.
 - [7] P.H.W. Leong and M. Jabri (1992a), "MATIC - An Intracardiac Tachycardia Classification System", Pacing and Clinical Electrophysiology (PACE), Sept. 1992, USA.
 - [8] S. Pickard, M. Jabri, P.H.W. Leong, B.G. Flower and P. Henderson (1992), "Low Power Analogue VLSI Implementation of A Feed-Forward Neural Network", Proceedings of the Third Australian Conference on Neural Networks, pp. 88-91, Canberra, Australia.
 - [9] J.R. Quinlan (1986), "Induction of decision trees", Machine Learning, 1(1):81-106.

CLASSIFICATION OF CELLS IN CERVICAL SMEARS

Mathilde E. Boon and L.P. Kok

*Leiden Cytology and Pathology Laboratory,
P.O. Box 16047, 2301 GB Leiden,
The Netherlands,*

*Institute for Theoretical Physics,
University of Groningen, Nijenborgh 4,
9747 AG Groningen,
The Netherlands.
E-mail: lpkok@th.rug.nl*

1 THE NEED FOR AUTOMATIC PRESCREENING

Presently, an increased effort to realize computer-assisted screening is generated in response to the American shortage of cytotechnologists after the Wall Street Journal in 1987 carried a major article questioning the accuracy of cervical screening in the USA (Bogdanich, 1987). Federal legislation was implemented limiting the number of smears screened (this number could formerly be as high as 300 per day for one cytotechnologist) and doubling in some states the salaries in six months.

Interest in cytology automation dates to the 1950s, but for many years such automation was technically impossible or prohibited by costs. These problems have been overcome as computers have become more powerful and less expensive. In 1992, several commercial firms have started to market their products, including Accuron Corporation, CYTYC Corporation, Neopath, Neuromedical Systems, Roche, and X-illix Technologies Corporation (Linder, 1992). It is not surprising that all these firms are active in the USA (IAC Committee, 1991). For many years, the emphasis was on developing instruments which could both find abnormal cells and classify the abnormality, with no human intervention, thus could generate a pathology report (Anderson and Nelson, 1992). Given the lack of consensus of such classification and incomplete understanding of the biological behavior of the intraepithelial lesions, this is not an attractive option. Recently, the concept has emerged of computer-aided instead of user-independent screening. Here, the human observer is responsible for the final diagnosis partly based on the abnormal cells classified by the system.

Screening as many as 100 smears per day is still common practice in the USA. In The Netherlands, the work load for a cytotechnician is much less, no more than 50 smears a day. The screening is preferably performed in the morning hours when the alertness is at its peak. But also in these almost ideal Dutch working conditions the major factor for false negativity proved not to be smear inadequacy but the failure of the cytotechnologist to detect the abnormal cells in the smear. It would be desirable to obtain a system in the laboratory relieving much of the cytotechnicians' burden by flagging the abnormal cells (Bartoo, 1992).

Several strategies can be followed and have been proposed. On the one hand, monolayer smears can be made from cell suspensions to facilitate computer analysis of the cells. A disadvantage here is that these systems cannot be tested on archive material. On the other hand, well-known conventional smears can be used applying neural-network technology, well suited for the problems of the uneven routine smears. Neural networks have been tested, amongst others, in histopathology (O'Leary et al., 1992). In this chapter we describe how neural-network programming is exploited by the firm Neuromedical Systems, by the introduction of the PAPNET system. Here we are dealing with computer-assisted screening, in which the cells classified by the neural networks are reviewed by the cytodiagnostician. In abnormal cases, the final diagnosis is a light-microscopical one, based on the parts of the smear selected by PAPNET.

First we will provide some information about the phenomenon neural of networks in general and focus on PAPNET, and second we will present our own experience with PAPNET.

2 APPPLICATION OF ARTIFICIAL NEURAL NETWORKS

Neural networks can classify even in cases where it is difficult to formulate simple rules to distinguish between different categories (Müller and Reinhardt, 1990). Correspondingly, the Pap smear provides a challenge to the academic and the commercial sector. In response to this challenge, the PAPNET system has been developed to facilitate cytodiagnosis.

The rules governing the recognition procedure of neural networks are generated by a process of trial and error. The network literally teaches itself how to do a task. It does so by comparing its results to the examples shown to it. It is therefore very hard to answer the question often asked by the cytotechnologists: why does the network select this cell or image, and not that one? An analogous question would be: why does a person select a certain face in a crowd?

3 THE PAPNET SYSTEM

The PAPNET system is based on a standard multilayer perceptron. First, all 300,000 or more cells on each conventionally prepared Pap smear are first scanned using a low-power objective to find areas of cellularity to which the medium-powered lens is directed. This scan detects those objects submitted (typically 10,000 to 80,000 per slide) which require further analysis by the algorithmic computer. Second, two neural networks are used for the selected images. One neural network is trained to be sensitive to the detection of single abnormal cells and the other is specifically sensitive to the detection of cell groupings. Each network classifies all of the candidate objects which pass the algorithmic criteria on a scale of 0.1000 (appearing least abnormal to the net) to 0.9000 (appearing most abnormal to the net). The highest ranking 64 image fields from each neural network are then revisited under high power (about 400 \times) and it is these 128 image fields (2 \times 64) or 'tiles' which are stored on the DAT tape for cytodiagnostic review. Thus from each slide two sets of 64 tiles ('page 1' and 'page 2') are available for the diagnostician to be displayed on high-definition color television screen. On page 1 single classified abnormal cells can be found and on page 2 abnormal cell clusters. In the absence of abnormal cells, only normal cells are displayed, on page 1 predominantly metaplastic cells, in page 2 clusters of benign endocervical cells. In addition 'unknown objects' are displayed on the tiles, for instance oxyuris eggs.

One by one, the classified tiles appear on the screen, to a total of 64 tiles, see Fig. 5.1. Next, a quadrant of 16 tiles can be shown at a higher magnification, see Figs. 5.2-4. On visual inspection of the monitor of the video display system, abnormal tiles can be marked (cf. Figs. 5.2 and 5.4) by the reviewer using a mouse of the control device, and with the Zoom function the tiles can be further magnified. Finally, the x , y coordinates of an image field on the smear can be displayed by the system aiding the needed additional light-microscopical inspection (see Figs. 5.2 and 5.4). Sixteen marked tiles, out of the collection of 128 classified tiles, can be brought together on the screen, facilitating mental imaging of the lesion. The effect of this can be seen in the example of Fig. 5.3.

Prior to us, the system was tested in the USA by Koss et al. (1992), in Japan by Ishizuka et al. (1992), and in Israel by Malberger et al. (1992).

4 DEVISING A WORKING PROTOCOL FOR PAPNET-ASSISTED SCREENING

The diagnostician can make the following three decisions when using the PAPNET system in primary screening: to give a final diagnosis based on visual inspection of the 128 tiles, to inspect the smear by light microscopy using the x, y coordinates of tiles, or to rescreen the smear partly or completely. We used 2000 smears from our archive, including 500 abnormal smears with a known follow-up. Our purpose was to devise a decision tree. In order to do so, we looked closely at the performance of PAPNET in various cases by checking the tiles and the matching smear.

We learned that infectious agents were displayed by PAPNET, including Actinomycetes, Entamoeba gingivalis, Candida, Gardnerella, Leptothrix, Oxyuris eggs, and Trichomonas. In addition, cells with changes due to herpes virus, HPV, and Chlamydia were present in the tiles. In all these cases, supplementary light microscopy was needed to reach the correct diagnosis.

In cases of mild and moderate dysplasia, the abnormal cells were predominantly found on page 1. In cases of severe dysplasia and carcinoma in situ, the most important page was page 2. Particularly for page 2, light microscopy was needed for the dense cancer clusters, thus we included this procedure in our protocol.

In the tested cases of invasive squamous cell carcinoma, single cancer cells and cancerous epithelial fragments were classified by PAPNET. The number of tiles displaying cancer cells was high, 15 to 54 on page 1 and 28 to 49 on page 2, so again the classification of epithelial fragments on page 2 proved to be very important. Also in these cancer cases, many otherwise unmarked tiles on page 2 contained diathesis and necrosis, so the feature of displaying necrosis proved to be of great help. An advantage of the PAPNET system is that the pieces of the puzzle on which the diagnostician bases his/her diagnosis, the detected abnormal cells dispersed over the smear, are 'brought together' in the tiles on the video screen. This feature facilitates the diagnosis of invasive carcinoma.

Also the adenocarcinomas of the cervix were predominantly found thanks to page 2. In cases in which PAPNET showed atypical cells on page 1, occasionally more severe changes were found in the same region of the smear. We included in our protocol that in these cases not only the marked tiles should be evaluated by light microscopy, but also the surrounding areas. Of the four tested adenocarcinomas of the endometrium, two had only one tile with adenocarcinoma cells while the smear contained many more adenocarcinoma clusters. The four cases of adenocarcinoma of the ovary had many carcinoma tiles: also the psammoma bodies were easily detected by PAPNET.

Because the marked tiles can be stored, we can analyze which tiles are marked. The decision to mark tiles on page 1 (single cells) is rather straightforward, but the interpretation of the epithelial fragments of page 2 is more difficult. Hence, for this page there is more marking variation between diagnosticians. Often, dense cell groupings cannot be classified *at all* on the video screen, thus light microscopy is a must here.

In some cases, PAPNET showed more than 20 tiles with blood, only leucocytes, or air bubble artefacts. In all of these cases, the smear had to be rescreened completely. Because we ask for a repeat smear when there are no endocervical cells present, we decided to rescreen all smears in which PAPNET did not have tiles with endocervical cells. In a few, we detected some endocervical cells by conventional rescreening.

The finally devised PAPNET protocol is shown in Fig. 5.5. The efficacy of the PAPNET method depends on the level of video training and experience of the reviewing cytologist (Rosenthal, 1992). In addition, familiarity with the use of the protocol of Fig. 5.5 proved to be of major importance.

It is of interest to get an impression of the importance of turning to the microscope for the final diagnosis in cases where the reviewer notes atypical or CIN cells in the tiles. Therefore we did a test with two cytologists with 33 CIN smears. In all smears, the abnormal cells were clearly visualized on the video monitor. Hence, it was possible to classify the abnormality of these cells (this is in contrast with the four false-negative smears discussed in Sec. 5.8, and reported in Table 5.5). First, the cytologists registered their video diagnosis, and second they screened by light microscopy the x , y coordinates of the abnormal tiles and the surrounding areas. Their final PAPNET-assisted diagnosis was based on the most abnormal cells detected. It is clear that in this setup the final diagnosis is always the same or higher than the video diagnosis because if only cells with more pronounced abnormality are found in the second light-microscopical step of the procedure, the diagnosis is upgraded.

Table 1 Comparison PAPNET-video diagnosis and final PAPNET-assisted diagnosis.

Conventional diagnosis	<i>n</i>	Cytologist A		Cytologist B	
		= video	> video	= video	> video
CIN I	10	8	2	9	1
CIN II	6	5	1	5	1
CIN III	17	12	5	13	4
Total	33	25	8	27	6

The results of this test are shown in Table 5.1. For Cytologist A, the final diagnosis was higher in 8 cases. For Cytologist B, the diagnosis was higher in 6 cases. Consequently the two cytologists did not perform identically. Yet, 8 and 6 out of the 33 cases (roughly: one fifth) were underclassified in the first step of the procedure, respectively. In the cases with CIN I, with an upgraded final diagnosis, the tiles only showed cells that were interpreted as atypical but not as CIN: the clear CIN I cells were found in the surrounding area of the smear. In the similarly underclassified CIN II case, the tiles contained only CIN I cells. In the underclassified CIN III cases, the smear contained CIN III epithelial fragments not shown in the tiles but found in the same areas as the displayed CIN I or CIN II cells. These results show the importance of the light microscope in this *test*. We have similar experiences in the primary PAPNET-assisted screening.

Because the number of abnormal tiles can be in some cases quite small (particularly in cases of endometrial adenocarcinoma), and because the abnormality seen in the tiles can be less than that present in the smear, the reviewer must take enough time to look at each tile with great care and the protocol must be strictly obeyed.

5 PAPNET-ASSISTED SCREENING VERSUS CONVENTIONAL SCREENING

In 1992, we started to test in the Leiden Cytology and Pathology Laboratory the PAPNET-assisted primary screening and compare it with conventional screening. Preliminary results were given in Boon and Suurmeijer (1992).

In the period 1 August 1992 till 1 June 1993, 54 907 cervical smears made by general practitioners were sent to our laboratory. The computer selected at random from the daily load a certain fraction for PAPNET ($n = 22\,692$). The remaining 32 215 smears were conventionally screened. Thus the daily practice was kept as complete as possible to avoid the introduction of any bias.

The PAPNET reviewing in the laboratory was performed by seven cytotechnologists trained with the 2000 above-mentioned cases from our archive. The protocol of Fig. 5.5 was used for the 22 692 PAPNET cases.

The conventional screening protocol for the remaining 32 215 smears was as follows: only in the morning hours screening was performed. One cytotechnologist screened 40 cases in 4 hours. All infection cases with cellular changes (including HPV and Chlamydia-associated), smears without endocervical cells, smears with thick cell

Table 2 Screening results, conventional versus PAPNET.

	Conventional (n = 32 215)	PAPNET (n = 22 692)
Unsatisfactory	1.47%	1.64%
Negative	93.9 %	92.9 %
Atypia	2.89%	3.46%
CIN I-II	1.33%	1.47%
≥ CIN III	0.40%	0.53%

groups with or without blood, smears with atypia, Cervical Intraepithelial Neoplasia, or frank carcinoma cells, were reviewed by the chief cytotechnologist. This encompasses approximately 15% of the cases. Of these, one third is finally diagnosed by the pathologist.

The complete screening results are summarized in Table 5.2. Atypia and CIN I was diagnosed more frequently in the PAPNET series. The diagnosis of CIN III and more was 0.53% in the PAPNET series versus 0.40% in conventional screening, thus also here the pick-up rate was better.

The cytologic diagnoses should be confirmed by further histology. Usually, we report on the histologic results a year after the cytology results because it takes that long before all patients are biopsied and we can collect all necessary data. For this chapter we collected as many histology results as possible to answer the question: do we see an indication that PAPNET introduces overdiagnosing, thus do we find in the PAPNET series disproportionately few histologic abnormalities? In the PAPNET series, we found 14 carcinomata in situ, 6 invasive squamous cell carcinomas, and 3 adenocarcinomas (0.10). In the conventional series, there were 20 carcinomata in situ, 4 invasive squamous cell carcinomas, and 4 adenocarcinomas (0.08). Thus also for histology, PAPNET data were better (improvement of 20%). These preliminary data indicate that using PAPNET there was no cytologic overdiagnosing introduced.

The cytodiagnostician is also interested in detecting infectious agents and their effects on the cells. In the first period we used PAPNET, we underdiagnosed Fungus, Acinomnyces, and Trichomonas. We further trained our staff to detect these on the videoscreen, and as a result the PAPNET scores for these improved in the second period (see Table 5.3). Thus one can improve PAPNET performance once one knows what the problems are. For the other infections (HPV, Chlamydia and Herpes), PAPNET was superior in both periods. Note that in the second period, the scores for HPV was lower for both PAPNET and for the conventional screening: this is due to seasonal effects.

For the sake of completeness, we emphasize here that the success of PAPNET-assisted screening is 100% dependent on the skills of the diagnostician to select the correct tiles from the video display and to select the correct cases for further light microscopy.

Table 3 Infections, conventional versus PAPNET, in the first and second PAPNET period.

	First PAPNET Period		Second PAPNET Period	
	Convent. n = 5797	PAPNET n = 2971	Convent. n = 17 220	PAPNET n = 11 117
Infectious agents				
–Trichomonas	0.35%	0.27%	1.92%	1.31%
–Fungi	2.26%	0.91%	1.18%	0.70%
–Actinomyces	1.14%	0.44%	0.41%	0.33%
Cellular infectious changes				
–HPV	1.04%	1.82%	0.55%	0.67%
–Chlamydia	0.41%	0.74%	0.30%	0.41%
–Herpes	0.00%	0.07%	0.01%	0.03%

6 PRACTICAL IMPLICATIONS OF PAPNET-ASSISTED SCREENING

What are the practical implications of PAPNET-assisted screening for the laboratory? Let us first look at the proportion of cases in which the final diagnosis could *not* be made on the video screen, but needed supplementary light microscopy (See Fig. 5.5). The protocol resulted in 20–40% rescreening or screening of indicated areas of the smear. There was an important psychological factor involved: the more insecure persons needed the light microscope more often. Even if the case requires supplementary light microscopy, the time spent is much less than in conventional screening.

In our laboratory, we emphasize the importance of reporting to the clinician the status of the bacterial flora. Thus we want to report this in our PAPNET cases. We learned that the bacterial flora is optimally visible on page 1. By using the zoom option the Döderlein bacteriae can be distinguished on the video screen from coccoid overgrowth. We experienced that PAPNET is not the perfect screener for microorganisms such as Trichomonas, Candida, and Actinomyces, but we can improve on this. There are two possible solutions: first, the clinicians should indicate in which cases they expect these infections, thus the screener can rescreen areas of these slides to find microorganisms. Second, the PAPNET diagnostician should be alert to associated changes visible on the

video screen, such as halo's around the nuclei in Trichomonas infection and changed staining characteristics of the smear.

In using PAPNET, the infections with clear *cellular* changes, such as Herpes, HPV, and Chlamydia, are more often found. In a smear with just a few of these cells, they are displayed next to each other on the screen, which facilitates the diagnosis.

In using PAPNET, the laboratory can store the interesting cases on an optical disk. Thus, a large library of exemplary cases can be formed which is easily accessible. Also the process of decision making of the various echelons in the laboratory can be analyzed because the marking of the tiles and the steps of the protocol (Fig. 5.5) can be stored.

But what is the benefit of using PAPNET for routine diagnostic work? Let us first look at the practice of conventional screening. Here one looks while screening at moving images. Thus, the cytodiagnostician should be very alert. It is evident that it is more difficult and more tiring to identify a flying bird than when the bird is sitting on a branch. Accordingly, for conventional screening, only the morning hours, when alertness is at its peak, are used in our laboratory. PAPNET shows us still pictures: it places the birds on a branch for us. We found this much easier and less tiring. The likelihood of fatigue-related errors is therefore diminished. It is this aspect of PAPNET which we find most important. Second in importance is the fact that the abnormal cells, derived from different parts of the smear, are displayed side by side in the tiles. The third advantage is the shorter time needed: one cytotechnologist can diagnose 80 smears in the morning hours instead of 40, and can easily do some additional screening in the afternoon. Without any exception, the cytotechnologists preferred the PAPNET-assisted screening over conventional screening because it is intellectually much more rewarding.

7 PAPNET-ASSISTED RESCREENING FOR QUALITY CONTROL

In 1992, PAPNET is used in the USA not for primary screening but mainly for quality control. (For FDA regulation of computerized cytology devices, see Brindza 1991.) It is applied for rescreening of 'negative' cervical smears. The detection rate in rescreening (both conventional and computer-assisted) depends on the original rate of positive smears. In a low-risk population with few positive smears one can anticipate only a few detected false negatives. In addition, the gain of rescreening depends on the quality of the primary screening: in a laboratory with a poor screening

performance many positive smears can be hidden in the 'negatives'. In a well-run cytology laboratory, 5% is the minimum false-negative rate for CIN (Elgert et al., 1992).

In the Leiden Cytology and Pathology Laboratory, we rescreened by PAPNET 1530 smears originally classified as 'negative' in the conventional screening process. These smears were primarily screened in June and July 1992. In this period, in addition to the mentioned 1530 negative smears, 29 were signed out as positive (21 as CIN I-II, 7 as CIN III, and 1 as Carcinoma in Situ). We used for this rescreening process the protocol depicted in Fig. 5.5. The final light-microscopical diagnosis of the PAPNET cases was made by the same staff, so identical diagnostic criteria were applied for both the primary screening and the PAPNET-assisted rescreening. In these 1530 negative smears, 69 smears were reclassified as atypia, whereas in the primary screening only 47 were signed out as atypia. For the sake of clarity, we mention here that these atypia cases (in which there is no chromatin clumping but exclusively nuclear enlargement) are placed in the negative group and the Dutch practice is to ask of these women a repeat smear within one year. In addition, in the PAPNET-assisted rescreening 1 smear was reclassified as CIN I and 1 as CIN II. These dysplastic cells, not only with nuclear enlargement but in addition with hyperchromasia and chromatin clumping, were completely overlooked at the primary screening. Originally, there were 29 CIN cases detected in this relatively small series of 1559 smears, thus the gain of PAPNET-assisted rescreening for our laboratory was as low as 7%, very close to the ideal 5% of Elgert et al. (1992). Much larger series than the one reported here are needed for reliable statistical analysis of PAPNET-assisted rescreening of 'negative' smears.

8 PAPNET FOR DETECTION OF CANCER CELLS IN FALSE-NEGATIVE SMEARS

The false-negative group presents a true challenge to computer-assisted screening, because of its grave effects for both the patient and the laboratory. For instance, in The Netherlands, there is a substantial false-negative rate. In a nationwide survey, using the Dutch National Database for Pathology Laboratories, it was found that of the 690 cases of invasive cervical carcinoma diagnosed in 1990, 360 were known to have a previous Pap smear in the preceding two-year period. Of these 360 slides, 48 were signed out originally as 'negative' (13.3%). So per year we can expect 48 false-negative cases in The Netherlands (population 15 million). Part of the false-negative diagnoses is due to inadequacy of the smear. In these cases the sample does not contain abnormal cells. On the other hand, there are cases in which cancer cells can be detected once the

cancer diagnosis is known and the original smear is scrutinized again in detail. Here, the cytotechnologist has failed to detect the abnormal cells during the initial screening.

The PAPNET system was tested in the Leiden Cytology and Pathology Laboratory on 10 of our false-negative cases, six of them used in an earlier study in which the smears were recirculated in the laboratory (cf. Bosch et al., 1992). In all cases, abnormal cells or epithelial fragments were detected and classified by the PAPNET system (Table 5.4). In two cases (nrs. 5 and 10) the only marked tile was on page 1 (single cells). In case 10, the marked tile contained the only three abnormal cells present in the smear. In nine cases, the marked tiles were on page 2 (epithelial fragments). In one of these cases the smear contained only a single cluster of diagnostic cells, in another only two. The latter case is shown in Fig. 5.4. In eight cases, the marked tiles were exclusively located on page 2. The number of marked tiles on page 2 varied from 0 to 8. Note that in four cases (marked with a question mark) the quality of the images of the video screen were not good enough to warrant a video diagnosis.

Table 4 Video diagnosis and final PAPNET-assisted diagnoses of 10 false negatives of cancer patients.

Case Nr.	single cells	epithelial fragments	Video diagnosis	Final diagnosis
1	0	3	atypia/carcinoma	suspicious for ca.
2	0	3	poorly differentiated ca.	carcinoma
3	0	1	1 highly suspicious group	suspicious
4	0	3	carcinoma?	suspicious
5	1	6	carcinoma?	suspicious
6	0	8	carcinoma?	suspicious
7	0	5	carcinoma in situ	suspicious
8	0	4	suspicious	suspicious
9	0	3	suspicious	suspicious
10	2	0	carcinoma?	carcinoma

It was possible to make a final PAPNET-assisted diagnosis of carcinoma by light microscopy of the original smear in only two cases (nrs. 2 and 10). In all other cases the final diagnosis was no more than 'suspicious for carcinoma'. No definite light-microscopical cancer diagnosis was possible because (a) there were too few diagnostic cells, (b) the fixation was too poor, or (c) the cancer cells were exclusively present in very dense epithelial fragments lacking chromatin detail. In all cases, a biopsy was recommended by the pathologist. The PAPNET approach is strong in the Achilles heel of human screening: the finding of a few cancer cells in a smear, thus bringing these difficult cases to the attention of the pathologist.

Whether a premalignant lesion is missed by screening ought to be studied in a set up in which the time between the two screenings is the same for all patients. This can only be achieved in organized screening programs in which all women are invited for a smear test twice. We could use for such a study the smears of a screening program in which the interval between the two screenings was 3 years (total population 470 000). In this material, there were 65 cases in which the second smear showed CIN III. Of these smears, we obtained PAPNET video tiles and asked two diagnosticians to review the cases using the protocol described earlier. The results are shown in Table 5.5. All smears were of good quality, thus there were no ambiguous diagnoses of "suspicious" as was the case in the above-discussed series of missed carcinomas. Fifteen cases were signed out by both as negative, but in the remaining cases there were abnormal cells detected by at least one of the two diagnosticians. Five cases were signed out by both as severe dysplasia, six by both as carcinoma in situ (CIS), and 2 by both as adenocarcinoma in situ (AIS). Many smears contained very few abnormal cells explaining why they were overlooked in the initial screening. It was easier to reach the correct diagnosis using PAPNET because these few abnormal cells were collected by PAPNET on the video screen. In the cases of CIS, the lesion was of the reserve cell type and the smear did not contain dysplastic cells. But also here (and in the cases of AIS), the diagnosis was facilitated because PAPNET brought the fragments of cancerous epithelium together on the videoscreen.

Table 5 Sixtyfive false negatives from a (re)screening program, PAPNET-evaluated by two different cytologists, A and B.

<i>A</i> <i>B</i>	negative	atypia	mild dysplasia	moderate dysplasia	severe dysplasia	CIS	AIS
negative	15	1					
atypia	4	4	3	1			
mild dyspl.	2	3	7	5			
mod. dyspl.				4	1	1	
severe dyspl.					5		
CIS					1	6	
AIS							2

It should be stressed that for both studies of false-negative smears, the reviewers were not in a normal screening situation but in a test situation. We have shown that in the resulting state of increased awareness the likelihood to find the abnormal cells in these difficult smears is then increased (Bosch et al., 1992). Thus we acknowledge that these tests do not reflect the normal working situation and accordingly, the sensitivity of these tests cannot be directly compared with that of human cytotechnologists screening the same slides without any foreknowledge.

An intriguing approach for a PAPNET test would be to “seed” videopages of known abnormal PAPNET cases among the routine PAPNET cases. In our experience, however, such “seeding” is extraordinarily difficult without inducing foreknowledge. Anyhow, it is highly reassuring that PAPNET can detect abnormal cells in these difficult cases, thus the likelihood to miss one is certainly decreased. Thus it is not surprising that we find more positive cases in our primary PAPNET-assisted screening (see Table 5.2).

References

- Anderson TL, Nelson AC (1992) Automatic screening of conventional Papanicolaou smears. The AutoPap 300. *Analyst Quant Cytol Histol* 14:246-247.
- Bartoo GT (1992) Automated prescreening of conventionally prepared cervical smears. A feasibility study. *Lab Invest* 66:116.
- Bogdanich W (1987) The Pap test misses much cervical cancer through lab's errors. *Wall Street Journal*.
- Boon ME, Kok LP (1993) Neural network processing can provide means to catch errors that slip through human screening of Pap smears. *Diagn Cytopath* 9:416-416.
- Boon ME, Suurmeijer (1992) *The Pap Smear*. Coulomb Press Leyden, Leiden.
- Bosch MMC, Rietveld-Scheffers PEM, Boon ME (1992) Characteristics of false-negative smears tested in the normal screening situation. *Acta Cytol* 36:711-716.
- Brindza LJ (1991) FDA regulation of computerized cytology devices. *Anal Quant Cytol Histol* 13:3-6.
- Elgert P, Re E, Schreiber K, Koss LG (1992) Rate of false-negative cervical smears in a laboratory with extensive Quality control. *Analyst Quant Cytol Histol* 14:256-257.
- IAC Committee on Quantitative Morphology (1991) Data on automated cytology systems as submitted by their developers. *Analyst Quant Cytol Histol* 13:300-306.
- Ishizuka Y, Mango LJ (1992) Evaluation of the PAPNET automated cytology screening system utilizing routine, conventionally prepared cervical Papanicolaou smears at the Japan Cytology and Pathology Laboratory. *Analyst Quant Cytol Histol* 14:264-265.
- Koss LG, Lin E, Schreiber K, Mango L, Elgert P (1992) Evaluation of cervical smears by the PAPNET apparatus. *Analyst Quant Cytol Histol* 14:269.

Linder J (1992) Automation in Cytopathology. *Am J Clin Pathol* 98:S47-S51.

Malberger E (1992) Determination of the suitability of the PAPNET automated cytology screening system to rapidly implement widescale screening for cervical cancer in a population that is currently largely unscreened. *Analyst Quant Cytol Histol* 14:272.

Müller B, Reinhardt J (1990) *Neural Networks. An Introduction*. Springer, Berlin.

O'Leary TJ, Mikel UV, Becker RL (1992) Computer-assisted image interpretation. Use of a neural network to differentiate tubular carcinoma from sclerosing adenosis. *Mod Pathol* 5:402.

Rosenthal DL (1992) Evaluation of PAPNET automated cytology screening system. Sensitivity and its relationship to the level of video training and experience of the reviewing cytologist. *Anal Quant Cytol Histol* 14:280.

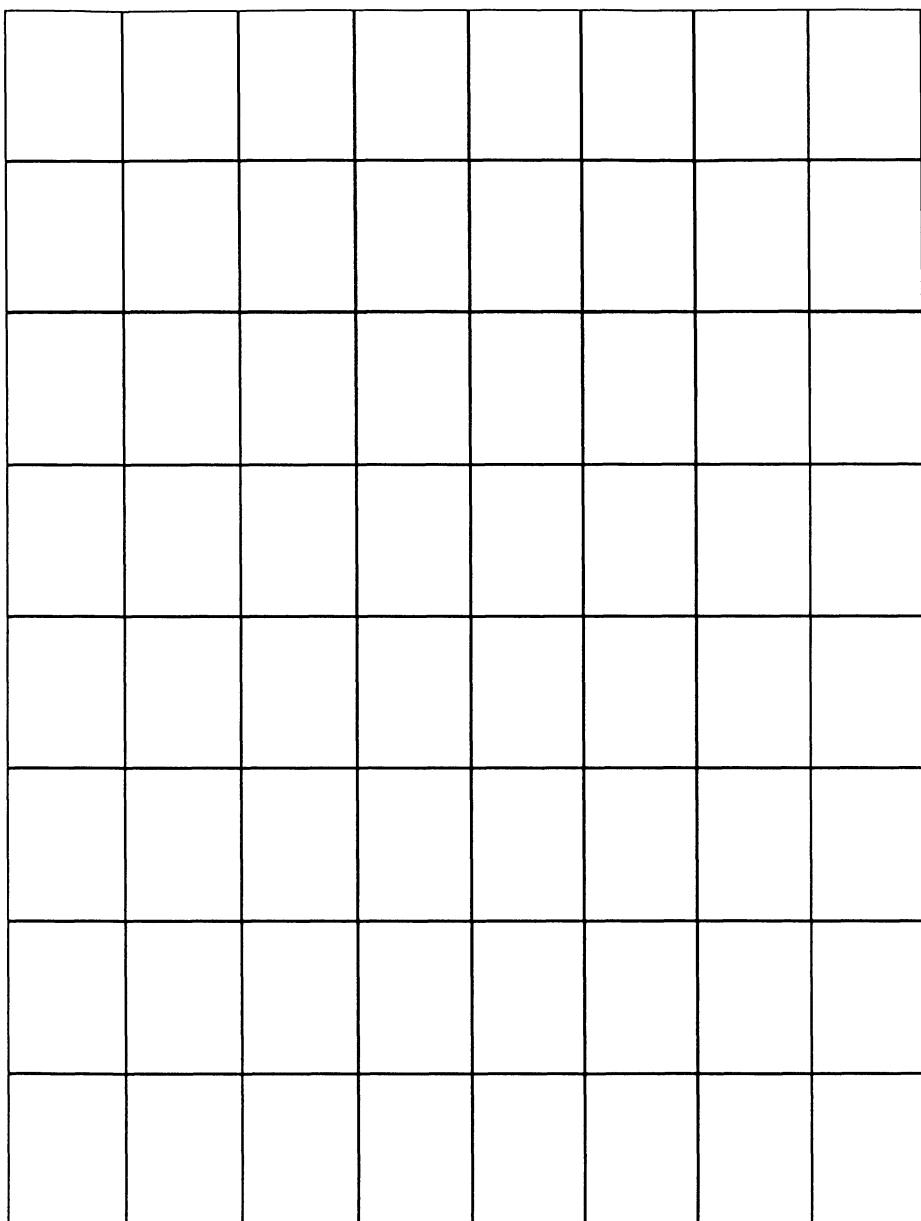


Figure 1 PAPNET video screen, page 1, 64 tiles. In almost every tile we can see an atypical or a dysplastic cell in this patient with moderate dysplasia and HPV. In the smear, these cells are located in different areas and are brought together on the videoscreen by PAPNET.

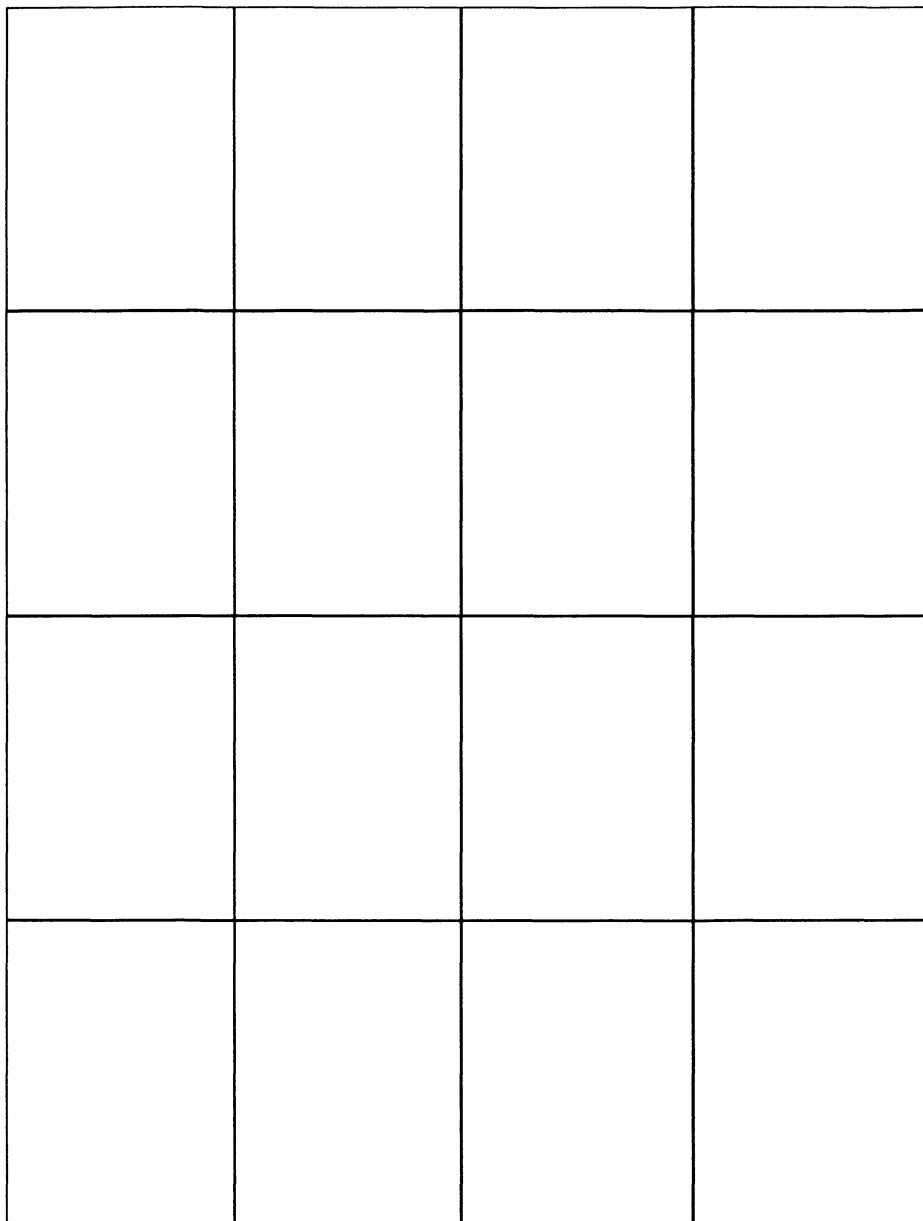


Figure 2 PAPNET video screen, shown sideways, page 1, quadrant 4, 16 tiles. The tiles contain benign reserve, cylindrical, and metaplastic cells. In addition, 4 tiles with benign squamous cells with perinuclear halo's. In $x = 95.6$, $y = 26.4$ a Trichomonad is visible (arrow).

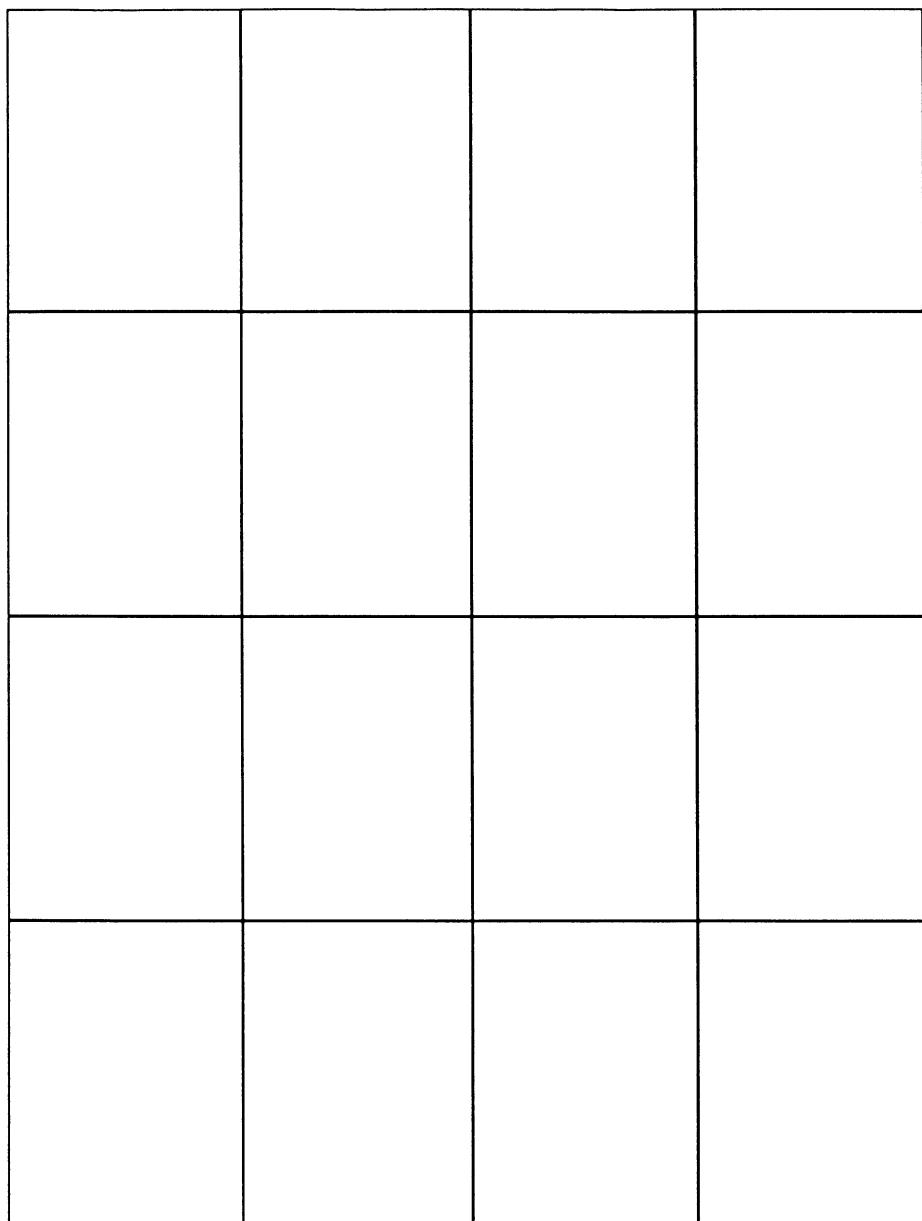


Figure 3 PAPNET videoscreen, shown sideways, page 1, quadrant 2, 16 tiles. In every tile we can see one or more carcinoma cells. In the smear, these cells are located in different areas and are brought together on the videoscreen by PAPNET. This is a smear from a patient with squamous cell carcinoma of the cervix.

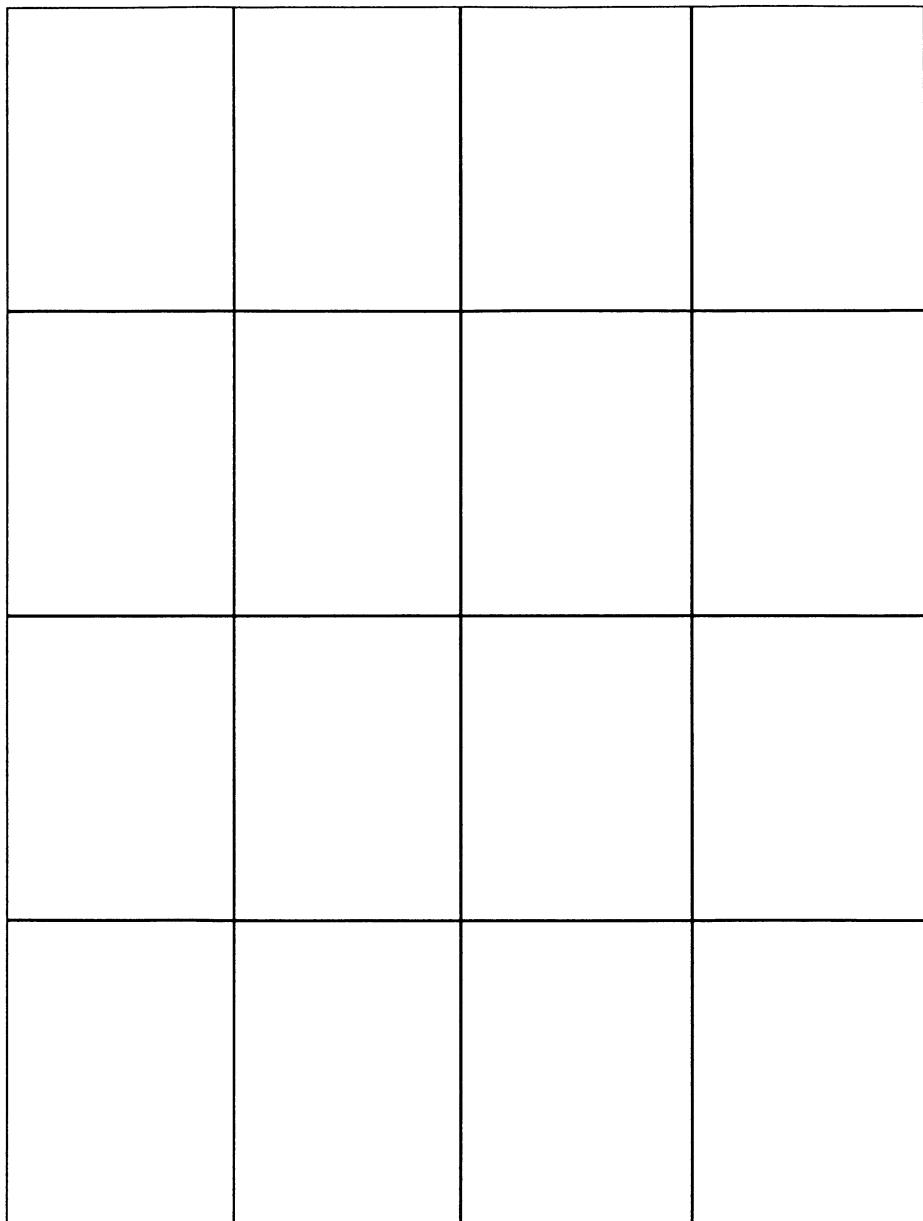


Figure 4 PAPNET video screen, shown sideways, page 2, 16 tiles. In this false-negative smear (see Sec. 5.8) PAPNET detected the two missed clusters of carcinoma cells ($x = 97.5$, $y = 16.8$ and $x = 97.5$, $y = 32.3$). Arrow: carcinoma cells with prominent nucleoli. Note the diathesis in the remaining tiles. Additional light microscopy of the areas indicated by the x, y coordinates is needed to evaluate these clusters properly.

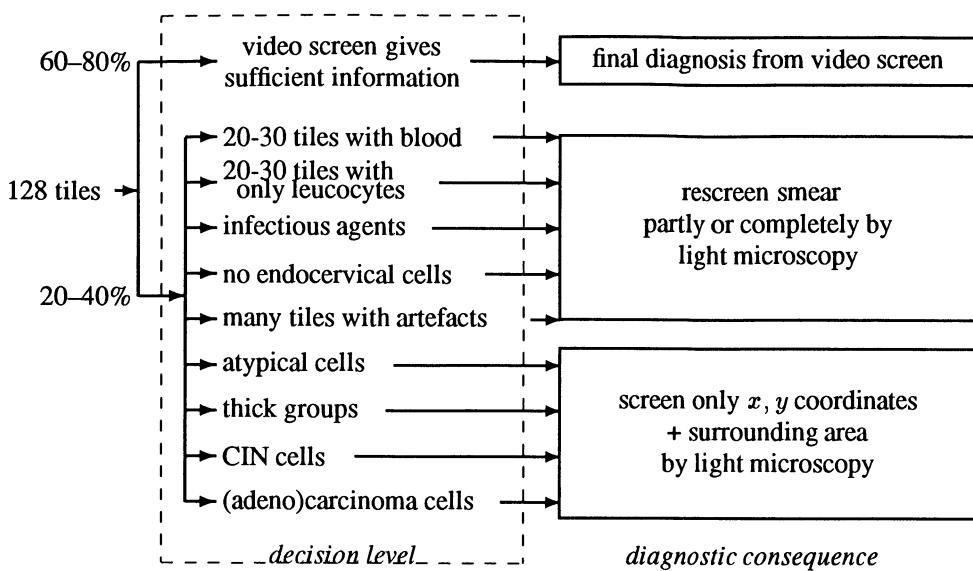


Figure 5 Leiden Protocol for PAPNET-assisted primary screening.

MULTIPHASE FLOW MONITORING IN OIL PIPELINES

Chris M. Bishop

*Neural Computing Research Group
Dept. of Computer Science and Applied Mathematics
Aston University
Birmingham B4 7ET, U.K.*

1 INTRODUCTION

Neural networks, and related statistical pattern recognition techniques, appear to be well suited to the solution of a wide range of monitoring and diagnostic problems. In many applications, it is difficult or impossible to perform first-principles modelling of the system under consideration. If, however, sufficiently large quantities of labelled training data can be made available, then a statistical approach becomes feasible.

In this paper we consider the non-invasive monitoring of oil flows along pipelines containing mixtures of oil, water and gas. This problem has arisen from use of multiphase pipelines to transport oil from offshore oil production platforms without the expense of offshore separation of the three phases. This in turn has led to the requirement for an accurate multiphase metering system for determining the oil fractions and flow rates for use in reservoir management and for custody transfer purposes.

Monitoring of multiphase flows is an extremely complex problem due to the large variety of geometrical configurations (stratified, annular, slug, etc.) which the phases can adopt. The approach described in this paper is based on the technique of gamma densitometry [1], which uses the attenuation of gamma beams passing through the pipe to gain information on the phase configurations. Each beam provides integrated data along its line of sight which is therefore only indirectly related to the volume fractions of the three phases.

A key problem, therefore, is to extract the phase fraction information from the gamma densitometer data. Computer modelling of the flows is exceedingly

difficult and unreliable, and in most practical situations is of little help. It is, however, possible to collect representative data on the flows by attaching the densitometer to a standard multiphase flow rig, and this suggests that a statistical approach, based on feedforward neural networks, would be worth investigating.

In Section 2 we give an overview of the problem of multiphase flow monitoring, and an introduction to the technique of dual-energy gamma densitometry. The neural network approach to the interpretation of densitometer data is then outlined in Section 4.

For the purposes of this study we have constructed synthetic data sets based on a representative sample of multiphase configurations, and the procedure for doing this is described in Section 3. The dominant source of noise in the densitometer data arises from the photon statistics, and this has been accurately modelled in an effort to ensure that the results will be representative of those obtained in the future using data from hardware rigs.

The problem of determining the volume fractions of oil, water and gas is then addressed in Section 5. We show that a non-linear neural network mapping, based on a multilayer perceptron, gives a level of accuracy which is more than sufficient for a practical monitoring system. In developing neural network applications it is important to provide some form of benchmark against which to assess the performance of the network solution. We shall therefore also consider a conventional statistical approach (the optimal linear mapping) to provide a suitable comparison.

The theoretical limitation in accuracy which can be achieved arises from the presence of noise on the densitometer data and this can be reduced in practice by increasing the integration time of the densitometer. We therefore study the effects of integration time on network performance, and we also give a general discussion of the rôle of noise on the input data during network training. Note that neural network techniques have also been used with great success to determine the phase configuration from densitometer data [1]. We shall not discuss this aspect here as it is of only indirect interest in determining the phase fractions.

Another key issue is that of validating the output from the trained network. We can regard the process of network training as being analogous to the problem of curve fitting, extended to multidimensional spaces. This suggests that the performance of the network may be much more robust when the network is 'interpolating' within the training data than when it is required to 'extrapolate'.

We give some theoretical justification for this heuristic viewpoint, and derive a quantitative measure of interpolation and extrapolation within the input space. This leads to a practical method for detecting when novel data are being presented to the network. In Section 7 we apply this technique to the problem of multiphase flow monitoring and show that it is indeed capable of detecting and rejecting novel configurations which could potentially give rise to significant output errors.

Finally, some conclusions are presented in Section 8.

2 GAMMA DENSITOMETRY AND MULTI-PHASE FLOWS

We begin with an overview of the technique of gamma densitometry and its application to the problem of monitoring multiphase flows. The aim here is simply to give sufficient information to make the paper self contained. Further detail can be found in ref. [1].

Gamma densitometry [2, 3] makes use of the attenuation of a beam of gamma rays passing through matter. The degree of attenuation is dependent on the path length within the material, the nature of the material and the wavelength of the gamma rays. Since, for a given substance, the fraction of the beam attenuated per unit length is constant, over a finite distance the beam intensity will fall exponentially. We therefore write the intensity of a gamma beam (of given wavelength) after passing through a length x of material in the form

$$I = I_0 e^{-\mu \rho x} \quad (1)$$

where ρ is the mass density of the material, μ is the mass absorption coefficient of the material at the given wavelength, and I_0 is the intensity of the gamma beam in the absence of material. Since, for a given material, μ and ρ can be measured separately, determination of I allows the length x of material between the gamma source and the detector to be determined.

A gamma beam passing through a multiphase pipeline will be attenuated by water, oil and gas each of which will have its own density and absorption coefficient. Figure 1 shows a schematic illustration of a collimated gamma beam passing through a circular cross-section pipe containing oil, water and gas in a stratified configuration. The intensity of the beam after passing through the

pipe will be given by

$$I = I_0 e^{-\mu_o \rho_o x_o} e^{-\mu_w \rho_w x_w} e^{-\mu_g \rho_g x_g} \quad (2)$$

where I_0 is now the beam intensity if there were no material in the pipe (i.e. a vacuum), and x_o , x_w and x_g are defined in Figure 1. Here we have introduced separate densities and mass absorption coefficients for each of the three phases. Note that even if the three phases are in a different geometrical configuration, for instance if they are homogeneously mixed, we can still apply (2) provided we interpret the x 's as the total effective path length through each phase.

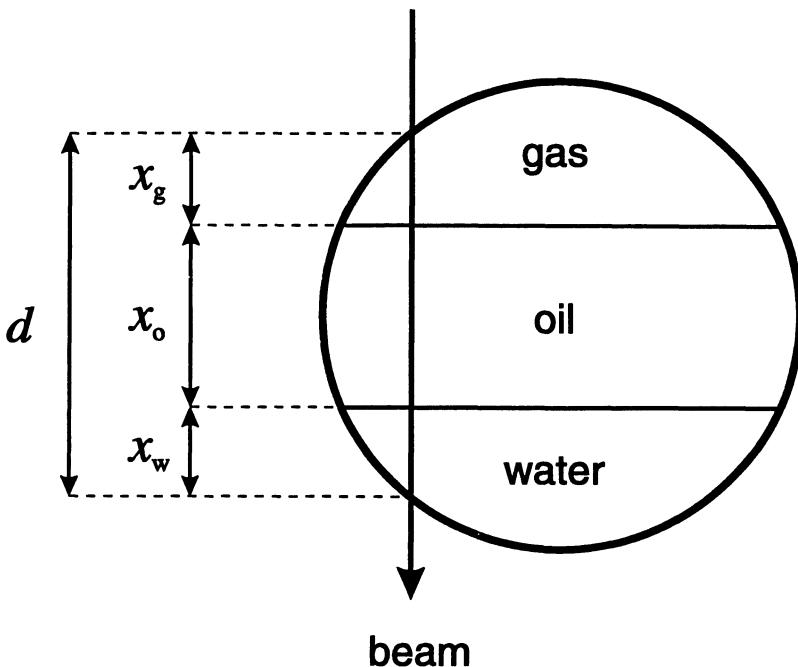


Figure 1 Schematic cross section of a pipe containing oil, water and gas in a stratified configuration, showing the path of a gamma beam together with the definitions of the path lengths x_o , x_w , x_g and d .

The basic problem is to determine the values of the three unknowns x_o , x_w and x_g . In dual-energy gamma densitometry this is achieved by sending a second gamma beam through the pipe along the same chord as the first beam. This second beam has a different wavelength, and so the absorption coefficients will have different values. For the second beam we have

$$I' = I'_0 e^{-\mu'_o \rho_o x_o} e^{-\mu'_w \rho_w x_w} e^{-\mu'_g \rho_g x_g} \quad (3)$$

It is convenient at this point to define the quantities

$$L = -\ln(I/I_0) \quad L' = -\ln(I'/I'_0) \quad (4)$$

We can then write (2) and (3) in the form

$$L = \mu_o \rho_o x_o + \mu_w \rho_w x_w + \mu_g \rho_g x_g \quad (5)$$

$$L' = \mu'_o \rho_o x_o + \mu'_w \rho_w x_w + \mu'_g \rho_g x_g \quad (6)$$

Since L and L' are measured from the gamma beam attenuation, and since the ρ 's and μ 's are known, (5) and (6) represent two equations in three unknowns. The third equation needed to determine the x 's comes from the geometrical constraint that the sum of the x 's must equal the total path length

$$x_o + x_w + x_g = d \quad (7)$$

as shown in Figure 1. We can therefore solve the three simultaneous algebraic equations (5), (6) and (7) to give the path length in oil in the form

$$x_o = \frac{\left\{ \frac{L - \mu_g \rho_g d}{\mu_w \rho_w - \mu_g \rho_g} - \frac{L' - \mu'_g \rho_g d}{\mu'_w \rho_w - \mu'_g \rho_g} \right\}}{\left\{ \frac{\mu_o \rho_o - \mu_g \rho_g}{\mu_w \rho_w - \mu_g \rho_g} - \frac{\mu'_o \rho_o - \mu'_g \rho_g}{\mu'_w \rho_w - \mu'_g \rho_g} \right\}} \quad (8)$$

A similar expression is obtained for x_w . The value of x_g can then be found from (7). Thus, dual-energy gamma densitometry allows us to determine the path length through each phase along the line-of-sight of the gamma beam. This, however, does not allow us to determine the oil fraction directly since the path lengths depend both on the oil fraction and on the geometrical configuration of the phases. Since many different phase configurations are possible with multiphase flows, we need further information.

The approach taken here is based on the use of multiple beam lines, each comprising a dual-energy gamma densitometer, to give additional information concerning the phase configuration. Throughout this paper we shall consider a particular system, employing six beam lines arranged as shown in Figure 2, which together provide 12 independent measurements giving information on the phase configuration.

In principle, one approach would be to determine the phase configuration from the densitometer measurements using tomographic reconstruction techniques

to obtain an image of the phase configuration directly. In practice such an approach cannot be adopted due to the very limited number of beam lines available. (Practical tomography systems generally make use of hundreds or thousands of independent lines of sight). Instead we shall adopt a statistical viewpoint and train a neural network to map the densitometer data directly onto the required phase fractions, without the need to determine the phase configuration as an intermediate step.

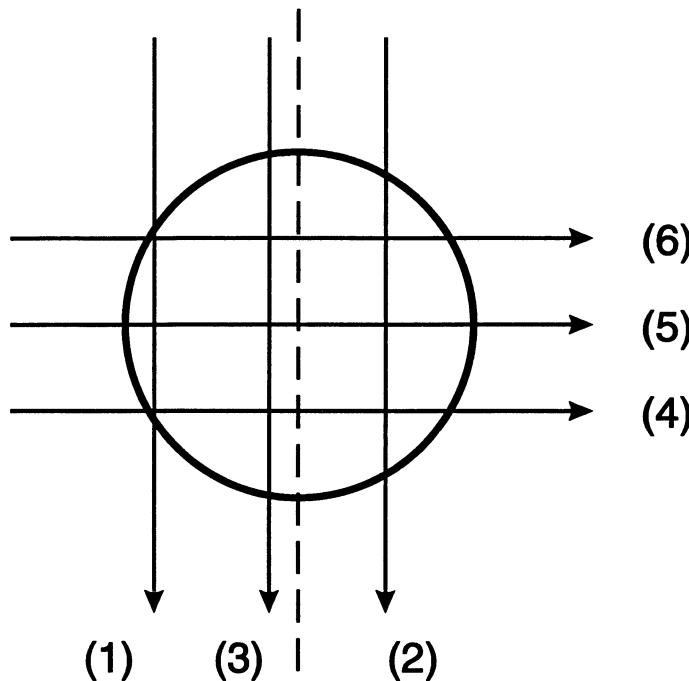


Figure 2 Cross section of the pipe showing the arrangement of the six beam lines. Each beam line comprises a single dual-energy gamma densitometer.

Note that we consider here only the problem of determining the phase volume fractions. In order to calculate flow rates it is also necessary to measure the flow velocities of the three phases, for which there exist several approaches. One of these again makes use of the gamma densitometer by correlating the signals from two beam lines displaced along the length of the pipe. If there are significant fluctuations in the flow, then correlation of the signals from the two densitometers can be used to determine the flow velocity [4, 5].

3 GENERATION OF DATASETS

For the purposes of this study we have used synthetic data sets generated in such a way as to model closely the kind of data to be expected from a hardware realisation of the gamma densitometer system. We have based the generation of data on the beam configuration shown in Figure 2. The dominant source of noise arises from photon statistics (except at very long integration times), and this too has been accurately modelled.

Multiphase flows along pipes can lead to many different geometrical configurations of the phase fractions. Due to the complexity of modelling multiphase flows this is the most difficult part of the problem to simulate. For the purposes of this study, however, it is sufficient to consider a small number of example configurations which are representative of the kinds of flows which might be expected in practice. We have considered four different configurations as shown in Figure 3. These are not intended to be accurate representations of true flows, but have been chosen to allow for straightforward analysis. Stratified, annular and homogeneous configurations occur in practice, and ‘inverse annular’ provides a fourth configuration for which the path lengths can easily be calculated.

For each of these configurations we have allowed all possible values of the oil and water fractions. Data sets have been generated using the following procedure

1. Choose one of the four phase configurations at random with equal probability.
2. Choose three random numbers f_1 , f_2 and f_3 selected uniformly in the interval $(0, 1)$, and set

$$f_o = \frac{f_1}{f_1 + f_2 + f_3} \quad f_w = \frac{f_2}{f_1 + f_2 + f_3} \quad (9)$$

This procedure treats each of the three phases on an equal footing and ensures that

$$f_o + f_w + f_g = 1 \quad (10)$$

3. For each of the six beam lines, calculate the effective path lengths through oil and water for the given configuration and phase fractions. The process of calculating the path lengths for a given phase configuration and phase fractions is a purely geometrical procedure and will not be described further. Details can be found in ref. [1].

- Perturb the path lengths to allow for the effect of photon statistics.

Each data set entry contains the following information: 12 values of path length (one in oil and one in water for each of the 6 beam lines), f_o and f_w . The path lengths in oil and water will form inputs to the network and the oil and water fractions will form the targets for network training. Note that the path lengths in gas would represent redundant information, as a result of the constraint (7), and so are not included in the inputs. Similarly, the gas phase fraction would represent a redundant output because of the relation (10), and so can also be excluded.

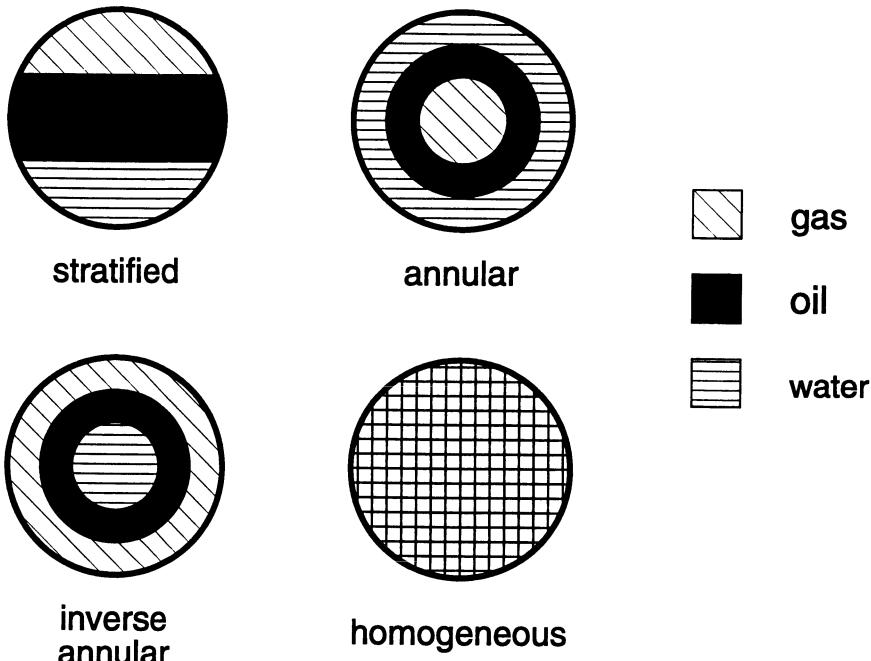


Figure 3 The four model phase configurations used in this study.

The effect of photon statistics is included as follows. If the expected number of counts on a particular beam during a given time interval is $\langle n \rangle$, then the actual number n observed on any particular measurement will be governed by a Poisson distribution \mathcal{P} , so that

$$n = \mathcal{P}(\langle n \rangle) \quad (11)$$

For each data point we begin by calculating the expected beam intensities using (2) and (3). The intensities I_0 and I'_0 are calculated by requiring that, for each

beam line, the count rate should be 60,000 counts sec.⁻¹ when the pipe contains only gas. This sets the maximum count rate to 60,000 for each of the detectors and thereby ensures that they are operating in a range where the count rates are as high as possible without saturation (since 60,000 cps is a typical upper limit for the scintillation counters used in the gamma densitometers). Thus we have

$$I_0 e^{-\mu_g \rho_g d} = 60,000 \quad (12)$$

$$I'_0 e^{-\mu'_g \rho'_g d} = 60,000 \quad (13)$$

The expected (i.e. average) number of photons is obtained by integrating the intensity over a time interval Δt

$$\langle n \rangle = I \Delta t \quad \langle n' \rangle = I' \Delta t \quad (14)$$

We then replace $\langle n \rangle$ by n using the Poisson distribution (11) to give the observed number of photons for the particular measurement (and similarly for n'). We then retrace our steps and calculate the path length in oil, using (4) and (8), together with the relations

$$I = n / \Delta t \quad I' = n' / \Delta t \quad (15)$$

A similar procedure is used to calculate the path length in water.

The integration time Δt determines the noise level on the measurements. If Δt is small the number of photons detected will be small and will therefore be subject to large fluctuations. Conversely for long integration times the error due to photon statistics will be small. In practice, Δt would be chosen as a compromise between minimizing noise levels and minimizing the time needed to take a measurement. We shall return to the issue of noise on the input data again in Section 6.

Note that, for each configuration, all possible values of f_o and f_w have been allowed. In practice we would anticipate significant correlations between the phase fractions and the phase configuration, which would be expected to simplify the pattern recognition problem. In this sense the data set used in this study may be more challenging than data obtained from an actual experiment. However, this may be offset to some extent by the fact that in practice there may be more than four possible phase configurations.

4 THE NEURAL NETWORK APPROACH

The basic problem to be solved is to find a mapping from the 12 path lengths measured by the densitometer beams to the volume fractions of the oil and water phases. The gas volume fraction is then trivially obtained from (10). The particular form of this mapping is to be determined from a data set of labelled examples. There are many procedures for constructing such mappings, including feedforward neural networks, and it is important to select an appropriate technique.

It is easy to see that the mapping must be non-linear if we consider the dependence of, say, the path length in oil on the oil volume fraction, for one of the configurations in Figure 3 (except for the homogeneous case for which the dependence is linear). However, we also expect the mapping to be smooth and well behaved, and this suggests that the multilayer perceptron (MLP) would be well suited to this problem. The MLP can generate a linear mapping as a special case (by suitable scaling of the weight and bias values) and can easily represent smooth departures from linearity. We shall consider MLP's with a single layer of hidden units governed by sigmoidal activation functions, so that the mapping from the input vector \mathbf{x} to the output vector \mathbf{y} is described by

$$\mathbf{y} = \mathbf{W}^{(2)} \cdot g(\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} \quad (16)$$

where $g(a)$ is the logistic sigmoid function given by

$$g(a) \equiv \frac{1}{1 + \exp(-a)} \quad (17)$$

and $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ represent the weights in the first and second layers of the network respectively, and $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ represent the corresponding biases. Such networks can approximate, to arbitrary accuracy, any continuous mapping from a finite domain, given a sufficient number of hidden units. Note that the output units are linear, since the use of invertible non-linearities on the output units would serve no useful purpose in this particular context.

In order to provide a benchmark for assessment of the performance of the MLP, we shall compare the results with those obtained from the optimal linear mapping, described by a function of the form

$$\mathbf{y} = \mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)} \quad (18)$$

which represents a linear relation between input and output variables.

Both the network and the linear mapping are trained by minimization of a sum-of-squares error function given by

$$E(\mathbf{w}) = \frac{1}{2} \sum_{q=1}^n \sum_{j=1}^m (y_j(\mathbf{x}^q; \mathbf{w}) - t_j^q)^2 \quad (19)$$

where q labels the particular training pattern, $y_j(\mathbf{x}^q; \mathbf{w})$ denotes the response of output unit j when the network is presented with input pattern \mathbf{x}^q , t_j^q is the corresponding target value for that unit, and \mathbf{w} represents the set of weights and biases in the network.

The training of the neural network corresponds to the minimization of $E(\mathbf{w})$ with respect to the weights and biases \mathbf{w} . This represents a non-linear multidimensional optimization problem. A particular feature of multilayer feedforward networks is that the derivatives of a function of the output units (such as the sum-of-squares error) with respect to the weights and biases can be obtained by an efficient procedure known as error backpropagation. The algorithm is efficient because the derivatives can be evaluated in $\mathcal{O}(\mathcal{N})$ steps (where \mathcal{N} is the total number of parameters in the network) rather than the $\mathcal{O}(\mathcal{N}^2)$ steps which would be needed in a naive approach [6, 7]. These derivatives can then be used in any of the standard non-linear optimization procedures such as conjugate gradients or quasi-Newton methods. We have chosen a technique known as the BFGS (Broyden-Fletcher-Goldfarb-Shanno) limited memory quasi-Newton algorithm which we have found to be very effective for a wide range of neural network problems. As with the method of conjugate gradients it involves successive line minimizations along carefully selected search directions. One ‘epoch’ of training involves \mathcal{N} line minimizations, and training of a network generally involves many complete epochs. A more detailed discussion of this and other training algorithms for neural networks can be found in ref. [7].

It is worth emphasising that the evaluation of the derivatives of an error function, and the use of those derivatives to minimize the error function, are two distinct processes. Multilayer perceptron networks were originally trained using simple, and inefficient, gradient descent optimization schemes, and the term backpropagation is sometimes used to describe this gradient descent approach. We prefer to reserve the term backpropagation for the procedure by which errors are propagated backwards through the network in order to evaluate the derivatives of a function. These derivatives can then be used in any of a variety of standard optimization algorithms. Backpropagation can also be applied to the evaluation of other derivatives such as the Jacobian of the network mapping and the Hessian matrix [7, 8].

For the linear mapping (18), the parameters can also be found using the BFGS algorithm, which is then guaranteed to converge in a single epoch. It should be noted, however, that the solution in this case can also be formulated in closed form in terms of the pseudo-inverse matrix [6, 9].

5 PREDICTION OF PHASE FRACTIONS

A training set and test set were generated, each containing 1,000 examples, but differing in the seed used for the random number generator (which determines the phase configuration and phase fractions). The integration time Δt was set to ten seconds, being typical of the value which might be used in an experimental set up. The effects of varying the integration time will be studied in Section 6.

The networks had 12 inputs corresponding to the 12 densitometer path length measurements, and two outputs, corresponding to the oil and water fractions, with target values for network training given by f_o and f_w . When presented with a new input vector, the trained network should then produce an estimate of f_o and f_w directly at the outputs.

Table 1: — Phase Fraction Results		
N_{hidden}	$E^{\text{RMS}} \text{ (train)} \times 10^2$	$E^{\text{RMS}} \text{ (test)} \times 10^2$
8	1.68	1.79
Linear	2.72	2.60

Table 1 Results for neural network prediction of phase fractions for the network with the optimal number of hidden units, and for the optimal linear map.

Networks having various numbers of hidden units were trained, each for 300 complete epochs of the BFGS optimization algorithm. To assess the results from the trained networks we evaluate a root-mean-square error defined by

$$E^{\text{RMS}} = \left\{ \frac{\sum_{q=1}^n |\mathbf{y}(\mathbf{x}^q) - \mathbf{t}^q|^2}{mn} \right\}^{1/2} \quad (20)$$

where n is the total number of patterns and m is the total number of output units.

The RMS error on both training and test sets for the trained neural networks are shown as a function of the number of hidden units in Figure 4. This graph shows a typical trend in which the training error decreases steadily as the number of hidden units, and hence the number of parameters in the network is increased. The test error, however, decreases at first as the greater flexibility in the network allows more complex functions to be generated, and then increases again when there are too many degrees of freedom in the network. The minimum test set error occurs for 8 hidden units as shown by the arrow in Figure 4. The errors both for this network and for the linear mapping are summarized in Table 1.

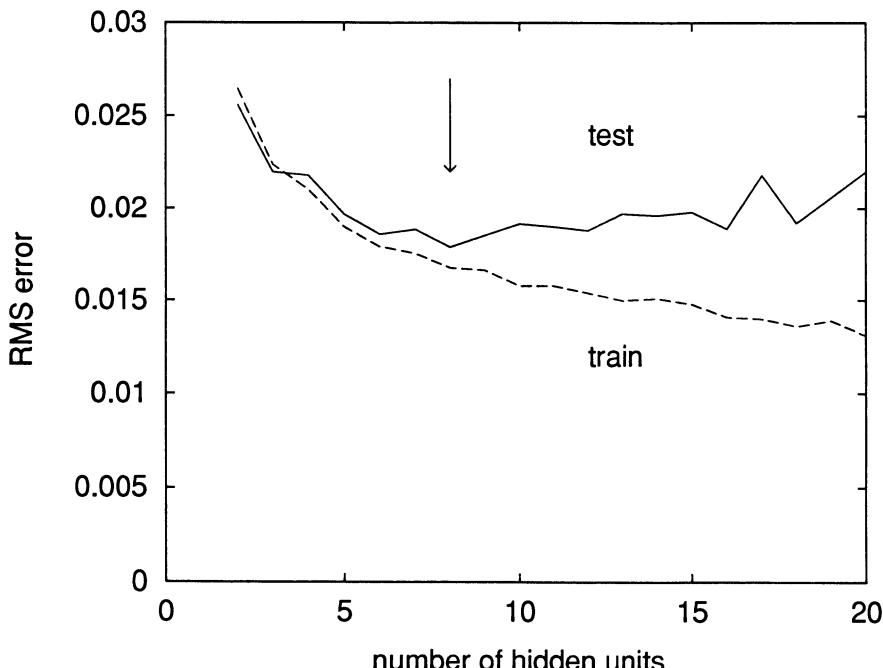


Figure 4 Plot of the RMS error produced by the neural network mapping for both the training and test sets as a function of the number of hidden units. The arrow indicates the minimum in the test error which occurs for the network having 8 hidden units, which is therefore selected as the best network.

Figure 5 shows a scatter plot of the oil fraction predicted by the network with 8 hidden units versus the actual oil fraction, for all of the points in the test set. The corresponding plot for the linear mapping is shown in Figure 6. We see that the neural network offers a significant improvement in performance over the linear mapping, and indeed is able to extract the phase fractions to a

sufficient accuracy to be used in a practical system.

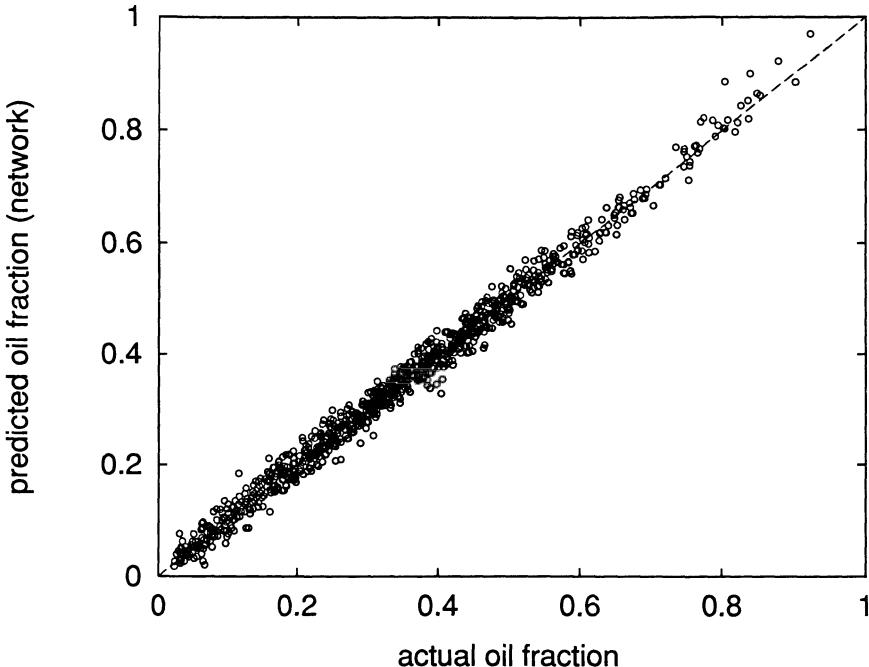


Figure 5 Scatter plot of the predicted oil fraction from the neural network versus the actual oil fraction, for all points in the test set.

Similar results are obtained for the water fraction f_w .

6 EFFECTS OF NOISE ON INPUTS

As explained in Section 3, the integration time Δt determines the noise level on the input data arising from photon statistics. For short integration times, we expect the performance of the network to deteriorate due to the increased noise level. In order to provide some indication of the integration times which would be needed in a practical system we have investigated the dependence of E^{RMS} on Δt .

We have generated several data sets corresponding to various values of Δt , each with 1,000 entries in both training and test versions (which differ only in the seed for the random number generator). In principle, the network archi-

ture should be optimized for each training set. However, in order to limit the computational demands to a tractable level we have opted to perform all training runs with networks having 8 hidden units since this was found to be the optimum number when Δt was set to ten seconds. The results are plotted in Figure 7 which gives an indication of the integration time which might be needed to achieve a given level of accuracy.

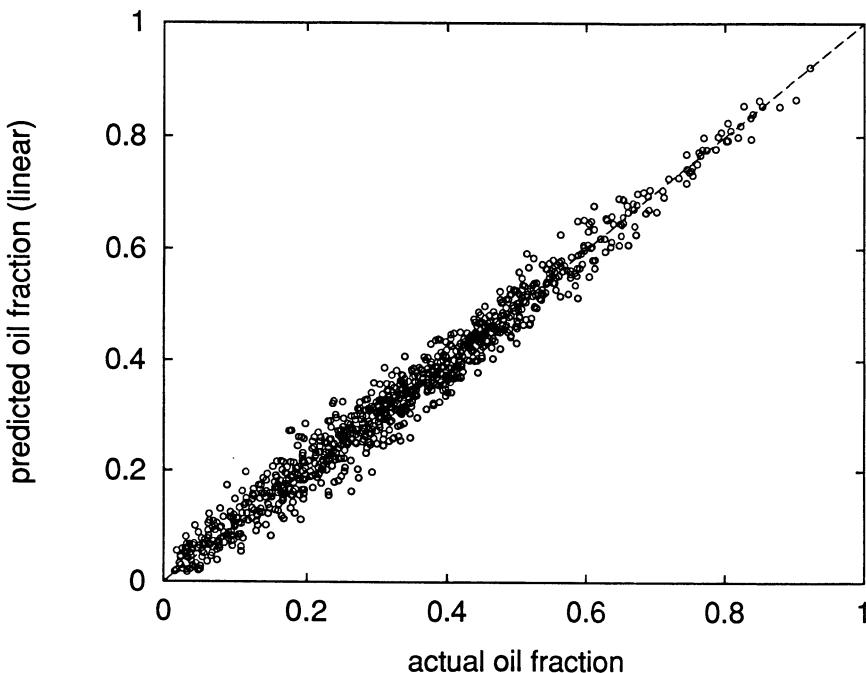


Figure 6 Scatter plot of the predicted oil fraction from the optimal linear mapping versus the actual oil fraction, for all points in the test set.

As expected, the error increases on both training and test sets as the integration time is reduced (for times below 100 seconds). It is interesting to note that the test RMS error actually increases with increasing Δt for values of Δt greater than about 100 seconds. This is interpreted as follows. The network architecture was optimized for a value of Δt of 10 seconds. For much larger values of Δt the data is much smoother and the optimum number of hidden units may be smaller than 8. In this case there will be some tendency towards overfitting, leading to a decrease in the training set error but an increase in the test set error. We have not, however, investigated this effect in detail. It is clear from Figure 7 that for integration times of a few tens of seconds the effects

of photon statistics should not give rise to significant errors in the predicted phase fractions.

It might be supposed that an appropriate strategy would be to train the network using data collected with large integration times (since the network training only needs to be performed once) and then to use shorter integration times when the system is put to use. In fact such a strategy is likely to yield poor results since the new data would then have a substantially different distribution to the training data. Training the network with noisy data avoids the problem of the network being excessively sensitive to noise on new data.

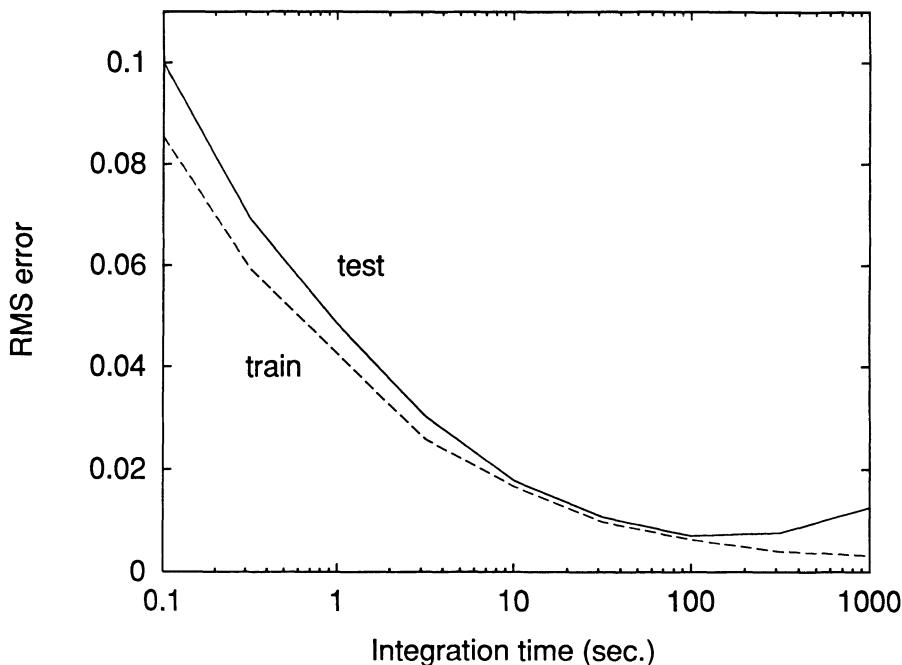


Figure 7 Plot of the RMS error from both training and test sets versus the integration time, for networks having 8 hidden units.

Some insight into the effects of random noise on the input data used to train a network can be obtained as follows [10, 11]. Consider the effect of adding a noise term ν to the input vector \mathbf{x}_p for pattern q , (note that for each presentation of the pattern a new value of ν is chosen). If the magnitude of the noise term is

small, we can Taylor expand the error function to give

$$E(\mathbf{x}^q + \boldsymbol{\nu}) = E(\mathbf{x}^q) + \sum_i \nu_i \left(\frac{\partial E}{\partial x_i} \right)_{\mathbf{x}^q} + \frac{1}{2} \sum_i \sum_{i'} \nu_i \nu_{i'} \left(\frac{\partial^2 E}{\partial x_i \partial x_{i'}} \right)_{\mathbf{x}^q} + \mathcal{O}(\nu^3) \quad (21)$$

where i and i' label the input units. Averaging over the input noise, and assuming $\langle \nu_i \rangle = 0$ and $\langle \nu_i \nu_{i'} \rangle = \eta \delta_{ii'}$, where η is a parameter related to the amplitude of the noise, we obtain

$$\langle E(\mathbf{x}^q + \boldsymbol{\nu}) \rangle = E(\mathbf{x}^q) + \frac{\eta}{2} \sum_i \left(\frac{\partial^2 E}{\partial x_i^2} \right)_{\mathbf{x}^q} \quad (22)$$

The total error measure is obtained by summing over all input patterns to give

$$\langle E \rangle = \sum_q E(\mathbf{x}^q) + \frac{\eta}{2} \sum_q \sum_i \left(\frac{\partial^2 E}{\partial x_i^2} \right)_{\mathbf{x}^q} \quad (23)$$

Thus, the minimization of an error measure in which noise is added to the inputs is equivalent to the minimization without noise of a modified error measure in which an extra term is added. This extra term depends on the second derivatives of the original error measure.

If we consider the particular case in which E is given by the sum-of-squares error

$$E(\mathbf{x}^q) = \frac{1}{2} \sum_j (y_j(\mathbf{x}^q) - t_j^q)^2 \quad (24)$$

then the additional term in the error measure becomes

$$\frac{\eta}{2} \sum_q \sum_i \sum_j \left\{ \left(\frac{\partial y_j^q}{\partial x_i^q} \right)^2 + (y_j^q - t_j^q) \frac{\partial^2 y_j^q}{\partial x_i^q \partial x_i^q} \right\} \quad (25)$$

This can be regarded as a *regularization* term. Such terms are widely used to impose prior knowledge (such as smoothness) in curve fitting and interpolation, and have recently been extended to both radial basis function and multilayer perceptron neural networks [9, 10]. They help to constrain the form of the network mapping and can thereby lead to improved generalization properties. The above analysis suggests that training with noise can also smooth the network mapping function and can thereby lead to improved generalization.

7 NOVELTY DETECTION AND NETWORK VALIDATION

As we have seen, the neural network approach to the analysis of data from multiple-beam dual-energy gamma densitometers relies on the provision of a suitable set of training data. When the trained neural network is applied in the field to new data it can be expected to perform satisfactorily provided the input data is similar to that used during training. If substantially novel data is presented to the network then it will be prone to significant errors. In practice, the availability of a controlled test rig should allow the collection of extensive data under a sufficiently wide range of conditions to ensure that the network performs satisfactorily in the field. Nevertheless, it is important to provide an operational system with some means of detecting novel input data in order that some degree of validation can be performed on the results generated by the network. Here we describe a relatively simple approach to the detection of novelty in the input data and show that is able to detect the occurrence of erroneous output signals.

Intuitively we expect a network to perform well when it is presented with data which is similar to that used during training, while we anticipate that it will give poor results when presented with substantially novel data. This intuitive picture can be given some theoretical support as follows. In the limit of an infinite training data set, it is straightforward to show [6, 7] that minimization of the sum-of-squares error (19) is equivalent to minimization of the error

$$\tilde{E} = \sum_j \int [y_j(\mathbf{x}; \mathbf{w}) - \langle t_j \mid \mathbf{x} \rangle]^2 p(\mathbf{x}) d\mathbf{x} \quad (26)$$

where $\langle t_j \mid \mathbf{x} \rangle$ is the conditional average of the target data and is defined by

$$\langle t_j \mid \mathbf{x} \rangle \equiv \int t_j p(t_j \mid \mathbf{x}) dt_j \quad (27)$$

Here $p(t_j \mid \mathbf{x})$ is the probability density of the target data, conditioned on the input vector \mathbf{x} . Provided the network has sufficient functional flexibility (which in practice means a sufficiently large number of hidden units) then the error (26) will be minimized when

$$y_j(\mathbf{x}; \mathbf{w}) = \langle t_j \mid \mathbf{x} \rangle \quad (28)$$

in other words when the network outputs represent the *regression* of the target data. In this sense the neural network solution can be regarded as optimal, since if the training data was generated from a deterministic function with

superimposed zero-mean noise, then the network will average over the noise and learn the underlying function. This indicates why the use of a sum-of-squares error can give good performance on interpolation problems.

The key point to note is that the error in (26) contains a factor of $p(\mathbf{x})$ which represents the *unconditional density* of the input data. For finite data sets, the extent to which (28) actually holds will therefore be weighted by this density. Thus the network outputs will be most reliable for those regions of input space where there is a lot of training data. We can therefore provide a quantitative measure of the degree of novelty of a new data point \mathbf{x}^q by first using the training data to find an estimate $\hat{p}(\mathbf{x})$ of the density $p(\mathbf{x})$, and then evaluating $\hat{p}(\mathbf{x}^q)$.

Suitable representations for $\hat{p}(\mathbf{x})$ can be found from standard techniques of non-parametric density estimation [12]. We have chosen a kernel-based approach with gaussian kernel functions in which the kernel function centres are given by the input vectors from the training set, so that

$$\hat{p}(\mathbf{x}) = \text{const.} \sum_{q=1}^n \exp \left\{ -\frac{|\mathbf{x} - \mathbf{x}^q|^2}{2\sigma^2} \right\} \quad (29)$$

where the parameter σ controls the degree of smoothness of the estimated density function. Its value must be neither too large (which would lead to a large bias in the estimate) nor too small (which would result in a large variance), and we have adopted the simple heuristic of setting σ to the average distance of the ten nearest neighbours, averaged over all points in the training data set.

In order to test the performance of this novelty detector we have generated a further data set, consisting of 1,000 examples with randomly chosen oil and water fractions, corresponding to a 5th configuration referred to as ‘inverted-stratified’ which is obtained by inverting the stratified configuration of Figure 3. This is not intended to represent a realistic 3-phase configuration but is chosen for computational simplicity. It suffices, however, to illustrate the detection of novel input data. As before, photon statistics were included, with an integration time of 10 seconds.

In this context, the density function $p(\mathbf{x})$ is generally called a likelihood, and in order to represent the results graphically, it is convenient to consider the logarithm of this quantity. Since ‘log’ is a monotonic function it does not affect the relative ordering of the likelihood values for different input vectors. Figure 8 shows a plot of the log likelihood versus the magnitude of the error between the oil fraction predicted by the neural network and the true value obtained from

the data set. Here the network with 8 hidden units described in Section 5 was used. The crosses show the 1,000 points from the test set (with an integration time of 10 seconds) as used to plot Figure 5. It is clear that for these data the network gives a small error (as we have already seen) and that the log likelihood is always larger than (say) -5 . The circles show the 1,000 samples corresponding to the inverted-stratified configuration. The majority of these points have log likelihood values which are substantially smaller than those of the test set points, and a correspondingly larger range of oil fraction errors. We see that the network can indeed generate poor results when presented with data from this new configuration. Such data points can, however, easily be rejected on the basis of their log likelihood values. Setting a threshold anywhere between -5 and -10 would reject all data points having significant phase fraction errors. In practice, a suitable value for the threshold can be determined by cross-validation with respect to an independent set of data such as the test set.

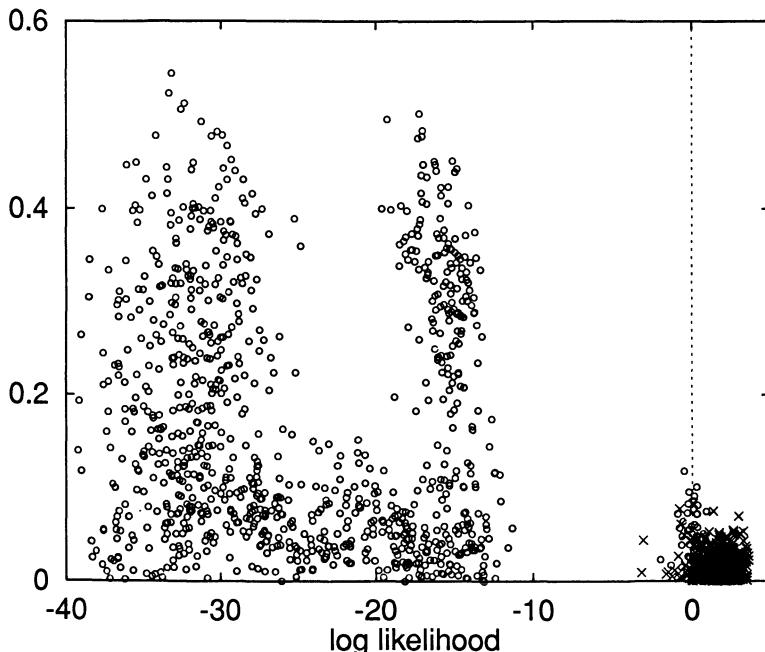


Figure 8 Plot of the log likelihood from the novelty detector versus absolute oil fraction error from the neural network. Crosses show data from the original test set, while circles correspond to data from a new 5th phase configuration.

It can also be seen from Figure 8 that there are some inverted-stratified points lying within the cluster of crosses. Examination of the phase fractions for these

points shows that they represent configurations which could also be classified as stratified configurations. For instance, if the oil phase fraction is sufficiently large then the three horizontal beam lines, shown in Figure 2, pass through oil only, in which case there are both stratified and inverted-stratified configurations having the same phase fractions which give rise to the same 12 path length measurements. The novelty detector 'correctly' interprets the inverted-stratified configurations as being similar to the training data, and indeed the network predicts the phase fraction to high accuracy.

In practice, we would expect the neural network to generate accurate phase fraction predictions under most circumstances. The rôle of the novelty detector is to prevent occasional novel configurations from generating spurious outputs. This ensures that the neural network can form the basis for a robust system which can be deployed in the field.

8 DISCUSSION

In this paper we have shown that neural network techniques, combined with dual-energy gamma densitometry, provide a powerful and accurate approach to the non-invasive analysis of multiphase flows. Although there is insufficient information to perform a tomographic reconstruction of the phase configuration, a neural network mapping can nevertheless extract the phase volume fractions to high accuracy.

The neural network approach described here relies on a set of characterized data for training the network. In a practical realisation this would be obtained using a calibration rig in which the phase fractions and fluid velocity could be controlled and measured. The need for any modelling of the multiphase flow, which would probably be complex and of limited accuracy, is thereby avoided.

One difficulty which can potentially arise in practice is the occurrence of phase configurations which were not present in the training set. Under such conditions the network could not be expected to generate reliable results. This problem can largely be circumvented by the provision of a suitably general training set spanning the full range of conditions likely to be encountered in practice. However, in order to ensure the robustness of a practical system, we have developed a technique for the detection of novelty in the input data and we have shown that it is able to signal the presence of new phase configurations which are not present in the training set. Such a novelty detector can be used

to validate the phase fractions generated by the neural network.

Although we have worked with the path length information in this study, the network could equally well take the photon count information directly as input data, thereby avoiding the need to perform the calculations normally required to extract the path lengths.

Acknowledgements

I would like to thank Dr G D James for his valuable contributions to work reported in this paper.

REFERENCES

- [1] C M Bishop and G D James (1993) Analysis of multiphase flows using dual-energy gamma densitometry and neural networks', Nuclear Instruments and Methods in Physics Research **A327** (1993) 580.
- [2] M S Abouelwafa and E J M Kendall, *J. Phys. E: Sci. Instrum.*, **13** (1980) 341.
- [3] K Rafa, T Tomoda and R Ridley Proc. Energy Sources Technology Conference and Exhibition, ASME (1989) 89-Pet-7.
- [4] M S Beck and A Plaskowski, *Cross Correlation Flowmeters*, (Adam Hilger, Bristol, 1987).
- [5] J S Watt, H W Zastawny, M D Rebgetz, P E Hartley and W K Ellis, Nuclear Techniques in the Exploration and Exploitation of Energy and Mineral Resources, (IAEA, Vienna, 1991) p. 481.
- [6] C M Bishop, invited review article: *Neural Networks and their Applications* to be published in *Reviews of Scientific Instruments* (1993).
- [7] C M Bishop, *Neural Networks for Pattern Recognition*, (Oxford University Press, 1994).
- [8] C M Bishop 'Exact calculation of the Hessian matrix for the multilayer perceptron' *Neural Computation*, **4** (1992) 494-501.
- [9] C M Bishop 'Improving the generalization properties of radial basis function neural networks' *Neural Computation* **3** (1991) 579-588.
- [10] C M Bishop 'Curvature Driven Smoothing: A Learning Algorithm for Feed-forward Networks', to be published in *IEEE Transactions on Neural Networks* (1993).
- [11] A R Webb (1991) 'Functional Approximation by Feedforward Networks: A Least-squares Approach to Generalisation' RSRE Memorandum 4453, R.S.R.E., St. Andrews Road, Malvern, Worcs, WR14 3PS, U.K.
- [12] R O Duda and P Hart, *Pattern Classification and Scene Analysis* (John Wiley, New York, 1973).

ELECTRICAL LOAD FORECASTING

Yuan-Yih Hsu and Chien-Chun Yang

*Department of Electrical Engineering
National Taiwan University,
Taipei, Taiwan*

ABSTRACT

Application of artificial neural networks (ANNs) to forecast the hourly loads of an electrical power system is examined in this chapter. Two types of ANN's, i.e., the Kohonen's self-organising feature maps and the feedforward multilayer neural networks, are employed for load forecasting. Kohonen's self-organising feature map, which is a kind of ANN with unsupervised learning scheme, is first used to identify those days with similar hourly load patterns. These days with similar load patterns are said to be of the same day type. The load pattern of the day under study is obtained by averaging the load patterns of several days in the past which are of the same day type as the given day. After the hourly load pattern has been reached, a multilayer feedforward neural network is designed to predict daily peak load and valley load. Once the peak load and valley load and the hourly load pattern are available, the desired hourly loads can be readily computed. The effectiveness of the proposed neural network is demonstrated by the short-term load forecasting of Taiwan power system.

1 INTRODUCTION

Electrical load forecasting is very important for power system operators and planers since many important functions in power system operational planning such as unit commitment and economic dispatch, maintenance scheduling, and expansion planning are usually performed based on the forecasted loads. Electrical load forecasting is usually divided into two major categories, long-term load forecasting which covers a period of one year to twenty years in the future and short-term load forecasting which covers a period of one day to one month in the near future. Long term load forecasting is useful for generation and transmission expansion planning. Since the time span is

long, certain factors which affect electrical load consumption most such as economic growth rate, population growth rate, . . . , etc. must be taken into account. In this chapter, only short term load forecasting is considered.

Usually, the hourly load for the next day or next week are to be forecasted in short term load forecasting. In this chapter, we consider the forecasting of power system load in the next 24 hours. The major factors which affect the hourly load in the next day include human social activities and weather variables such as temperature, humidity, wind speed and cloud cover[1-4].

In the literature, many statistical methods such as multiple regression[3-5], exponential smoothing[6], Box-Jenkins[7], and Kalman filter[8] have been proposed for short-term load forecasting. Usually, the statistical approaches are effective for the forecasting of normal days but fail to yield good results for those days with special events, e.g., the superbowl day. To take these heuristic rules into account, operators' experience and heuristic rules must be employed. This motivated the development of rule-based expert systems for load forecasting[9-12].

In recent years, artificial neural networks (ANNs) have been applied to forecast the hourly load of a power system[13-18]. Generally speaking, quite promising results have been reported in these works by using the ANN approach. A root-mean-square(RMS) error of less than 3% can be achieved. The hourly load of a power system can be directly obtained from the outputs of the ANNs. This approach is adopted in References[13-16]. An alternative approach employed in References[17-19] is to divide the entire load forecasting problem into two subproblems: the problem of determining the hourly load pattern and the problem of predicting daily peak load and valley load. The main reason for adopting the two-stage approach is based on the observation that the hourly load pattern is mainly affected by customers' social habits while the daily peak load and valley load are functions of weather variables such as temperatures and humidities. In this chapter, we will follow this two-stage approach.

In the next section, the load forecasting problem is briefly described. This is followed by an introduction to Kohonen's self-organising feature maps which will be employed in Section 8.4 to identify those days with similar load patterns and to determine the desired hourly load patterns. Then, we briefly describe the multilayer feedforward ANN in Section 8.5 and apply this ANN to predict the daily peak load and valley load in Section 8.6.

2 THE LOAD FORECASTING PROBLEM

Figure 8.1 depicts the hour loads of a typical day for Taiwan power system. Let the 24

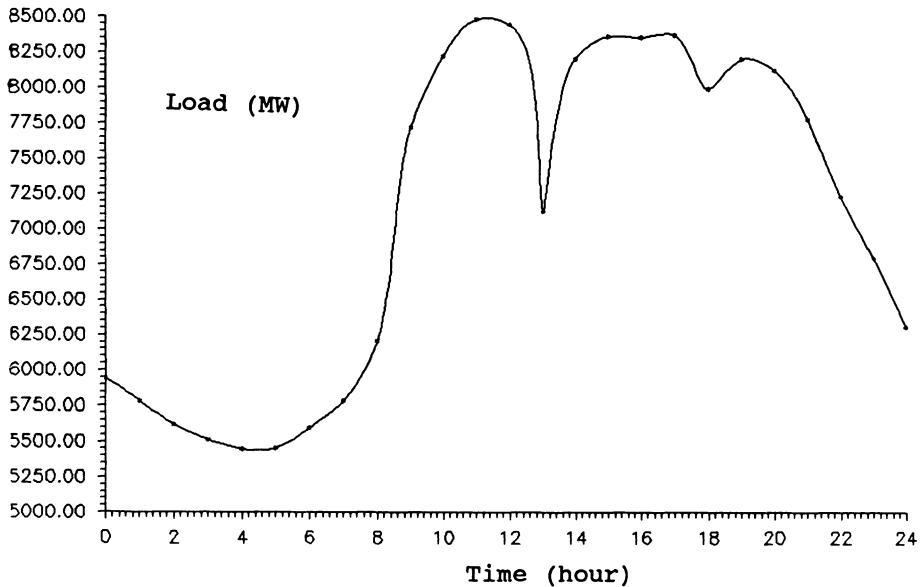


Figure 1 Daily hour load curve for Taiwan power system

hourly loads in a day be $L(i)$, $i = 1, \dots, 24$. To get the hourly load pattern of the day, we normalise these hourly loads by using the following equation

$$L_n(i) = \frac{L(i) - L_v}{L_p - L_v} \quad i = 1, 2, \dots, 24 \quad (1.1)$$

where $L_n(i)$ is the normalised load for hour i , $L(i)$ is the load for hour i , L_v is the valley load and L_p is the daily peak load. After the hourly load have been normalised, the normalised hourly load $L_n(i)$, $i = 1, \dots, 24$ will fall within the range from zero to one with daily valley and peak being normalised to zero and one, respectively, as show in Figure 8.2.

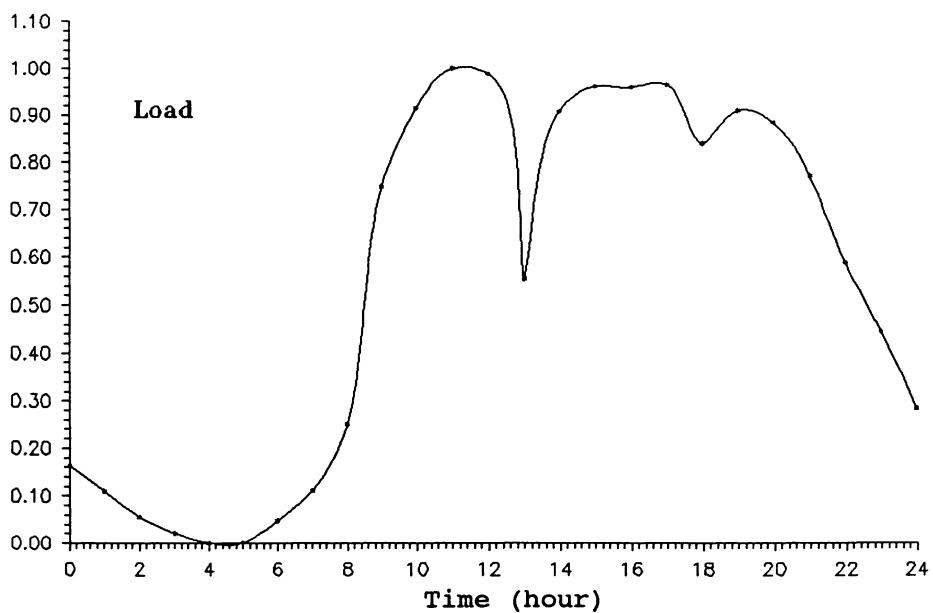


Figure 2 Normalised hourly loads

Define load pattern vector

$$X' = [x'_1, x'_2, \dots, x'_{24}]^T \quad (1.2)$$

$$= [x_n(1), x_n(1), \dots, x_n(24)]^T \quad (1.3)$$

It is observed that the elements of the load pattern vector, $x'_i, i = 1, 2, \dots, 24$, always fall within the interval (0,1) no matter how high or how low the daily peak and valley loads are. The load pattern vector provides us with the information how the system load changes with time in a day. Thus, the load pattern vector is mainly affected by customers' social habits. For example, the load pattern vectors for weekdays will be rather different from those for weekends. To be specific, the daily peak for a typical weekday may take place in the afternoon while that for weekends may occur in the evening.

It should be noted that, from eq.(8.1), the desired hourly loads $L(i)$ can be computed by using the equation

$$L(i) = L_v + (L_p - L_v)L_n(i) \quad (1.4)$$

Therefore, we need to figure out the load pattern $L_n(i), i = 1, 2, \dots, 24$, and the daily peak load L_p and valley load L_v in order to get the desired hourly loads. In other words, the load forecasting problem can be divided into two subproblem: the problem of load pattern computation and the problem of peak load and valley load prediction.

2.1 Computation of Load Patterns

To reach a proper load pattern for the day under study, the 24 hourly loads $L(i), i = 1, \dots, 24$, of each day in the five-year period are first compiled. These data are available from the database of the Taiwan Power Company. Then these hourly loads are normalised using eq.(8-1) and the resultant load pattern vectors are stored. By analyzing the load pattern vectors over the past five-year period, all days in a year can be divided into several groups, with each group comprising the days with similar load patterns. These days with similar patterns will be referred to as the days of the same day type. Once all the day types have been identified from the load data in the past, each day including the day under study can be assigned a day type. Therefore, the desired load pattern for the given day can be obtained by averaging the load patterns for several recent days of the same day type as the given day.

In Reference[11] it was mentioned that several important day types such as weekdays, Saturdays, Sundays, holidays and the Chinese New Year had been identified by the operators through their previous operational experience. Because the load pattern is a function of the social habits of customers, which change with time, new patterns

may emerge and some original patterns may lose their significance or even disappear. It takes time for the operators to notice these changes in load pattern. Therefore it is desirable to develop a tool to help the operators to identify new load patterns and update the present load pattern. A pattern recognition method has been employed in Reference [11] to serve this purpose. Indeed, some other day types such as Mourning Day have been identified through this approach.

In this chapter, an approach using artificial neural networks (ANNs) is proposed for the classification of hourly load patterns. For the problem of load pattern classification, we are not able to specify the output (day type) for an input pattern (hourly load pattern) during training because we want the neural net to yield potential 'new' day types and merge some existing day types which now have similar patterns. In this case, we do not know in advance how many solutions (types of day) there will be for the output units of the neural network. Thus a neural net with an unsupervised learning method is necessary. In this chapter, Kohonen's self-organizing feature map[20,21] as described in Section 8.3 will be employed to classify the 365 days in the year into several groups according to the hourly load pattern of each day, with each group consisting of those days with similar load patterns. Each such group is referred to as a day type. In Section 8.4, we will present the results from the application of self-organising feature maps to identify the day types of the Taiwan power system.

2.2 Prediction of peak load and valley load

It has been recognised that daily peak load and valley load are sensitive to weather variables such as temperature, humidity, wind speed and cloud cover[1-4]. In recent work conducted by the authors[11], it was pointed out that the available weather data at the Taiwan Power Company (TPC) included the daily high and low temperatures and the maximum and minimum values of humidity at three regions on the island of Taiwan. Extensive studies had been performed on the effect of these weather variables on daily peak load and valley load using a regression model before it was concluded that temperature was much more important than humidity as far as the peak load and valley load at TPC were concerned [11]. Therefore, the daily high and low temperatures will be employed as the weather variables for short-term load forecasting.

With the weather variables and load data in the past at hand, one can build a model to relate peak and valley loads to weather variables. This can be achieved by various approaches such as multiple regression[5], the Box-Jenkins method[7], the Kalman filter[8] and so on.

In the present work, we propose to use a multilayer feedforward ANN [22] to predict peak load and valley load. The basic algorithm for the multilayer feedforward ANN will be described in Section 8.5. In Section 8.6, the ANN will be applied to forecast the peak load and valley load of the Taiwan power system.

3 KOHONEN'S SELF-ORGANISING FEATURE MAPS

The self-organising feature map originated from the observation that the fine structure of the cerebral cortex in the brain is self-organised during training. It maps the N inputs x_1, x_2, \dots, x_N ($N = 24$ in our study) on to the M output nodes y_1, y_2, \dots, y_M ($M = 324$ in our study), as shown in Figure 8.3. The output nodes are arranged on a two-dimensional grid-like network. Each output node is described by a pair of rectangular co-ordinates. Note that the number of output nodes or the size of the grid (M) has been arbitrarily chosen to be $18 \times 18 (= 324)$. Other grid sizes can be employed, as long as computer memory capacity permits.

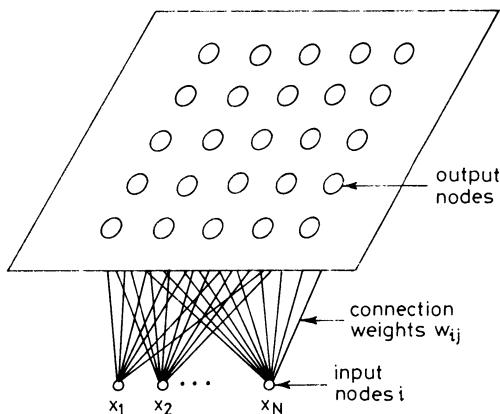


Figure 3 Structure of self-organising feature map

For our problem of load forecasting, the load pattern vector X' in eq.(8.2) is first normalised to obtain a vector X of unity length before it can be used as input vector to the self-organising feature map

$$X = [x_1, x_2, \dots, x_{24}]^T \quad (1.5)$$

where

$$x_i = \frac{x'_i}{\left(\sum_{i=1}^{24} X_i^2\right)^{1/2}} \quad i = 1, 2, \dots, 24$$

Given a number of input pattern vectors $X(1), X(2), \dots, X(P)$, our objective is to divide them into several clusters with each cluster comprising similar pattern vectors X .

As shown in Figure 8.3 each input unit i is connected to an output unit j through connection weight w_{ij} . The continuous-valued input patterns and connection weights will give the output node j a continuous-valued activation value a_j as follows

$$a_j = \sum_{i=1}^N w_{ij} x_i \quad (1.6)$$

$$= W_j X \quad (1.7)$$

where X is the input pattern vector as given in eq.(8.5) and $W_j = [w_{1j}, w_{2j}, \dots, w_{Nj}]^T$ is the connection weight vector for output node j . Note that $N = 24$ in our study.

When an input pattern is presented to the neural net without specifying the desired output, the neural net computes the activation value for each output node based on present connection weights. The input pattern is said to be mapped to the output node with maximum activation value. After enough input pattern vectors have been presented, input patterns with similar features will be mapped to the same output unit or to output units within a small neighbourhood. Because the clustering of input pattern vectors is self-organised in the learning process, and the ordering of the output node of an input pattern vector is based on that feature, this kind of neural network is called the self-organising feature map by Kohonen[20,21].

Figure 8.4 shows the flowchart of the proposed algorithm to produce the self-organising feature map. The algorithm begins with reading in the input pattern vectors $X(1), X(2), \dots, X(P)$ (block 1). It is assumed that we use P different input patterns in the training process.

The next step is to select the initial values for the connection weights w_{ij} ($i = 1, \dots, N; j = 1, \dots, M$) and the radius of the neighbourhood N_c . Kohonen suggested that the connection weights be initialised to small random values[20,21]. The approach works well for the cases where the input vectors are widely spread over the whole pattern space. But the input vectors in the present work are restricted to a small portion of the space. Therefore we propose to set the initial weight vectors to be around

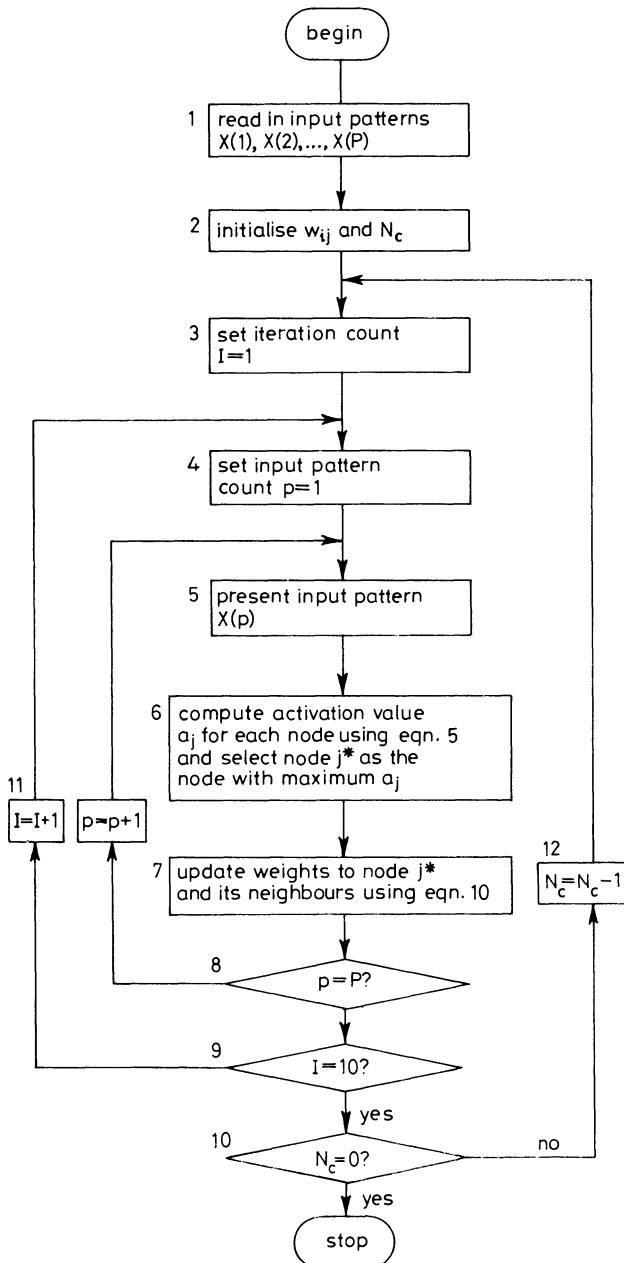


Figure 4 Flowchart for algorithm to produce self-organising maps

the mean of these vectors. In other words, we first define

$$W'_j = \text{mean of}(X(1), X(2), \dots, X(P)) \quad j = 1, \dots, M \quad (1.8)$$

or

$$w_{ij} = \text{mean of}(X_i(1), X_i(2), \dots, X_i(P)) \quad j = 1, \dots, N \quad (1.9)$$

Then perturb W'_j with a random noise as follows:

$$W''_j = W'_j + [5r \times \text{variance of}(X(1), X(2), \dots, X(P))] \quad (1.10)$$

where r is a random number uniformly distributed in the range from -0.5 to 0.5. Finally, normalise W''_j to give unity vector length

$$W_j = \frac{W''_j}{\left(\sum_{i=1}^{24} w_{ij}^2\right)^{1/2}} \quad (1.11)$$

The radius of neighbourhood N_c is useful in updating the weights(see block 7 of Figure 8.4). As shown in Figure 8.5, the neighbours of an output node j are defined to be the $(2N_c + 1) \times (2N_c + 1)$ output nodes within the square block, with node j being the centre of the block. For example, the neighbours for output node j with $N_c = 4$ include 81 nodes in the largest square block NE_j ($N_c = 4$) of Figure 8.5.

In this study, the radius of neighbourhood N_c is initialised to be 4. In the training process, N_c is decreased by 1 after ten iterations(see block 12 of Figure 8.4) until N_c equals zero. Here, by one iteration we mean the P input pattern vectors $X(1), X(2), \dots, X(P)$ have all been presented once. Detailed procedures within one iteration are described as follows.

After the initial connection weights have been specified, the activation value of each output node can be computed using eq.(8.6). The node j^* with the maximum activation value is picked up. This is what the training algorithm performs is block 6 of Figure 8.4.

In block 7, the connection weights for node j^* and all nodes in the neighbourhood defined by NE_{j^*} as shown in Figure 8.5 are updated using the following equation

$$w_{ij, \text{updated}} = w_{ij} + \eta(x_i - w_{ij}) \quad \text{for } i = 1, \dots, N, j \in NE_{j^*} \quad (1.12)$$

where η is the step size for the updating. Note that η can be fixed or variable throughout the iterations. A value of η from 0.003 to 0.008 has been found to give satisfactory results in this study.

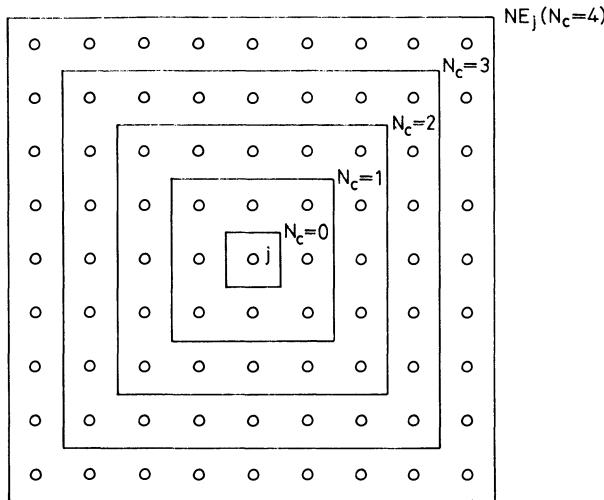


Figure 5 Neighbours for node j , $NE_j(N_c)$

The above procedures are repeated for each input pattern. When all P input patterns have been presented, we say that one iteration has been completed. Because there are ten iterations sharing one common radius of neighbourhood and we use five different values of N_c ($N_c = 4, 3, 2, 1, 0$) in the training process, there will be 50 iterations in all.

4 APPLICATION OF SELF-ORGANISING FEATURE MAPS TO DAY TYPE IDENTIFICATION

The developed self-organising feature maps have been applied to identify the day types of the Taiwan power system using the five-year hourly load data in the database of the Taiwan Power Company. Owing to limited space, only the results from some typical studies are given.

4.1 Example 1: Load Patterns in April 1987

The 30 load patterns in April 1987 are mapped to a two-dimensional output network with 324 nodes. Figures 8.6, 8.7, 8.8, and 8.9 depict the feature maps after 1, 5, 25 and 50 iterations of training, respectively. The step size η is decreased from 0.008 to 0.001 throughout the iterations. Table 8.1 summarises the results in Figures 8.6-8.9. To see how the step size η affects the resultant feature maps, Figures 8.10, 8.11 and 8.12 show the feature maps after 50 iterations of training for fixed step sizes of 0.002, 0.005 and 0.008, respectively. These maps are summarised in Table 8.2.

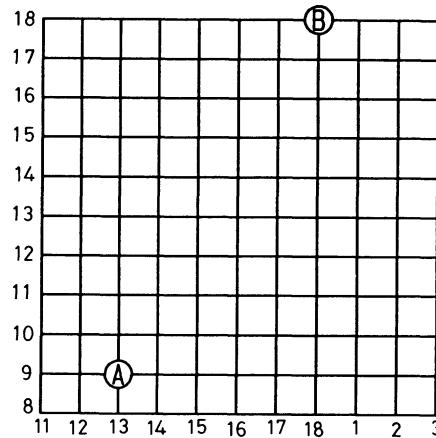


Figure 6 Feature map for load patterns in April 1987 (variable step size, 1 iteration)

1 iteration (Fig.8.6)		5 iterations (Fig.8.7)		25 iterations (Fig.8.8)		50 iterations (Fig.8.9)	
Output node	Date(day)	Output node	Date(day)	Output node	Date(day)	Output node	Date(day)
A(13,9)	5(SU)*	A(14,17)	19(ST)	A(1,18)	9(W), 8(F), 14(T)	A(1,18)	1(W), 2(TH), 3(F)
B(18,18)	Others	B(16,2)	8(SU), 12(SU), 36(SU)	B(2,1)	15(W), 16(TH), 17(F)	B(2,1)	14(T), 18(W), 16(TH)
		C(18,18)	Others	C(8,16)	7(T), 18(M), 20(M)	C(2,17)	6(M)
				D(14,1)	27(M)	D(3,16)	8(W), 9(TH), 10(F)
				D(12,9)	12(SU), 18(SU), 36(SU)	D(1,16)	21(T), 23(TH), 29(W)
				E(14,8)	8(SU)	E(14,1)	13(M), 15(M), 30(M)
				F(18,16)	4(S), 11(S), 18(S)	F(14,2)	27(M)
					25(S)	G(14,3)	12(SU)
				G(18,18)	Others	H(18,16)	8(SU)
						I(1,17)	11(S), 18(S), 25(S)
						J(17,13)	1(M)
						K(18,18)	30(TH)
							22(W), 24(F), 28(T)

*Abbreviations: SU, Sunday; M, Monday; T, Tuesday; W, Wednesday; TH, Thursday; F, Friday; S, Saturday

Table 1 Summary of the feature map for April 1987 (variable step size)

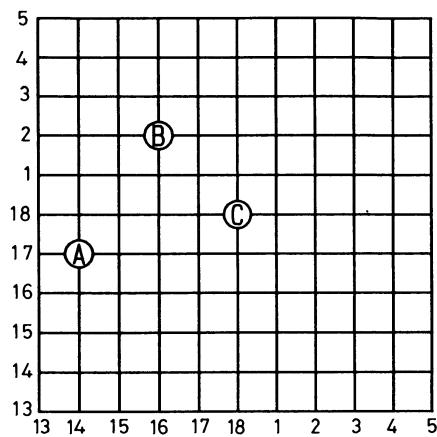


Figure 7 Feature map for load patterns in April 1987 (variable step size, 5 iterations)

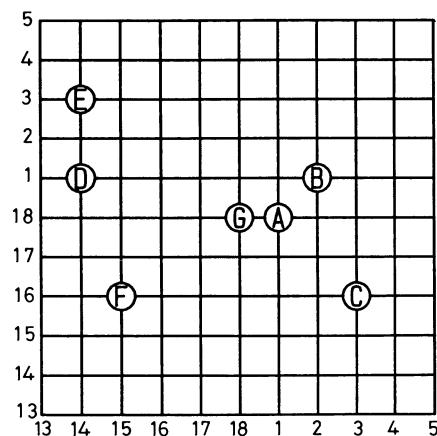


Figure 8 Feature map for load patterns in April 1987 (variable step size, 25 iterations)

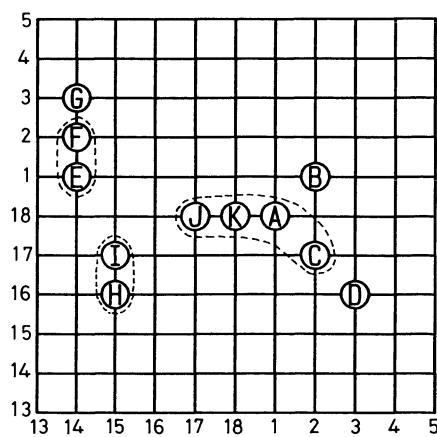


Figure 9 Feature map for load patterns in April 1987 (variable step size, 50 iterations)

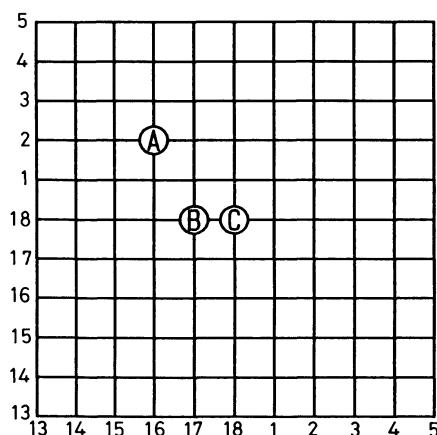


Figure 10 Feature map for load patterns in April 1987 ($\eta=0.002$, 50 iterations)

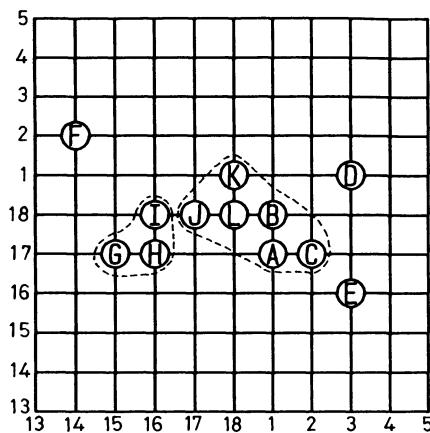


Figure 11 Feature map for load patterns in April 1987 ($\eta=0.005$, 50 iterations)

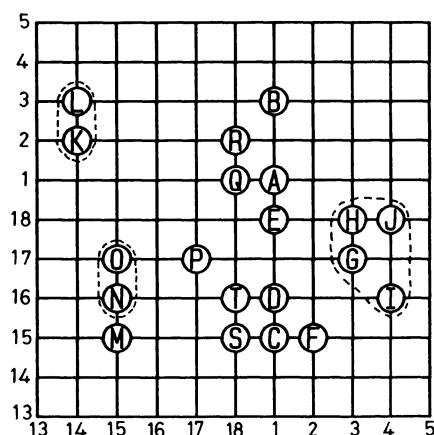


Figure 12 Feature map for load patterns in April 1987 ($\eta=0.008$, 50 iterations)

$\eta=0.002$		$\eta=0.005$		$\eta=0.008$	
Output node	Date(day)	Output node	Date(day)	Output node	Date(day)
A(16,2)	5(SU), 12(SU), 19(SU), 26(SU)	A(1,17)	24(F)	A(1,1)	2(TH), 15(W)
		B(1,18)	1(W), 2(TH), 3(F) 14(T), 15(W)	B(1,3)	16(TH), 17(F)
B(17,18)	6(M)	C(2,17)	8(W), 9(TH), 10(F) 21(T), 23(TH), 29(W)	C(1,15)	23(TH), 29(W)
C(18,18)	Others	D(3,1)	6(M)	D(1,16)	9(TH)
		E(3,16)	7(T), 13(M), 20(M), 27(M)	E(1,18)	1(W)
		F(14,2)	5(SU), 12(SU), 19(SU), 26(SU)	F(2,15)	21(T)
		G(15,17)	4(S). 11(S)	G(3,17)	27(M)
		H(16,17)	18(S)	H(3,18)	7(T)
		I(16,18)	25(S)	I(4,16)	6(M)
		J(17,18)	30(TH)	J(4,18)	13(M), 20(M)
		K(18,1)	16(TH), 17(F) 28(T)	K(14,2)	26(SU)
		L(18,18)	22(W)	L(14,3)	5(SU), 12(SU), 19(SU)
				M(15,15)	30(TH)
				N(15,16)	25(S)
				O(15,17)	4(S), 11(S), 18(S)
				P(17,17)	22(W)
				Q(18,1)	3(F), 28(T)
				R(18,2)	14(T)
				S(18,15)	8(W), 10(F)
				T(18,16)	24(F)

Abbreviations: see Table 8.1

Table 2 Summary of the feature map for April 1987 (fixed step size, 50 iterations)

4.2 Example 2: Load Patterns in May 1986 and November 1986

To examine the effectiveness of the proposed neural network in different seasons, day type identification is also performed on other months of the year. Figures 8.13 and 8.14 show the feature maps for the load patterns in May 1986 and November 1986, after 50 iterations of training. The maps are summarised in Tables 8.3 and 8.4, respectively.

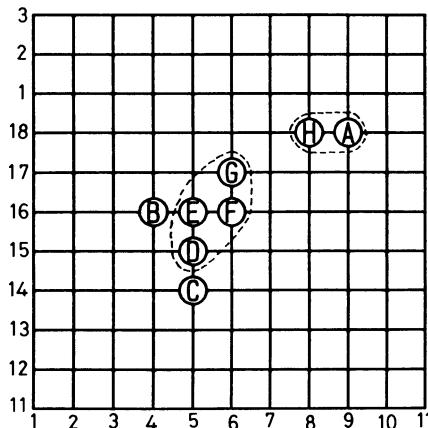


Figure 13 Feature map for load patterns in May 1986

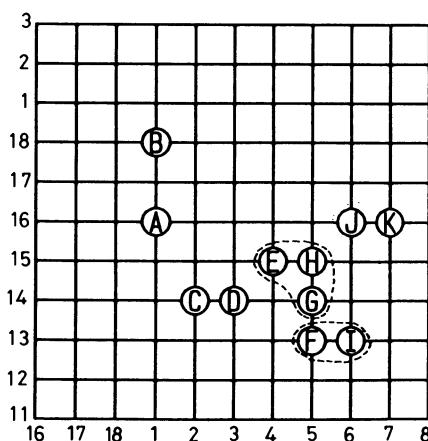


Figure 14 Feature map for load patterns in November 1986

Output node	Date(day)
A(9,18)	1(TH), 11(SU)
B(4,16)	2(F), 5(M), 12(M), 19(M), 26(M)
C(5,14)	3(S), 10(S), 17(S), 24(S), 31(S)
D(5,15)	6(T), 7(W), 9(F), 16(F), 20(T), 28(W), 29(TH)
E(5,16)	8(TH), 13(T)
F(6,16)	14(W), 15(TH), 21(W), 22(TH), 27(T), 30(F)
G(6,17)	23(F)
H(8,18)	4(SU), 18(SU), 25(SU)

Abbreviations: see Table 8.1

Table 3 Summary of the feature map for May 1986

Output node	Date(day)
A(1,16)	2(SU), 9(SU), 16(SU), 23(SU), 30(SU)
B(1,18)	28(F)
C(2,14)	12(W)
D(3,14)	1(S)
E(4,15)	4(T), 26(W), 27(TH)
F(5,13)	22(S), 29(S)
G(5,14)	5(W), 11(T), 19(W), 20(TH), 25(T)
H(5,15)	6(TH), 7(F), 18(T), 21(F)
I(6,13)	8(S), 15(S)
J(6,16)	3(M), 10(M), 13(TH), 17(M), 24(M)
K(7,16)	14(F)

Abbreviations: see Table 8.1

Table 4 Summary of the feature map for November 1986

4.3 Discussion of Results

From the results in Figures 8.6-8.14 and Table 8.1-8.4, the following observations can be made.

First, from the feature maps in Figures 8.6-8.9 it is observed that new output nodes, each corresponding to a group of input patterns with a particular feature, emerge as input patterns, are sequentially presented, and the connection weights are updated during training. From these feature maps, the day types as listed in Table 8.5 can be identified. Note that days of which the load patterns are mapped to the same output node or neighbouring nodes are referred to as of the same day type.

In Table 8.5 it is observed that after 5 iterations of training, only the output nodes (A,B) for Sundays appear in the feature map. But all the six important load patterns appear in the feature map after 25 iterations. The basic principle for clustering the output nodes is to put neighbouring nodes in a group, but the way of clustering is by no means unique. For example, output node G, which is the map of the load pattern for Mourning Day, is regarded as a separate group in Table 8.5, making Mourning Day a special type of day. In fact, because Mourning Day is a national holiday and output node G is close to nodes F and E, one can also regard nodes E, F, and G as the nodes in the same group. It is also noted that the day after Mourning Day (Monday 6th April) is also a holiday. Thus the output node of this day, B, is very different from output node D for Mondays. In addition, the following day (Tuesday 7th April) is mapped to node D for Mondays.

Day type	Description	Output nodes			
		1 iteration (Fig.8.6)	5 iterations (Fig.8.7)	25 iterations (Fig.8.8)	50 iterations (Fig.8.9)
1	Sunday	A	A, B	D	E, F
2	Weekdays except holidays			A	A,C,J,K
3	Monday			C	D
4	Saturday			F	H, I
5	Special day: Mourning Day (Sunday 5th April)			E	G
6	Special day: the day after (Monday 6th April)			B	B

Table 5 Identified day types based on the feature maps in Figures 4-7

Secondly, the value of step size η affects the learning speed and the resultant feature maps to a great extent. As shown in Figure 8.10, a small step size of 0.002 will make the neural network converge very slowly during training. In fact, only the output nodes for Sundays (node A) and Mondays (node B) appear after 50 iterations of training.

On the other hand, a large step size (e.g. 0.008) will improve the learning speed. But similar load patterns can be mapped to several nodes which are not close to one another. For example, the weekdays are mapped to 13 output nodes (nodes A, B, C, D, E, F, H, M, P, Q, R, S, T) in the feature map of Figure 8.12. It appears that a step size of 0.005 will give good results (see Figure 8.11 and Table 8.2) for the load patterns in April 1987. To have fast learning and to avoid the drawback of mapping a similar pattern to widespread output nodes, we use a variable step size[21] in the present study. The step size η is initialised to be 0.008 and is gradually decreased to 0.001 as the training proceeds. It was found that this variable step size would yield satisfactory results for most of the load patterns in five years.

Thirdly, from the feature map for the load patterns in May 1986 (see Figure 8.13 and Table 8.3), the day types as shown in Table 8.6 can be identified. Note that the weekdays in May are mapped to four nodes (D, E, F, G) which are neighbours on the plane (see Figure 8.13). Only one day type is assigned to these four nodes. It is also observed that the load pattern of 1st May (Labour Day) and of Sunday are mapped to two neighbouring nodes A and H. The day after Labour day has a load pattern similar to that of Monday. Thus the load pattern of 2nd May is mapped to the node of Monday (node B).

Day type	Description	Output nodes
1	Sunday and holiday	A, H
2	Monday and day after holiday	B
3	Saturday	C
4	Weekdays except holidays	D, E, F, G

Table 6 Summary of day types for May 1986

Fourthly, in Section 8.6 we will give an example of load forecasting for 5th May 1987. In conducting the load forecasting, we first have to identify the type of day for the given date. Thus we need a list of day types for the month of May. The list of day types in Table 8.6 can serve that purpose because it summarises the types of days for the same month in the preceding year (May 1986). The day type of 5th May 1987 can then be obtained by picking out a proper one from those in Table 8.6. After the day type of the given date has been determined, the desired hourly load pattern can

be achieved by averaging the load patterns of several recent days in April and May of 1987 which are of the same day type as the given date. Detailed procedures will be described in Section 8.6.

Fifthly, in the feature map for November 1986 (Table 8.4) the load pattern of 1st November (Saturday) is mapped to a separate node D, whereas the load patterns of other Saturdays are mapped to two neighbouring nodes F and I. This is because 1st November is a day following a holiday (31st October). In addition, 12th November is a national holiday, and therefore it is mapped to a node C which is separate from the nodes for weekdays (E,H,G); 13th November is mapped to node J for Mondays. The load pattern for 28th November (node B) seems to result from poor data.

Sixthly, when a node separate from all existing clusters appears on the feature map, it can be either the mapping of poor data or the mapping of a new type of load pattern. In this case, operators have to distinguish between the two possibilities. In the event of a new load pattern, the operator may wish to update his original list of day types. Thus, the self-organising feature map is especially useful to system operators when a new load pattern emerges due to changes in the social habits of customers.

Finally, the proposed feature map is capable of generating all the day types in the Taiwan power system that have been identified using other approaches such as the expert system approach [11]. A distinct feature of the proposed approach is that the day after a public holiday (such as Labour Day) is always grouped together with Mondays. This has not been reported in previous work[11].

5 APPLICATION OF MULTILAYER FEEDFORWARD ANN TO PEAK LOAD AND VALLEY LOAD PREDICTION

Artificial neural networks can be divided into those trained with and without supervision [21]. The self-organising feature map described in Section 8.3 is a kind of unsupervised neural net, as we do not specify the desired output for each input pattern. For the problem of peak and valley load forecasting, we know in advance the output (peak and valley load) for each input pattern (weather data and others) in the training set. Thus we need a neural net with supervised learning. The multilayer feedforward neural network [22] will be employed for this purpose.

To demonstrate the effectiveness of the multilayer feedforward ANN, load forecasting is performed on the Taiwan power system. The forecast results for 5th May 1987 are described below.

As we mentioned earlier, the first step in load forecasting is to identify the type of day to be forecasted. To do this, the day types identified for the same month in the preceding year (May 1986) are examined. From Table 8.6, we have four types of day for the month of May, i.e. Sundays and holidays; Mondays and the day after a holiday; Saturdays; and normal days. Since the day under consideration (5th May 1987) is a normal working day of Tuesday, it is classified as a normal day. To obtain the hourly load pattern $L_n(i)$, $i = 1, 2, \dots, 24$, of the day, the hourly load patterns of ten previous normal days in April and May 1987 are fetched from the database and averaged. The results are listed in Table 8.7.

Hour i	Hourly load pattern $L_n(i)$
1	0.1102
2	0.0550
3	0.0207
4	0.0000
5	0.0015
6	0.0485
7	0.1108
8	0.2497
9	0.7282
10	0.9145
11	1.0000
12	0.9877
13	0.5522
14	0.9075
15	0.9599
16	0.9578
17	0.9619
18	0.8373
19	0.9065
20	0.8801
21	0.7681
22	0.5858
23	0.4439
24	0.2828

Table 7 Averaged hourly load pattern $L_n(i)$, $i = 1, 2, \dots, 24$, used in load forecasting of 5th May 1987

With the hourly load pattern at hand, we can proceed to train the multilayer feedforward neural networks which will be employed to predict daily peak load and valley load. In this paper, we propose to use two different neural networks to predict peak load and valley load separately. In training the neural net for peak load forecasting, we compile 35 input-output patterns from the normal days in March and April 1987. Each input-output pattern contains the load and weather data of a day that serve as a training pattern in the training set. The input signals for a particular training pattern include the daily high temperatures at three areas (north, centre, and south) of Taiwan on that day and on the previous day, and the recorded high temperatures at three areas and system peak loads on ten preceding days of the same day type as the particular day. Thus there are 46 input signals to the neural network. There are two hidden layers with 46 neurons at each layer. The output layer contains only one unit which provides the only output: system peak load.

The structure of the neural network for valley load forecasting is the same as that for peak load forecasting. However, load temperatures and valley loads must be employed to replace high temperatures and peak loads.

It is noted that the original input-output data in the training set must be normalised so that the output of the neural net can fall within the interval (0,1). In addition, the input signals should be kept small owing to the saturation effect caused by the sigmoid function. Thus the input-output patterns must be normalised before training of the neural network can be initiated. In the present work, the daily peak load and valley load (in MW) are divided by 15000 and the temperature ($^{\circ}\text{C}$) is divided by 50 before they are used for training.

After the multilayer feedforward neural networks for peak load and valley load forecasting have been trained using the generalised delta rule as described in [22], we can apply them to forecast the daily peak load L_p and valley load L_v , respectively. In predicting the peak load, we need the forecasted daily high temperatures at three areas (north, centre and south) on the island of Taiwan for 5th May 1987, the recorded high temperatures at the three areas on 4th May 1987, and the recorded high temperatures at three areas and peak load on ten preceding normal days in April and May 1987. These data are given in Table 8.8. As mentioned earlier, these source data must be normalised before they can be fed into the multilayer feedforward neural network. These normalised data will serve as the inputs on the neural network.

In predicting the valley load, daily low temperatures and valley loads must be used to replace high temperatures and peak loads. The data for valley load forecasting are not given owing to limited space.

Forecasted high temperatures ($^{\circ}\text{C}$) for 5th May 1987:
 21.1(north), 19.1(centre), 23.7(south)
 Recorded high temperatures ($^{\circ}\text{C}$) for 4th May 1987:
 19.4(north), 16.1(centre), 22.6(south)
 Recorded high temperatures ($^{\circ}\text{C}$) for peak loads (MW) for ten preceding normal days in April and May 1987:

High temperatures ($^{\circ}\text{C}$)			Peak Load
North	Centre	South	(MW)
28.3	19.6	28.8	8808
29.5	18.5	27.1	9188
22.1	17.5	26.2	8740
28.8	22.0	29.0	9144
30.2	20.4	29.7	8958
24.9	20.8	28.6	8918
31.4	19.6	31.3	8838
24.1	16.2	25.7	8238
20.3	14.1	23.4	8136
17.1	14.9	20.9	8181

Table 8 Source data for peak load forecasting

	Peak load L_p (MW)	Valley load L_v (MW)
Forecasted value	8484.3	5446.4
Actual value	8570	5445
Error: MW	85.7	1.4
%	1.005	0.024

Table 9 Forecasted peak load and valley load for 5th May 1987

The predicted peak load and valley load for 5th May 1987 are given in Table 8.9. Using the hourly load pattern $L_n(i)$ in Table 8.7 and forecasted peak load L_p and valley load L_v in Table 8.9, the desired hourly load $L(i)$ can be computed using eq.(8.4). The results are listed in Table 8.10 and are depicted in Figure 8.15.

It is observed from Table 8.9 that the errors in the forecasted peak load and valley load by the neural net are around 1% or less. It is also observed from Figure 8.15 and Table 8.10 that the forecasted hourly load curve is very close to the actual load curve. In fact, the average error in the forecasted hourly load is only 0.536%. Therefore it is concluded that accurate forecast results can be achieved by the developed neural network.

Load forecasting has been conducted for other days using the same approach. Only the results for three other days are listed in Tables 8.11-8.13 owing to limited space.

In general, the proposed neural network approach works as well as other approaches such as the expert system approach [11] or statistical techniques, e.g. the Box-Jenkins time series method. In fact, a mean absolute error of 1.64% has been reported in [11].

6 CONCLUSIONS

Two types of artificial neural networks, i.e., Kohonen's self-organising feature maps and a multilayer feedforward artificial neural network, have been developed for electrical load forecasting. The self-organising feature map is employed to identify the day types which are essential to electrical load forecasting. The inputs to the neural network are the 24 hourly load data of a day. Each input load pattern is mapped to a node on the output plane according to its feature. Similar load patterns are mapped to the same node or neighboring nodes. By examining the feature maps for a set of historical load patterns, important day types of the system can be identified. A special feature of the self-organising feature map is that it is capable of identifying a new type of load pattern before the operators can recognise the new day type. Thus it can serve as a valuable aid to system operators in electrical load forecasting.

After all possible day types have been identified based on historical load data, we can proceed to identify the day type for the day under study. To obtain the hourly load pattern for the day, the hourly load patterns of several days in the past which are of the same day type as the day under consideration are averaged. In addition to the hourly load pattern, daily peak load and valley load must be predicted before the hourly loads

Hour	Forecasted loads (MW)	Actual loads (MW)	Error (MW)	% error
1	5780.7	5809.0	-28.3	0.487
2	5616.1	5619.0	-2.9	0.052
3	5509.1	5533.0	-23.9	0.432
4	5446.4	5474.0	-27.6	0.504
5	5450.9	5445.0	5.9	0.108
6	5593.4	5578.0	15.4	0.276
7	5782.4	5707.0	75.4	1.321
8	6203.5	6166.0	37.5	0.608
9	7715.3	7773.0	-57.7	0.742
10	8219.5	8318.0	-98.5	1.184
11	8478.9	8570.0	-91.1	1.063
12	8441.8	8483.0	-41.2	0.486
13	7121.1	7118.0	3.1	0.044
14	8198.3	8252.0	-53.7	0.651
15	8357.2	8398.0	-40.8	0.486
16	8350.8	8386.0	-35.2	0.420
17	8363.7	8471.0	-107.3	1.267
18	7985.8	8034.0	-48.2	0.600
19	8195.3	8125.0	70.3	0.865
20	8115.2	8117.0	-1.8	0.022
21	7775.7	7754.0	21.7	0.280
22	7223.0	7201.0	22.0	0.306
23	6792.4	6795.0	-2.6	0.038
24	6304.1	6265.0	39.1	0.624

Average error 0.536%; maximum error 1.321%; minimum error 0.022%

Table 10 Load forecast results for 5th May 1987

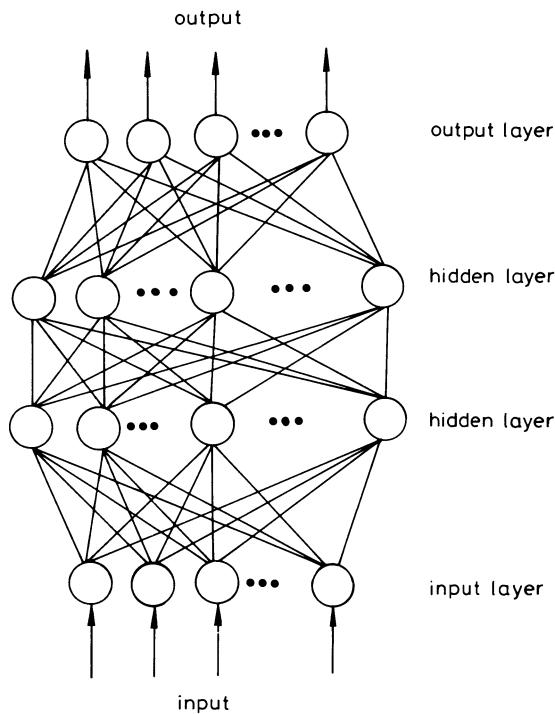


Figure 15 Comparison of forecasted hourly loads () and actual hourly loads(—) for 5th May 1987.

Hour	Forecasted loads (MW)	Actual loads (MW)	Error (MW)	% error
1	5839.9	5816.0	23.9	0.411
2	5671.4	5661.0	10.4	0.184
3	5561.8	5561.0	0.8	0.014
4	5497.7	5486.0	11.7	0.213
5	5502.3	5498.0	4.3	0.078
6	5648.2	5606.0	42.2	0.753
7	5841.7	5748.0	93.7	1.630
8	6272.9	6192.0	80.9	1.307
9	7820.8	7754.0	66.8	0.861
10	8337.2	8242.0	95.2	1.155
11	8602.8	8492.0	110.8	1.305
12	8564.7	8519.0	45.7	0.536
13	7212.4	7177.0	35.4	0.493
14	8315.5	8302.0	13.5	0.163
15	8478.2	8430.0	48.2	0.572
16	8471.6	8460.0	11.6	0.137
17	8484.8	8479.0	5.8	0.068
18	8097.8	7973.0	124.8	1.565
19	8312.4	8063.0	249.4	3.093
20	8230.3	8118.0	112.3	1.383
21	7882.7	7794.0	88.7	1.138
22	7316.8	7317.0	-0.2	0.003
23	6875.9	6900.0	-24.1	0.349
24	6375.9	6383.0	-7.1	0.111

Average error 0.730%; maximum error 3.093%; minimum error 0.003%

Table 11 Load forecast results for 6th May 1987

Hour	Forecasted loads (MW)	Actual loads (MW)	Error (MW)	% error
1	5896.4	5916.0	-19.6	0.331
2	5734.1	5758.0	-23.9	0.415
3	5628.6	5634.0	-5.4	0.096
4	5566.8	5576.0	-9.2	0.165
5	5571.2	5555.0	16.2	0.292
6	5711.8	5623.0	88.8	1.579
7	5898.1	5780.0	118.1	2.043
8	6313.3	6262.0	51.3	0.819
9	7804.0	7884.0	-80.0	1.015
10	8301.3	8371.0	-69.7	0.833
11	8557.1	8730.0	-172.9	1.981
12	8520.4	8691.0	-170.6	1.963
13	7218.1	7377.0	-158.9	2.154
14	8280.4	8479.0	-198.6	2.342
15	8437.1	8636.0	-198.9	2.303
16	8430.8	8631.0	-200.2	2.320
17	8443.5	8648.0	-204.5	2.365
18	8070.8	8218.0	-147.2	1.791
19	8277.4	8267.0	10.4	0.126
20	8198.4	8249.0	-50.6	0.613
21	7863.7	7915.0	-51.3	0.648
22	7318.6	7364.0	-45.4	0.617
23	6894.0	6917.0	-23.0	0.333
24	6412.6	6428.0	-15.4	0.240

Average error 1.141%; maximum error 2.365%; minimum error 0.096%

Table 12 Load forecast results for 7th May 1987

Hour	Forecasted loads (MW)	Actual loads (MW)	Error (MW)	% error
1	5948.8	5951.0	-2.2	0.037
2	5771.6	5793.0	-21.4	0.369
3	5656.4	5661.0	-4.6	0.081
4	5589.0	5567.0	22.0	0.395
5	5593.8	5569.0	24.8	0.445
6	5747.2	5677.0	70.2	1.237
7	5950.6	5807.0	143.6	2.473
8	6403.9	6260.0	143.9	2.299
9	8031.3	7888.0	143.3	1.817
10	8574.1	8420.0	154.1	1.830
11	8853.3	8681.0	172.3	1.985
12	8813.3	8659.0	154.3	1.782
13	7391.7	7320.0	71.7	0.980
14	8551.3	8429.0	122.3	1.451
15	8722.3	8654.0	68.3	0.789
16	8715.4	8568.0	147.4	1.720
17	8729.3	8621.0	108.3	1.256
18	8322.5	8203.0	119.5	1.457
19	8548.0	8342.0	206.0	2.469
20	8461.8	8387.0	74.8	0.892
21	8096.4	8010.0	86.4	1.079
22	7501.4	7419.0	82.4	1.111
23	7037.9	6993.0	44.9	0.642
24	6512.3	6410.0	102.3	1.596

Average error 1.258%; maximum error 2.473%; minimum error 0.037%

Table 13 Load forecast results for 8th May 1987

can be forecast. Thus, a multilayer feedforward ANN is designed to predict the daily peak load and valley load.

To demonstrate the effectiveness of the proposed ANN approach, load forecasting has been conducted on the Taiwan power system. It is concluded from the study results that accurate forecasting of hourly loads can be achieved by the neural network in a very efficient manner.

7 NOMENCLATURE

$L(i)$ = load for hour i

$L_n(i)$ = normalised load for hour i

L_v = daily valley load

L_p = daily peak load

w_{ij} = connection weight between input node i and output node j

N = number of input nodes

M = number of output nodes

N_c = radius of the neighbourhood

η = step size for updating

net_j = input of neuron j

o_j = output for neuron j

θ_j = bias for neuron j

t_{pj} = target output for output node j

α = momentum constant

P = number of training patterns in the training set

8 ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to The Institution of Electrical Engineers, United Kingdom, for allowing us to rewrite the paper which was published in IEE Proceedings, Part C, Vol. 138, No.8, pp. 407-418, 1991, in book form.

REFERENCES

- [1] Gross, G., and Galiana, F.D., "Short term load forecasting," Proc. IEEE, Vol. 75, 1978, 1558-1572.
- [2] Campo, R., and Ruiz, P., "Adaptive weather-sensitive short term load forecast," IEEE Transactions on Power Systems, Vol.2, 1987, pp. 592-600.
- [3] Davey, J., Soachs, J.J., Cunningham, G.W., and Priest, K.W., "Practical application of weather sensitive load forecasting to system planning," IEEE Transactions on Power Apparatus and Systems, Vol. 91, 1972, pp.971-977.
- [4] Thompson, R.P., "Weather sensitive demand and energy analysis on a large geographically diverse power system — application to short-term hourly electric demand forecasting," IEEE Transactions on Power Apparatus and Systems, Vol.95, 1976, pp.384-393.
- [5] Papalexopoulos, A.D., and Hesterberg, T.C., "A regression-based approach to short-term system load forecasting," Proceedings of 1989 PICA Conference, pp. 414-423.
- [6] Christiaanse, W.R., "Short-term load forecasting using general exponential smoothing," IEEE Transactions on Power Apparatus and Systems, Vol. 90, 1971, pp.900-911.
- [7] Meslier, F., "New advances in short term load forecasting using Box and Jenkins approach," IEEE/PES Winter Meeting, 1978, Paper A78 051-5.
- [8] Irisarri, G.D., Widergren, S.E., and Yehsalsul, P.D., "On-line load forecasting for energy control center application," IEEE Transactions on Power Apparatus and System, Vol. 101, 1982, pp.71-78.
- [9] Rahman, S., and Bhatnagar, R., "An expert system based algorithm for short term load forecast," IEEE Transactions on Power Systems, Vol.3, 1988, pp.392-399.
- [10] Jabbour, K., Riveros, J.F.V., Landsbergen, D., and Meyer, W., "ALFA: Automated load forecasting assistant," IEEE Transactions on Power Systems, Vol. 3, 1988, pp. 908-914.

- [11] Ho, K.L., Hsu, Y.Y., and Chen, C.F., Lee, T.E., Liang, C.C., Lai, T.S., and Chen, K.K., "Short term load forecasting of Taiwan power system using a knowledge-based expert system," IEEE Transactions on Power Systems, Vol. 5, 1990, pp.1214-1221.
- [12] Hsu, Y.Y., and Ho, K.L., "Fuzzy expert systems: an application to shrot-term load forecasting," IEE Proceedings, Part C, Vol. 140, 1993.
- [13] Park, D.C. ,El-sharkawi, M.A., Marks, R.J., Atlas, L.E. and Damborg, M.J., "Electric load forecasting using an artificial neural network," IEEE Transactions on Power Systems, Vol. 6, 1991, pp. 442-449.
- [14] Lee, K.Y., Cha, Y.T., and Park, J.H., "Short-term load forecasting using an artificial neural network," IEEE Transactions on Power Systems, Vol.7, 1992, pp. 124-132.
- [15] Peng, T.M., Hubele, N.F., and Karady, G.G., "Advancement in the application of neural networks for short-term load forecasting," IEEE Transactions on Power Systems, Vol. 7, 1992, pp. 250-257.
- [16] Chen, S.T., Yu, D.C., And Moghaddamjo, "Weather sensitive shrot-tem load forecasting using nonfully connected artificial neural network," IEEE Transactions on Power Systems, Vol. 7, 1992, pp. 1098-1105.
- [17] Hsu, Y.Y., and Yang, C.C., "Design of artificial neural networks for short-term load forecasting. Part I: Self-organising feature maps for day type identification," IEE Proceedings, Part C, Vol.138, 1991, pp.407-413.
- [18] Hsu, Y.Y., and Yang, C.C., "Design of artificial neural networks for short-term load forecasting. Part II: Multilayer feedforward networks for peak load and valley load forecasting," IEE Proceedings, Part c, Vol.123, 1991, pp.414-418.
- [19] Ho, K.L., Hsu, Y.Y., and Yang, C.C., "Short term load forecasting using a multi-layer neural networks with and adaptive learning algorithm," IEEE Transactions on Power Systems, Vol.7, 1992, pp.141-149.
- [20] Kohonen, T., "Self-organization and associative memory," Springer, Berlin, 1988.
- [21] Lippmann, R.P., "An introduction to computing with neural net," IEEE ASSP Magazine, 1987, pp.4-22.
- [22] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning internal representations by error propagation," in "Parallel distributed processing, Vol.1" (MIT Press, Cambridge, MA, 1986), pp. 318-362.

ON THE APPLICATION OF ARTIFICIAL NEURAL NETWORKS TO PROCESS CONTROL

M.J. Willis, G.A. Montague and C. Peel

*Dept. of Chemical and Process Engineering
University of Newcastle upon Tyne
Newcastle upon Tyne, NE1 7RU, UK.*

Synopsis: In this chapter, the suitability of the artificial neural network methodology for solving some process engineering problems is discussed. First the concepts involved in the formulation of artificial neural networks for the modelling of dynamic (time dependent) systems are presented. Next the suitability of the technique to provide estimates of difficult to measure quality variables is demonstrated by application to industrial data. Measurements from established instruments are used as secondary variables for estimation of the 'primary' quality variables. The advantage of using these estimates for feedback control is then demonstrated. The possibility of using neural network models directly within a model based control strategy is also considered, making use of an on-line optimisation routine to determine the 'optimal' settings for standard industrial controllers. Application of the control algorithm to a nonlinear distillation system is used to indicate the potential of the neural network based control philosophy.

1 INTRODUCTION

In general, an obligatory requirement when designing any control system is a description of the process. This, more often than not, assumes the form of a mathematical model. Such mathematical models allow the evaluation and analysis of process behaviour by computer simulation as well as providing a means by which processes can be controlled and optimised. Unfortunately, the development of a realistic nonlinear model of a chemical process system based upon a first principle understanding (the underlying mathematics, physics and chemistry) is an extremely demanding task. Process systems may be complex hence often

considerable periods of time must be devoted to first principles (or mechanistic) modelling. Moreover, simplifying assumptions have to be made in many instances to enable a tractable solution to the modelling problem. A first principle model will, therefore, often be very costly to construct and will be subject to inaccuracies due to the assumptions made during the development. A desirable objective is therefore the application of a technique which possesses generality of model structure (facilitating rapid and cheap development) but which could also be capable of learning and expressing the process nonlinearities and complexities. The artificial neural network (ANN) appears to offer this possibility. Indeed, within the Chemical Process Control fraternity the use of such methodologies would appear to be extremely successful (eg. see Bhat et al, 1990, Di Massimo et al, 1991).

The basic motivation behind this chapter is to demonstrate how ANN's can be used within chemical process control. This necessitates a discussion as to how ANN's may be modified in order to model dynamic systems. After introducing the basic concepts of ANN's, and highlighting how dynamical systems may be modelled using this approach, inferential estimation is discussed and a tentative exposition of an inferential controller comprising a neural network model (NNM) and a simple proportional + integral (PI) algorithm is presented. The NNM is used to provide estimates of 'difficult-to-measure' controlled variables by inference from other easily measured outputs. These estimates are then used for feedback control. The philosophy is used to provide more frequent measurements than could be achieved by hardware instrumentation. The particular advantage is that standard industrial controllers can then be employed.

Although the concept of inferential measurement can improve control performance using conventional instrumentation, there are certain situations where more advanced methodologies are required. In particular, the use of model based controllers have been shown to be useful when the process is non-linear or large time delays exist. Significant attention has already been directed to the use of a nonlinear model directly within a control strategy (eg. Lee and Sullivan, 1988; Economou and Morari, 1986; Eaton and Rawlings, 1990). Unfortunately, the above techniques are primarily based upon mechanistic models and are thus dependent upon the accuracy of the nominal model used during control law synthesis. Initial studies by Willis et al (1991) have shown that it is possible to develop a long range predictive controller where the nominal model is a neural network thus facilitating rapid and cheap development of a nonlinear control philosophy. This contribution will discuss an alternative approach to controller design: using an ANN to auto-tune a standard industrial controller.

2 FEEDFORWARD ARTIFICIAL NEURAL NETWORKS

The concept of artificial neural networks as models of cerebral function is far from reality. The link between real neural function and the artificial counterparts primarily resides in the processing functions of individual neurons. However, even these parallels are somewhat tenuous. Indeed it could cynically be argued that the importance of the biological comparison lies more in attaining acceptance for the methodology rather than contributing to development studies. During the past fifty years, since the first inception of artificial neural networks (McCulloch and Pitts, 1943), many alternative network processing scenarios have been suggested. Although a number have found widespread application (Hecht-Neilson, 1991) the feedforward network, one of the most simple in structure, has predominated.

The basic feedforward network is shown diagrammatically in figure 1.

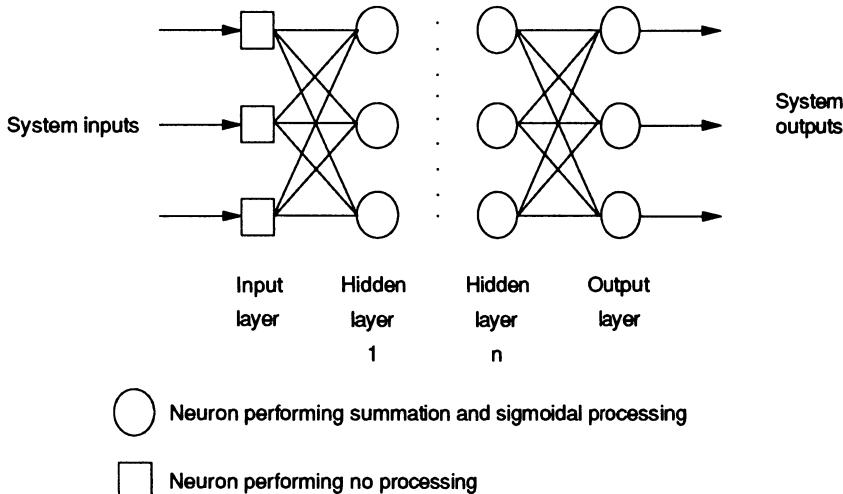


Figure 1 - The basic feedforward network

Scaled data (usually within the range 0-1) enters the network at the input nodes. The data is propagated forward through the network via the connections between hidden layers to the output layer. The network is fully connected (ie. every neuron in a layer is connected to every neuron in adjacent layers). Each connection acts to modify the strength of the signal being carried. In the basic network the signals are modified by scalar multiplication (the connection weight). At neurons the input strengths are summed with a bias term (eqn. 1) and the resulting value passed through a sigmoidal non-linearity (eqn. 2).

$$N_s = w_{tb} + \sum_{i=1}^{N_{in}} w_{ti} s_i \quad (1)$$

$$N_{out} = \frac{1}{1 + \exp(-N_s)} \quad (2)$$

Here N_{in} is the number of neuron inputs, w_{ti} the scalar weight associated with each connection, s_i the signal strength and w_{tb} the bias weight. Data flows forward through successive layers of the network, with the outputs of neurons in one layer (N_{out}) becoming the signal strengths (s) of connections to the following layer. Ultimately data arrives at the output layer, following which it is rescaled to engineering units.

2.1 Topology Selection

The procedure for constructing an artificial neural network model given process data relies to a considerable degree on an understanding of the process problem. Given the architecture shown in figure 1, it is necessary to specify:

- a) the number of network inputs and outputs. This choice is primarily based upon an engineering appreciation of the problem. As with linear identification techniques, highly correlated inputs can degrade the quality of the resulting model. It is therefore necessary to attempt to minimise redundancy in network information.
- b) the number of hidden layers. The literature to date provides conflicting information regarding the choice of the number of hidden layers. Cybenko (1989) postulated that two were sufficient to model any given continuous non-linearity, whilst other papers (Hornik et al, 1989) claimed that one hidden layer was adequate. Experience to date suggests that this decision is best made heuristically; if one hidden layer is insufficient then move to the use of two.
- c) the number of neurons per hidden layer. Whilst trial and error determination of the 'optimal' number of neurons per layer usually produces acceptable results, more considered methods have been suggested. For instance, Wang et al (1991) utilise an approach which assesses redundancy through analysis of hidden layer unit outputs.

Clearly the methods for the selection of artificial neural network topology are not as yet refined, and in particular the problem of hidden unit/layer specification.

Nevertheless, even using trial and error methods topologies that lead to acceptable models can be obtained rapidly.

2.2 Training

Once the topology of the network has been specified, the network weights and biases must be obtained. Since essentially the problem is one of non-linear function minimisation, a variety of methods exist for this task. Willis et al (1991) outline and compare alternative training approaches. The objective, whatever the training methodology, is to minimise the sum of the squared network output prediction error (a cost function F). One of the most straightforward of the learning algorithms is termed Chemotaxis (Willis et al, 1991) and is effectively a random search procedure. Available data is split into two sections, training data and test data. The algorithm is then as follows:

- Step 1 Initialise weights and biases with small random values.
- Step 2 With this set of weights predict outputs for the training data set.
- Step 3 Determine the cost function F_1 over the whole training data set.
- Step 4 Generate a Gaussian distributed random vector
- Step 5 Increment the weight and bias vector with the random vector.
- Step 6 Determine the new cost function F_2 as in step 3.
- Step 7 If $F_2 < F_1$ retain the new weight, set $F_1 = F_2$ and goto step 5. If $F_1 \leq F_2$ goto step 4.

Although the algorithm may appear inelegant, convergence speed compares favourably with conventional gradient based methods (Willis et al, 1991). Furthermore the simplicity of the optimisation approach enables great flexibility in the modification of neuron processing (highlighted in next section). Since the artificial neural network model possesses many parameters it is usually possible to fit (obtain a low prediction error) on the training data. The true quality of the model can only be measured, however, on data other than that which was used for learning (the test data). A poor network fit to the test data may indicate topology modifications are necessary or a more fundamental consideration of the network input/output data is required. If the model fit to the test data is acceptable then it can be concluded that the desired relationship (model) has been captured.

3 DYNAMIC ARTIFICIAL NEURAL NETWORKS

The methodology presented above essentially performs a non-linear mapping of inputs to outputs. Claims by Cybenko (1989) and others that given the appropriate topology the network can 'capture' any continuous non-linear mapping are intriguing. Nevertheless if the technique is to achieve widespread industrial application then it must be capable of modelling dynamic systems. Thus the question arising is how to construct a network model capable of representing process dynamics. Clearly, the most straightforward way to move from steady state neural network models to dynamic is to adopt an approach similar to that taken in linear ARMA (Auto-Regressive Moving Average) modelling. Here a time series of past process inputs and outputs are used to predict the present process outputs. Obviously important process characteristics such as system delays can be accommodated by utilising only those process inputs beyond the dead time of the process. Additionally, any uncertainty in process time-delay can be taken into consideration by using an extended time history of process inputs. Adopting this philosophy when using a neural network inevitably necessitates a significant number of network inputs.

An alternative methodology would be to modify the neuron processing to incorporate dynamics inherently within the network. Hence, in addition to the sigmoidal processing of nodes, the neurons (or transmission between neurons) can be given dynamic characteristics. The simplest approach here is to incorporate a first order transfer function, ie. assume a first order dynamic response. Thus by discretisation of the continuous first order response, the neuron output, N_{out} , is transformed :

$$(N_{filt})_t = (1 - a) (N_{out})_t + a (N_{filt})_{t-1} \quad (3)$$

where a , the filter time constant, is determined along with the network weights.

Whilst the neuron filter approach can eliminate the need to utilise a time history of process variables, any uncertainty in system delays may still warrant a limited history of variables. Even this can be avoided by the incorporation of delays along with the dynamics of the neuron. For instance, by applying a bilinear transformation to the second order Pade approximation. Here the time delay may be specified in terms of sample intervals and is identified along with the weights and filter time constants. Thus the combined neuron processing function is as shown in Fig.2.

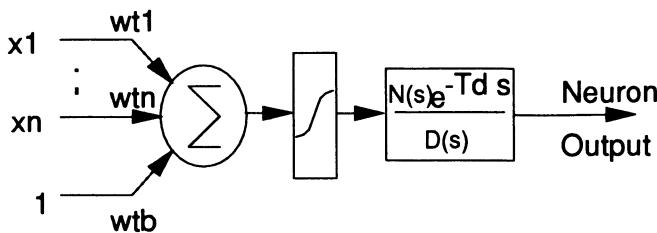


Figure 2 - Combined neuron processing function

The advantage of adopting the filter/time delay neuron approach is the avoidance of the need for a time series of network inputs. As a result network dimension and thus the number of process 'parameters' to be determined a-priori is reduced. Although parallels can be drawn between 'real neuron' processing and the filter/time delay representation in the artificial counterparts (Holden, 1976), the motivation for the use arises more from engineering need than biological analogy.

Although both the dynamic neuron and time history of inputs can both provide a reasonable dynamic representation, if the ARMA modelling philosophy is adopted then additional care is needed. In particular, the training philosophy and assessment of the quality of the resulting model need increased attention. Willis et al (1991b) demonstrate the problems that can occur using industrial examples, however, a very simple example can serve more effectively as a warning. Taking a simple first order model (gain of one, time constant of 10 seconds) sampled at one second intervals with a time delay of 2 seconds, two neural networks were developed to capture the process dynamics; one using the time series approach the other using the dynamic neuron concept. The simple system was subjected to step changes in input (u_t) and the output (y_t) monitored.

Using the 'time series' approach 4 network inputs were specified, these were u_{t-2} , u_{t-3} , u_{t-4} and y_{t-1} ie. delayed values of process input and output, in an attempt to predict process output, y_t . Note that because of an inherent delay due to sampling u_{t-1} has no effect on y_t hence it was not utilised as a network input. The topology of the network was selected as one that had one hidden layer with 3 neurons in this layer, thus a 4-3-1 network was employed. Although, the topology selection procedures employed for the 'static' networks have not as yet been proven on a dynamic network, intuitively the network dimension will be smaller than that of the time series representation. Hence, a 1-2-1 network was utilised with the network input being u_{t-1} . Moreover, it should be noted that in both cases a single hidden layered network was employed as this was found to be sufficient.

The parameters of the two networks were then determined and the results based upon training the network to minimise the current output prediction error gave a high quality of fit. Indeed it appeared that both the network models had captured the essential process characteristics. When, however, the models were used to predict the output of the process further into the future, as may be required for modelling for control system design and optimisation, it is obvious that the alternative modelling approaches are not comparable. Fig. 3 shows a fifteen step ahead prediction of system output.

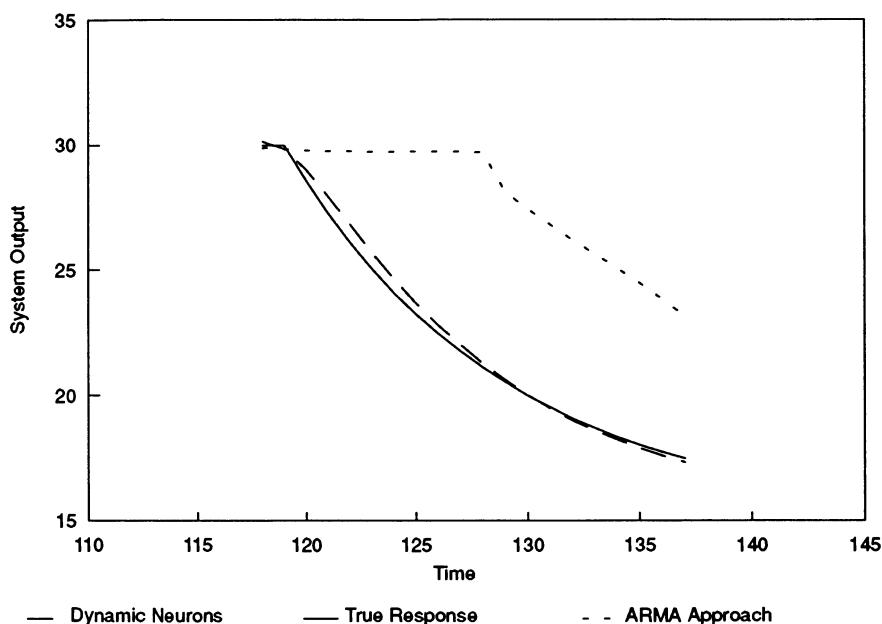


Figure 3 - Fifteen step ahead model prediction

Clearly, the time series network model has not captured the process dynamics. The reason for this lies in the auto-regressive nature of the approach. In order to predict further into the future past estimates must be used as network inputs. Thus, any errors in the past estimates accumulate as the prediction horizon increases. Since the dynamic network model is not auto-regressive this problem does not arise and the predictions of the process output are thus reasonable. The problems of using the time series approach can be overcome by minimising the network prediction error not just for the current output but also for all output predictions to a prediction horizon. Nevertheless, a lesson to be learnt is that when using model fitting techniques that involve minimisation of error, a careful assessment of the quality of the resulting model must be made.

4 INFERENTIAL ESTIMATION

There are many industrial situations where infrequent sampling of the 'controlled' process output can present potential operability problems (eg. distillation columns, chemical reactors and fermentation processes). In these cases, the sampling delay is generally a direct result of the large cycle time of the on-line (or in some situations off-line) analyser. As a direct consequence, when attempting to regulate such variables, even the use of advanced control techniques may often be inadequate. This is due to disturbances remaining undetected for significant periods of time which can result in long disturbance rejection time and significant deviations from set point.

Linear adaptive estimators have been employed to provide 'fast' inferences of variables that are 'difficult to measure' (Tham et al, 1989). Although results from industrial evaluations have been promising, it is suggested that due to their ability to capture nonlinear characteristics, the use of NNMs may provide improved estimation performances. The following sections present the results from recent evaluation studies.

5 INDUSTRIAL EXAMPLES

In order to demonstrate the application of artificial neural networks for inferential estimation two industrial systems are utilised; a continuous fermentation process and an extrusion operation. The dynamic neuron concept has been used in both cases to formulate the process models.

5.1 Continuous Fermentation

To investigate the applicability of artificial neural network modelling approach an industrial fermentation (*Fusarium graminearum*) is considered. An in-depth discussion of the process can be found in Edelman et al (1983). The process operates in a continuous mode, on a large scale (40m³), to produce biomass. Typically the fermenter can run for up to 1000 hours of continuous operation. Adequate accumulation of desired biomass product takes place only in the presence of excess carbohydrate so biomass control through straightforward nutrient limitation is not possible. Process analysis reveals that regulation of biomass concentration is required to achieve the desired product quality. The present regulatory philosophy is based upon the use of off-line biomass assays, carried out every 4 hours, and the results made available to the process operator 1 to 3 hours after sampling. Once the assay result is determined, the dilution rate is

adjusted in an attempt to compensate for process disturbances. The shortcoming of this control scheme is due to the low sampling frequency of biomass concentration compared to process dynamics. Furthermore, because the laboratory analysis of dry weight is subject to significant error, a weighted average of the last three biomass assays is used. Consequently, data up to 15 hours old is utilised for feedback control of biomass. It is therefore not surprising that the control scheme often results in excursions outside of acceptable operating conditions.

Although on-line measurements of biomass concentration are not available to the process operators, a number of other process measurements are available. CER (carbon dioxide evolution rate), AAR (alkali addition rate) and dilution rate are available almost continuously, and all have been shown to give a reasonable indication of the level of biomass present in the fermenter. However, the relationship between biomass concentration and the other process variables has not been quantitatively analysed and indeed, it is expected to change with different operating conditions. The purpose of the modelling exercise was therefore to model the relationships between the on-line data and the biomass concentration. The resulting model(s) could then be used as on-line high frequency estimators for biomass concentration. The increase in the frequency of biomass data, say to every hour, would provide the opportunity to increase the effectiveness of process regulation.

Experience suggests that the most effective variable to use for biomass estimation is the CER (Tham et al 1989). An artificial neural network was therefore specified with a 1-4-1 structure, the input being CER (as measured on the plant) and the output biomass concentration. In this case one hidden layer containing four neurons was found to be sufficient. Data from a recent batch (figure 4) was used for training.

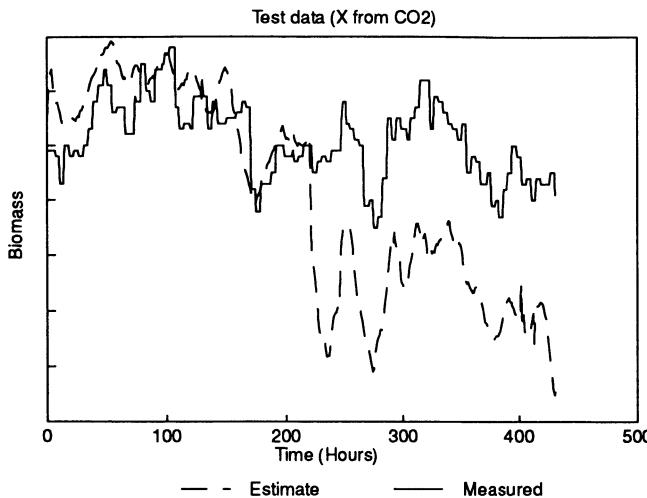


Figure 4 - Biomass estimation (training data)

Data was fed to the network every hour with the estimation error calculated when a new assay becomes available (approximately every four hours). It is worth noting that the quality of fit obtained with the neural network, even during this relatively settled operation, is significantly better (15% reduction in absolute error) than that of a linear moving average model based upon CER. The relative merits of the neural network and linear models are more striking under more severe dynamic upsets. It is possible to track the changing dynamics with a linear model by allowing adaptation (Tham et al, 1991), but the benefits gained must be balanced against a reduction in robustness.

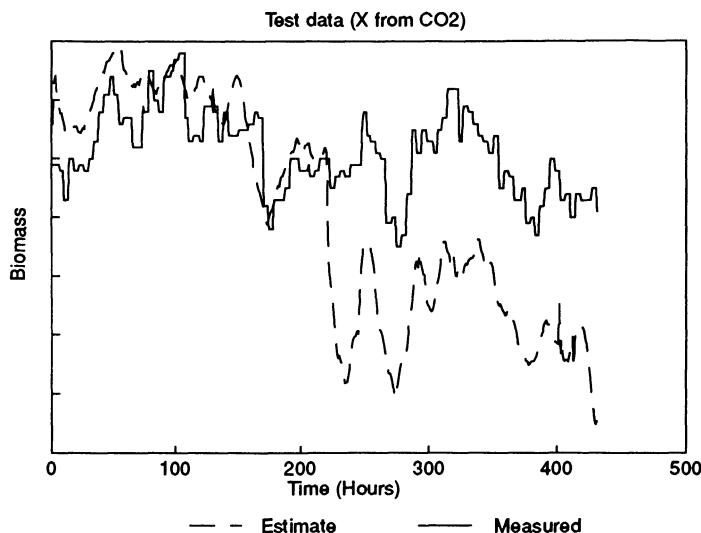


Figure 5 - Biomass estimation (test data)

Whilst a good fit to the training data is obtained with the artificial neural network, a poor fit to test data is observed (figure 5). It is clear that the developed biomass / CER relationship no longer holds. The problem arises due to the saturation of the air flowrate measurement device. It is normal practice when calculating the CER to make allowance for air rate changes. Here although the air rate at the start of the fermentation is similar to that used in the training data, a step change in flowrate during the run can be seen to have disastrous effects on the biomass estimate. Due to measurement limitations it was not possible to account for the air flowrate variation on measured CO₂ concentration. Whilst in this case simple modification of process instrumentation would overcome the problem, similar although less severe effects would be observed for analyser drift.

The problem of CER measurement error can be overcome to a certain extent by considering the standard statistical technique of deviations about mean values. A moving mean can be calculated by heavy filtering of the input / output data. An artificial neural network model relating mean-deviational variables can then be identified. Figure 6 shows the fit to the training data. Infrequent off-line assays for mean determination, coupled with the mean deviational model, can then be used to demonstrate the prediction quality (figure 7). Whilst the quality of model prediction is increased (cf. figure 5) problems still exist at around 230 hours into the run. Since there is a significant change in the mean, offset persists until the calculated mean re-adjusts to the new mode of operation.

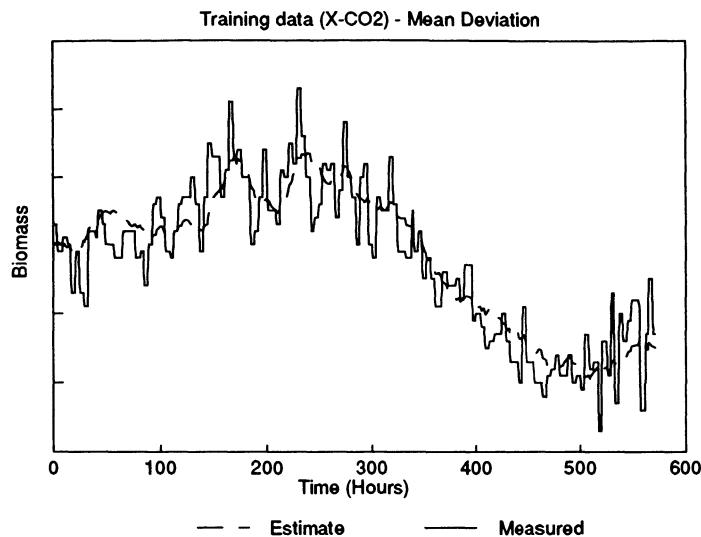


Figure 6 - Mean deviational data network training

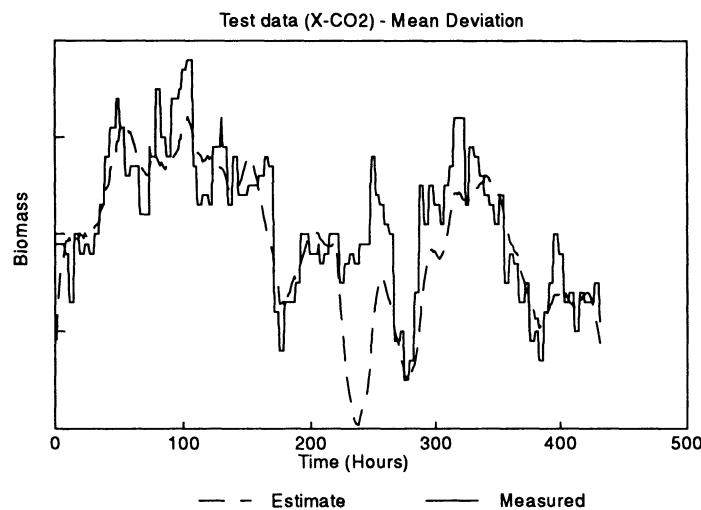


Figure 7 - Mean deviational data network testing

Although the CER / biomass relationship will not always provide accurate estimates due to process problems, as observed in figure 7, other measurements on the plant can be utilised for process estimation. Neural network models relating AAR / biomass and dilution rate / biomass, in addition to the CER / biomass relationship were determined. As a result it is possible to obtain multiple estimates and hence increase the robustness of estimation. When faced with the demands of industrial process operation there is a clear benefit to have redundancy in data, particularly where the measurements are being used for control purposes. Plant data obtained during virtually steady state operation is used to highlight the potential. The laboratory assays shown in figures 8 and 9 confirm steady state operation, with variations within the bounds of measurement error.

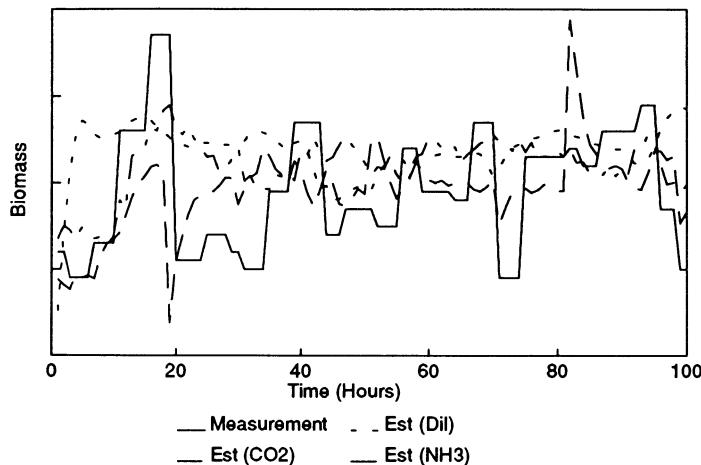


Figure 8 - Biomass estimation with pH problems

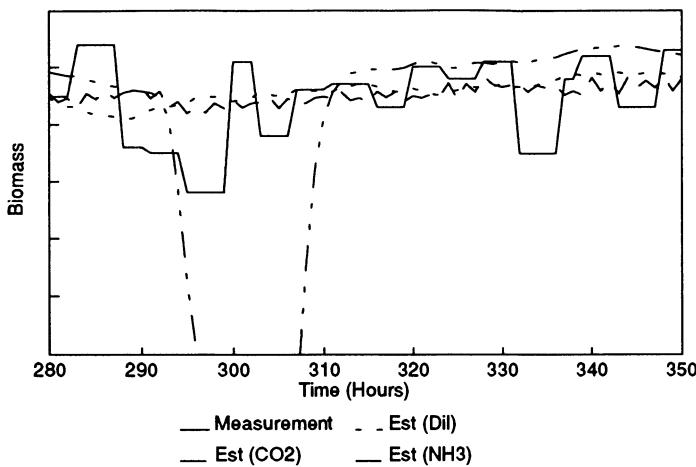


Figure 9 - Biomass estimation with CO_2 analyser fault

Figure 8 shows the three alternative estimates of biomass providing consistent information until 20 hours into the fermentation when pH problems occur. As a result the biomass estimate based upon AAR is no longer valid. When pH control returns to normality the estimate quality is restored. pH problems reoccur at 80 hours and a similar pattern in estimator performance is observed. Figure 9 depicts the effects of CO_2 analyser failure on biomass estimation. Between 296 and 307 hours into the fermentation the gas measurement is lost and hence the estimate becomes invalid. To date multiple process estimates have been obtained; work is presently underway aimed at assessing the estimates in order to provide a combined single 'robust' estimate.

In summary, the advantage of artificial neural network application in this example is that not only can a model be identified of a complex industrial system but also the task of constructing multiple process models is significantly eased. The models developed are more accurate than the linear counterparts and development time was considerably less than first principle models.

5.2 Extrusion Process

The second industrial example concerns the application of the artificial neural network modelling philosophy to an extrusion process (shown diagrammatically in figure 10).

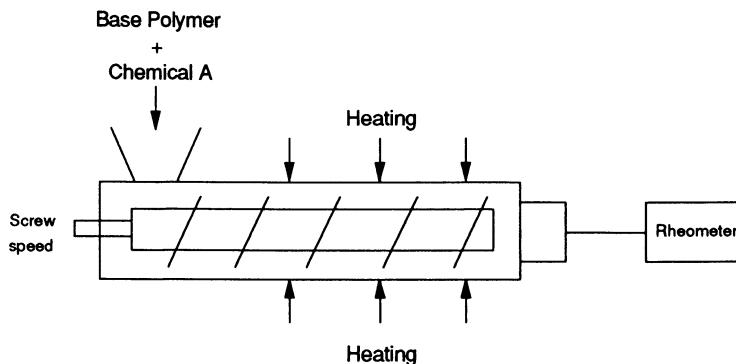


Figure 10 - Extrusion Process Schematic

The system operates to produce a desired polymer by reaction of a base polymer with a chemical A. The quality of the polymer product, determined by viscosity, is influenced primarily by variation of extruder screw speed, operating temperature and chemical A dose rate. The uncertain chemical reaction kinetics and complex process characteristics have inhibited system modelling via the conventional 'structured' approach. It is however desirable to obtain a process model for the purposes of control system design. Artificial neural networks were therefore considered as a means by which to capture the dynamic structure of the process. Quantitative modelling considerations not only suggest that the polymer viscosity is a function of temperature but also a variable process time delay exists which is a linear function of screw speed. Experimental trials have indeed confirmed these assumptions. Whilst conventional feedforward artificial neural networks can cope with fixed delay systems, variable process dead time can not be accommodated. It was therefore essential to model the variable process dead time as a separate relationship and utilise the artificial neural network to capture the relationship from the time adjusted data.

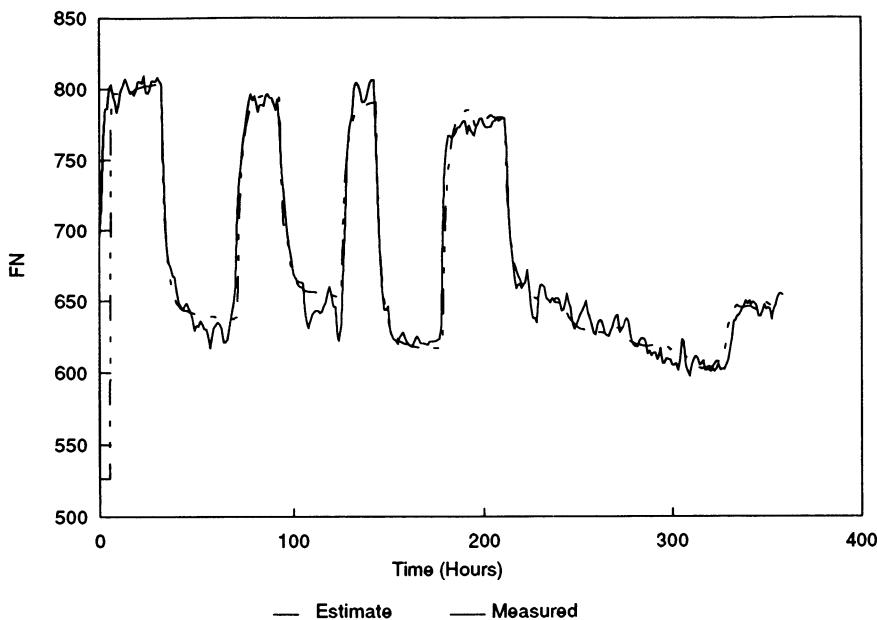


Figure 11 - Estimation of extrusion process performance

Figure 11 shows that with due account of time delay variation it is possible to accurately capture the system dynamics. Here half the data has been used for training and half for testing. With these results it is now possible to address process control considerations. It is important to note, however, that here again data pre-processing was necessary in order to overcome initial problems faced in this application. This serves to highlight that pure 'black box' treatment of data could lead to poor model identification.

6 INFERENTIAL CONTROL VIA ANNs.

With the availability of 'fast' and accurate product quality estimates, the option of closed loop 'inferential' control becomes feasible. The effectiveness of such a strategy has been demonstrated by Guilandoust et al (1987), where adaptive linear algorithms were used to provide inferred estimates of the controlled output for feedback control. Here, the practicality of an NNM based inferential control scheme is explored via application to a penicillin G fermentation.

During this fed-batch process operation two distinct operating regimes can be identified. In the early stages of the fermentation, the system is operated to produce large quantities of biomass, predominantly utilising the substrate in the initial batched media. Towards the end of this phase, the feed additions being made to the fermenter are increased as the initial substrate becomes exhausted. During the second phase of the fermentation, substrate is added at a rate such that the substrate concentration in the broth will be maintained at a low level. As a consequence of the low substrate concentration and the resulting low growth rate, penicillin is produced by the organism. In order to maximise the yield of penicillin, it has been observed that the growth rate should be maintained above a pre-determined minimum value to avoid lysis. However, the maximum penicillin production efficiency is achieved close to the constraint, with the yield of penicillin decreasing as the growth rate increases further.

The present operating regime of off-line analysis to determine fermenter condition results in a conservative feeding strategy. Lysis conditions are avoided at the sacrifice of productivity improvements. The approach to lysis and optimal production of penicillin is restricted since samples are taken relatively infrequently. Thus, it is highly desirable to gain some insight about fermentation behaviour at a higher frequency to enable closer operation to the growth rate constraint and move closer towards achieving the maximum penicillin yield. To increase the frequency of information routinely available from the fermentation would require a move from off-line analysis to on-line measurement or estimation. To date, the instrumentation which provides the necessary data is unfortunately not available. It is for this reason that an estimation scheme has been developed. With more frequent process knowledge available a feedback control scheme could be utilised in order to directly improve fermentation operation. The inferential control scheme is illustrated schematically in Fig. 12.

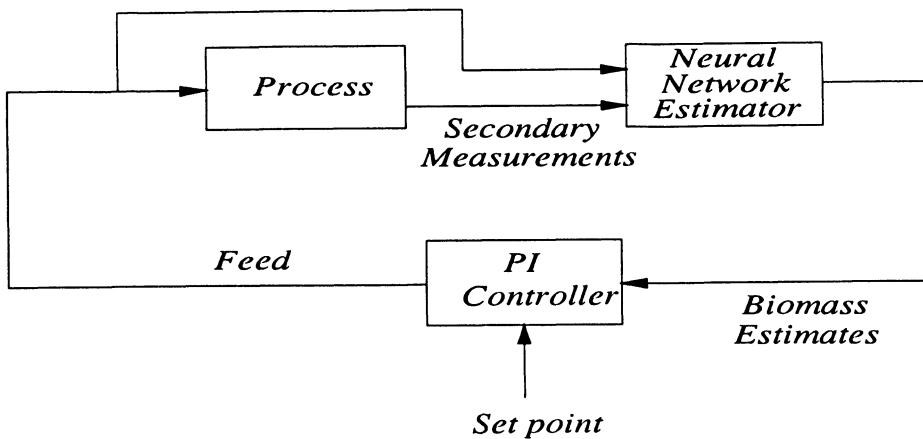


Figure 12 - Inferential Control Scheme

Fig. 13 demonstrates the on-line performance of the inferential controller. Here, a Proportional-Integral (PI) controller is used to regulate biomass to a prespecified setpoint profile. Since biomass cannot be measured on-line, the PI algorithm utilises the estimates provided by the artificial neural network. In the initial batch media there is a high concentration of substrate, therefore control will have negligible effect until this substrate becomes exhausted at around 40 hours. At this time, the feedback control loop is closed and the feed addition is regulated by the PI algorithm. In Fig.13 the biomass is below the desired level initially and it may be observed that the control algorithm brings biomass to the set-point. Biomass estimates and off-line laboratory assays then follow the set-point profile until around 140 hours. At this time an air rate disturbance causes major fermentation problems and the fermentation growth slows. When air flow rate returns to within normal operational bounds then the controller returns the system to set-point. It should be noted that although it may be observed in Fig.13 that the estimate and actual concentrations compare favourably, if there is a difference between the two values, offsets between the actual process output and the desired value will occur. Whilst it was not deemed necessary to address off-set removal in this application, offset free inferential control may be achieved by treating the elimination of estimation errors as a control problem.

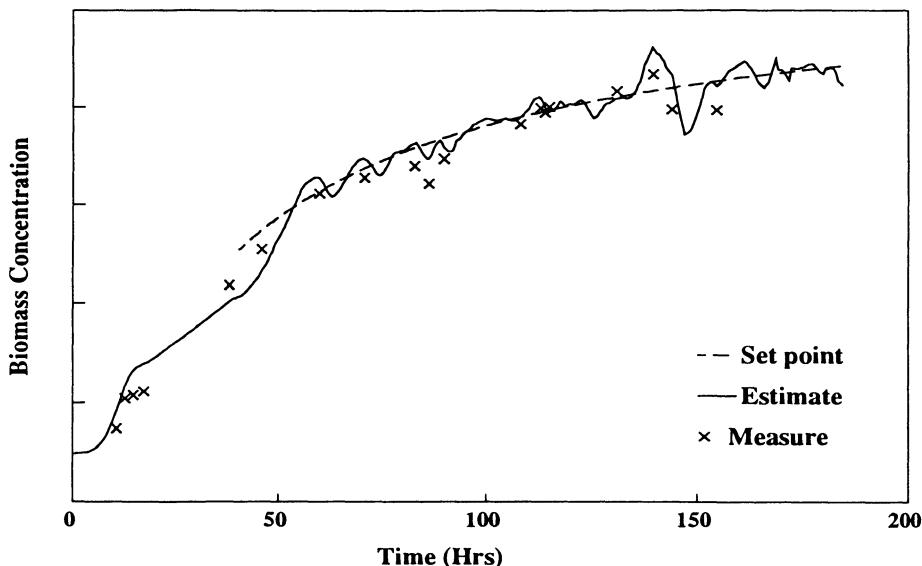


Figure 13 - Closed loop Inferential Control of Biomass

6.1 Neural Network based control

The development of advanced control techniques have yet to have a great impact on industrial practice. Predominantly PI(D) controllers are utilised for process system regulation. Model based control is now emerging as a technology that can offer significant benefits when applied to tackle complex industrial control problems. However, the majority of approaches assume linearity of the process and adopt a control system approach that is far removed from the 'simple' PI(D) structure. For systems with small dead-time, linear models have been shown to be capable of providing a conventional controller auto-tuning capability and achieving performance comparable to more advanced model based controllers. It is therefore of limited benefit to adopt a more complex model based structure. This approach does not, however, overcome the problems caused by process non-linearity. This chapter addresses this issue in conventional controller tuning and investigates the additional benefits to be gained from utilising non-linear neural network process models for conventional control system design.

The attraction of using the neural network instead of any other model form is the ability to effectively represent complex nonlinear systems. Indeed, this attribute

has been exploited in the development of alternate 'advanced' control strategies. For instance, control philosophies such as Long Range Predictive Control and Internal Model Control based on an artificial neural network models have appeared in many applications (eg. Willis et al, 1991; Hunt and Sbarbaro, 1992) and the results presented serve to highlight the benefits of the techniques. Given that model is accurate it is obvious that performance will be improved over 'conventional' control techniques based upon linear models. This is particularly the case when the controller is applied to systems containing significant dead-time or severe non-linearity.

Whilst existing neural network model based control procedures have been shown to improve control performance, a major drawback to industrial acceptance of the technique is the complexity of the resulting algorithm. An alternative approach, which should be more industrially appropriate, is to utilise the neural network model to design a standard controller such as PI(D). Arguably this approach is equally complex, however, the tuning procedure operates at the supervisory level and the control parameters generated may be easily verified and implemented when desired. At the local loop level control is implemented via the standard PI(D) algorithm. Whilst the structure of the PI(D) controller is non-optimal for non-linear systems, the performance of the controller can be tailored to yield the best response characteristics for the current process operating point. Thus non-linearity, constraints and dead-time can be accommodated by controller parameter variation.

The derivation of a discrete PI(D) algorithm may proceed along various lines (Isermann, 1981). In this contribution an incremental discrete PI(D) algorithm is used. This recursive form imparts operational advantages such as inherent anti-reset, windup protection, and bumpless transfer from open to closed loop control.

6.2 Optimisation of performance

The application of a PI(D) controller to any process requires selection of the controller parameters. In this contribution a time response based objective function method is used to select the 'optimal' controller settings. A process model is used to simulate the response characteristics of the process for a variety of conditions around current operating point. An objective function is formulated and the PI(D) controller settings adjusted in a manner such that the 'optimum' response is sought, and hence the optimum PI(D) settings for the plant controller determined. Whilst many objective functions could be utilised, two common techniques have been adopted. The first minimises the Integral Squared Error (ISE) and the cost function used was as follows:

$$J = \sum_{n=N_1}^{N_2} \{ [w(n) - y(n)]^2 + [\lambda \Delta u(n)]^2 \} \quad (4)$$

where $y(n)$, $u(n)$ and $w(n)$ are the controlled output, manipulated input and set-point sequences. N_1 is the minimum output prediction horizon, N_2 is the maximum prediction horizon and λ is a weighting which penalises excessive changes in the manipulated input. While the second approach attempts to achieve a desired decay ratio hence adopts a cost function of the form:

$$J = \sum_{n=1}^{N_d} \text{abs}(\phi_d - \phi) \quad (5)$$

where ϕ_d is the desired decay ratio, ϕ is the model predicted decay ratio and N_d is the number of disturbances to which the model is subjected.

The neural network model is used to simulate the PI(D) controlled process. In order to minimise the chosen cost function it is necessary to utilise an optimisation routine. Due to the need for generality of the proposed controller, gradient free methods are more appropriate. Furthermore, their use imparts flexibility since any model form, either linear or non-linear, can easily be incorporated. Fletcher (1980) showed that the most efficient gradient free technique was due to Powell (1964). This method located the extremum of a function using a sequential unidirectional search procedure. Starting from an initial point, the search proceeds according to a set of conjugate directions generated by the algorithm until the extremum is found. This technique was therefore chosen as the basis of the auto-tuning PI(D) control philosophy. The implementation procedure is shown in Fig. 14.

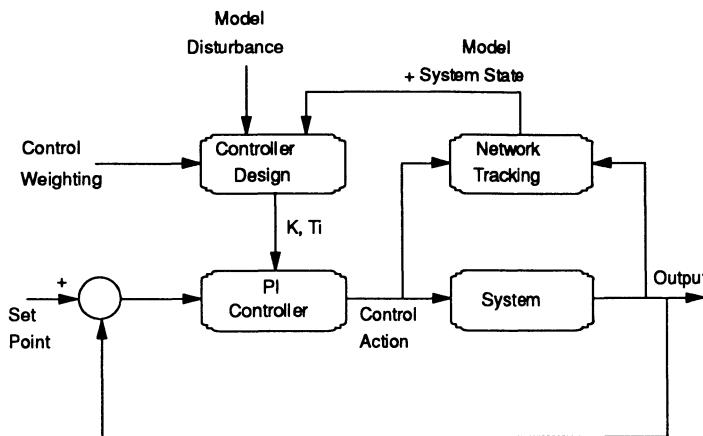


Fig. 14 - Neural Network tuning of PI(D) controller

6.3 Distillation Column Application

The process used to demonstrate the control performance is a nonlinear simulation of a binary distillation column. The 10 stage pilot plant column separates a 50/50 wt% methanol water feed mixture. The column is modelled by a comprehensive set of dynamic heat and mass balance relationships. Both the column and the nonlinear model have been used by many investigators to study advanced control schemes. In these studies the controllers have been assessed in a single loop implementation by subjecting the column to a series of bottom composition set-point changes and feed flow disturbances. Prior to controller implementation it is necessary to construct a dynamic neural network model of the distillation column. The column was subjected to a range of set-point changes and feed flow disturbances spanning the region of likely operation. The resulting data was used to train the neural network model and verify its accuracy. The dynamic neural network model was constructed using a time series of steam flow, column feed and past compositions as network inputs to predict the current bottom composition. On having trained the neural network model it was implemented within the auto-tuning PI(D) control scheme discussed above. Both ISE and decay ratio cost function approaches were investigated.

Fig. 16 shows the response of the bottom composition of the distillation column when subjected to a series of set point changes and a feed flow disturbance (-10%) at 150 minutes.

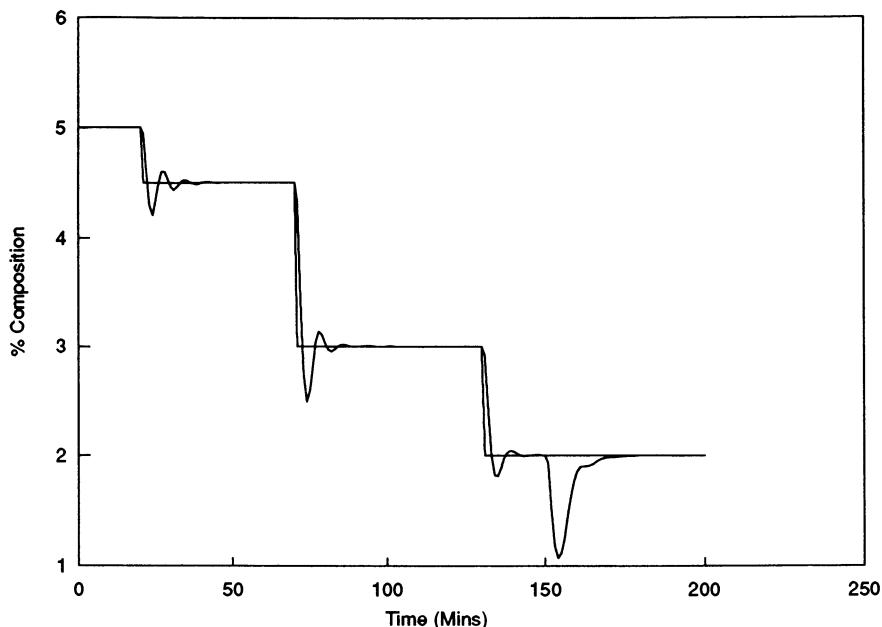


Fig. 15 - PI control response auto-tuned via ISE criteria

A discrete PI controller was implemented with a sample time of one minute with initial estimates of controller gain and integral time. These initial parameters were then updated every eight samples using the supervisory auto-tuning procedure. This choice is made as a compromise between computational burden and the ability to effectively reflect changing system dynamics in the control algorithm design. The auto-tuning procedure involved cost function minimisation by controlling the model subject to set point and feed flow disturbances around the current operating level. Control weighting was adjusted to achieve acceptable performance.

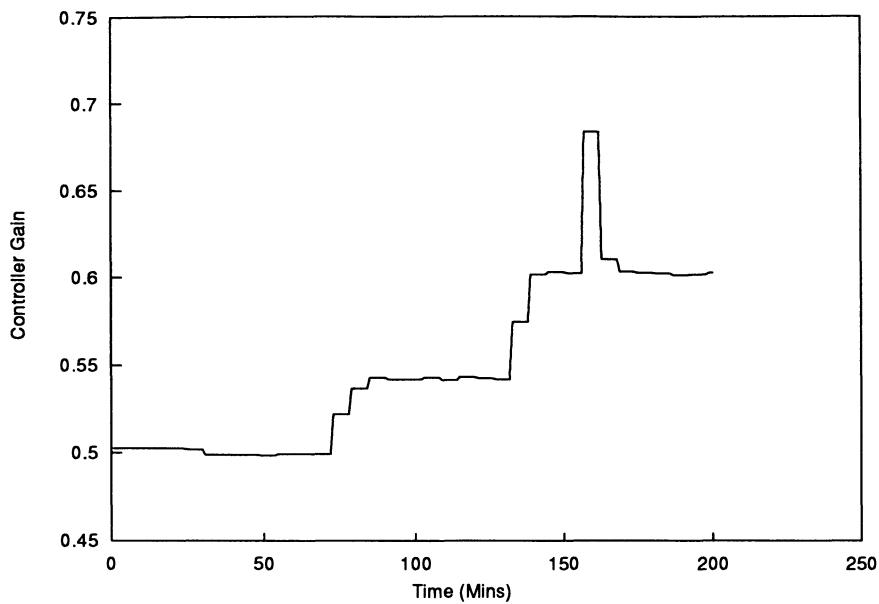


Fig. 16 - Variations in the PI controller gain

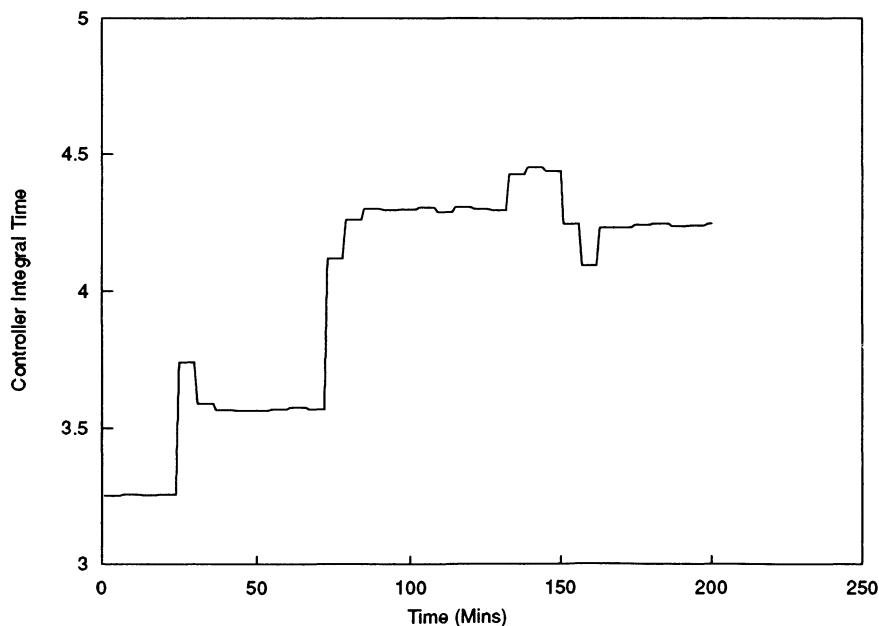


Fig. 17 - Variations in the PI controller integral time

The variation in the controller parameters are shown in Figs 16 and 17. It can be observed that the controller gain and time constant vary to accommodate the changing process condition. In particular, the integral time changes more significantly than the controller gain in response to the changing process operating level when operating at a relatively high composition. As purity increases, more significant changes in process gain necessitate increased controller gain modifications to maintain closed loop performance.

7 CONCLUDING REMARKS

In this contribution, artificial neural networks were discussed with emphasis being placed on the applicability of neural networks as a potential aid in the development of both an inferential estimator and a nonlinear auto-tuning PID controller. It was demonstrated that the neural estimator could provide a 'fast' inference of a 'difficult to measure' process output, from other easily measured variables. Applications to data obtained from industrial processes revealed that given the appropriate network topology, the network could be trained to characterise the behaviour of the systems considered. ANN's thus exhibit potential as soft-sensors; ie sensors based on software rather than hardware (Tham et al, 1989). Additionally, it was demonstrated that significant improvements in process regulation could be achieved if the estimates produced by the neural estimator were used as feedback signals for control. This is possible because the use of secondary variables means that the effects of load disturbances are anticipated in a feedforward sense.

Additionally this chapter has demonstrated how artificial neural networks may be utilised to design conventional industrial controllers. The system to which the procedure was applied, a distillation column simulation, is commonly assumed to be significantly non-linear. However, it is clear that a well tuned PI(D) algorithm is sufficient to provide reasonable regulatory and servo response provided the controller parameters are modified to account for changing process conditions. Here the neural network based model and PI(D) tuning approach may be beneficial when compared to linear adaptive techniques in that a fixed process model is utilised.

Whilst the results presented are evidence that ANN's could be a valuable engineering tool for alleviating many current process engineering problems it must be remembered that in many senses the technique is simply just a tool. The technique should not be applied before a process analysis is performed. In many cases this will reveal where and how savings can be made: either through process redesign or through the application of tighter control (whether or not this is achieved via neural networks the most important point that should be borne in

mind is that the first improves the overall performance of the process whilst the second only provides local improvements). Moreover, it must be emphasised that there are many arbitrary facets of an otherwise promising philosophy which still remain unsolved. For instance, determining the 'optimum' network topology and the development of an established methodology for assessing the stability characteristics of ANN's are two pertinent examples.

8 ACKNOWLEDGEMENTS

The support of the Dept. of Chemical and Process Engng., Uni. of Newcastle upon Tyne; SmithKline Beecham and Zeneca Bioproducts are gratefully acknowledged.

9 REFERENCES

- Bhat N. and McAvoy T.J. (1990). 'Use of neural nets for dynamic modelling and control of chemical process systems', *Comput. Chem. Eng.*, pp 573-583
- Cybenko G. (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, 2, pp. 303-314.
- Di Massimo, C., Montague, G.A., Willis, M.J., Tham, M.T. and Morris, A.J. (1991). Towards Improved Penicillin Fermentation via Artificial Neural Networks', Accepted for Publication in *Comput. Chem. Eng.*
- Eaton, J.W. and Rawlings, J.B. (1990) 'Feedback Control of Chemical Processes using On-line Optimisation Techniques', *Comput. Chem. Eng.*, 14, 4/5, pp 469-479.
- Economou, C.G. and M. Morari (1986). Internal model control 5: Extension to nonlinear systems. *Ind. Eng. Chem. Process Des. Dev.*, 25, 403-411.
- Edelman J., Fewell A. and Solomons G.L. (1983). Myco-protein - a new food. *Nutrition Abstracts and Reviews in Clinical Nutrition, Series A*, Vol 53, 6, pp 471-480
- Fletcher, R. (1980). *Practical methods of optimization*, Volume 1. John Wiley.
- Guilandoust, M.T., Morris, A.J. and Tham, M.T. (1987). 'Adaptive Inferential Control'. *Proc.IEE*, Vol 134, Pt.D.
- Hecht-Nielson R. (1991). 'Neuro-computing', Addison Wesley
- Holden, A.V. (1976). Models of the stochastic activity of neurones, Springer Verlag.
- Hornik K., Stinchcombe M., and White H. (1989). 'Multilayer feedforward networks are universal approximators', *Neural Networks*, Vol 2, pp 359-366
- Hunt K.J. and Sbarbaro D. (1992). 'Studies in neural network based control' in 'Neural networks for control and systems' Ed Warwick K., Irwin G.W. and Hunt K.J. pp 94-122
- Isermann R. (1981). 'Digital control systems', Springer Verlag

Lee, P.L. and G.R. Sullivan (1988). 'Generic Model Control', Comput. Chem. Eng., 12, 6, 573-580.

McCulloch W.S. and Pitts W. (1943). 'A logical calculus of the ideas immanent in nervous activity', Bulletin of Math.Bio., 5, 115-133.

Powell M.J.D. (1964). 'An efficient method for finding the minimum of a function of several variables without calculating derivatives', Comput. J., 7, pp 155-162

Tham, M.T., Montague, G.A., Morris, A.J. and Lant, P.A. (1991). 'Soft-sensors for process estimation and inferential control', J.Proc.Cont., Vol 1, pp 3-14.

Tham, M.T., Morris, A.J. and Montague, G.A. (1989) 'Soft sensing: A solution to the problem of measurement delays', Chem. Eng. Res. and Des., 67, 6, 547-554.

Wang Z., Tham M.T., Morris A.J., (1991). 'Multilayer Feedforward Neural Networks: Approximated canonical decomposition of nonlinearity'. Accepted for publication Int. J. Cont.

Willis, M.J., C. Di Massimo, G.A. Montague, M.T. Tham and A.J. Morris (1991). On neural networks in chemical process control, Proc IEE, PtD., 138, 3, 256-266.

Willis, M.J., Montague, G.A., Morris, A.J. and Tham, M.T. (1991b) 'Artificial neural networks - A panacea to modelling problems ?', Proc. American Control Conference, Boston.

NESTED NETWORKS FOR ROBOT CONTROL

Arjen Jansen* +, Patrick van der Smagt* and Frans Groen

* *Department of Computer Systems
University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
The Netherlands*

+ *I.B.M. The Netherlands N.V.
Marketing Information and Services Centre
Watsonweg 2, 1423 ND Uithoorn
The Netherlands*

1 INTRODUCTION

Sensor based robot control systems can overcome many of the difficulties which are caused by unknown or uncertain models of the environment. Conventional sensor based control systems require explicit knowledge of the kinematics and dynamics of the robot arm and a careful calibration of the sensor system. Instead, adaptive neural controllers can be used to build an internal representation of the camera–motor correspondence from exemplar behaviour and adapt where necessary. In that case, static models are not necessary anymore, and the system can cope with changing behaviour of the robot (wear and tear, payload correction) and its sensors (changing lighting conditions, calibration and re-calibration).

We concentrate on the pick-and-place task. The system gathers its information from angle sensors mounted on the joints of the robot, and a single camera which is situated in the end-effector of the robot. This placement of the camera increases visual precision when the end-effector is near the target and avoids problems of occlusion and pixel correspondence. The position of the target, as seen by the camera, together with the joint angle information of the robot is used to generate a joint angle rotation with which the robot must reach the target.

In our study on neural adaptive robot controllers we have found that proposed methods which attack the problem of camera–motor coordination either suffer from long learning times or from a less precise approximation. For example, Kohonen networks as initially proposed by Ritter et al. [1, 2, 3] can get a precision of around 0.5 cm in the end-effector position in a few feedback steps, but need thousands of iterations to attain reasonable results. The use of a single feed-forward network trained with conjugate gradient back-propagation has been shown to give fast and highly adaptive approximation of inverse kinematics functions, but a large number of feedback steps is needed to get high-precision results [4]. Thus, accurate representations can be obtained with local representations, but it takes many learning samples (and therefore a long time) to get a reasonable distribution amongst all the learning kernels. Conversely, global approximations can quickly establish reasonable overall approximations but the final result will be coarse; also, parallel implementation is more cumbersome.

We combine both global and local representations in our method. The approximation starts with a global map in the form of a fast-learning feed-forward network, but as learning proceeds local maps will be built in a tree-like structure where needed. This method we named the *nested network method* [5, 6].

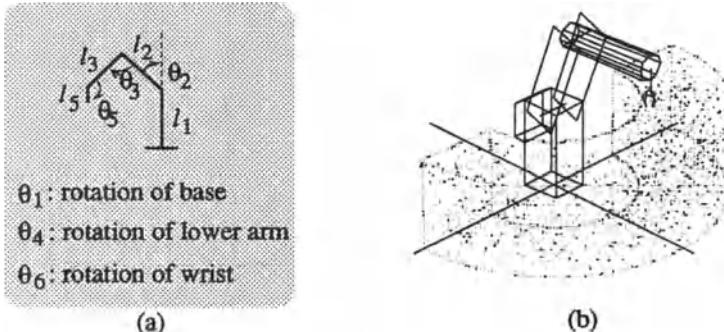


Figure 1 Structure of the OSCAR-6 robot. The robot is depicted with the reach space used in the simulations.

Our neural network approach to learn the inverse kinematics is trained on a robot simulator of a six degrees of freedom (DoF) anthropomorphic robot arm which we have at our disposal (the OSCAR-6 robot); see figure 1. We have restricted ourselves to a 3 DoF system which suffices for many industrial applications (e.g., performing pick-and-place operations). Also, it places such restrictions on the image processing software that object position identification can be realised in real time (i.e., object identification within the time needed for grabbing one image frame).

Restricting ourselves to 3 DoF we have chosen to keep the camera always looking downwards. The position in the image frame, combined with the observed area versus the ‘known’ area, can be translated in a position of the object relative to the end-effector (cf. [7]). Having an ego-centered system we have chosen to control the robot with delta joint values, i.e., a rotation from the current state of the robot.

The aim of our system is to get the object in the centre of the camera image at a predefined size. Both these conditions can be observed by the camera. From the joint and camera information input samples to train the control system can be constructed. This is done according to the input adjustment method [8]. Given a current state of the robot $\vec{\theta} = (\theta_2, \theta_3)$ and the object position $\vec{v} = (x, y, A)$ where A is the observed area of the object, extracted from the camera, the network must generate delta joint values $\vec{\vartheta} = (\Delta\theta_1, \Delta\theta_2, \Delta\theta_3)$ to reach the object position. The value of the θ_1 is not necessary due to the structure of the robot. Figure 2 shows the control structure.

The figures 3 show the inverse kinematics function for a chosen value of θ_2 and

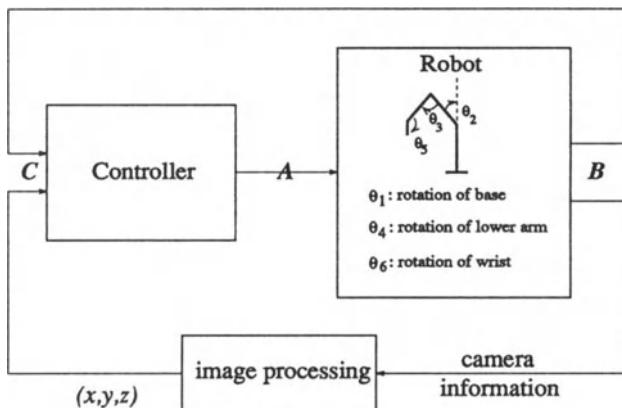


Figure 2 Control scheme of the robot. From the current robot position determined by (θ_2, θ_3) and the target position observed by the camera, both measured at point *B*, a unique input vector \vec{u} is fed to the controller at point *C*. From this input vector, the controller has to generate a robot move $(\Delta\theta_1, \Delta\theta_2, \Delta\theta_3)$ (point *A*). This move has to bring the object in the centre of the camera image at a certain size. If the previous displacement is not accurate enough, this process can be repeated through feedback steps of the system.

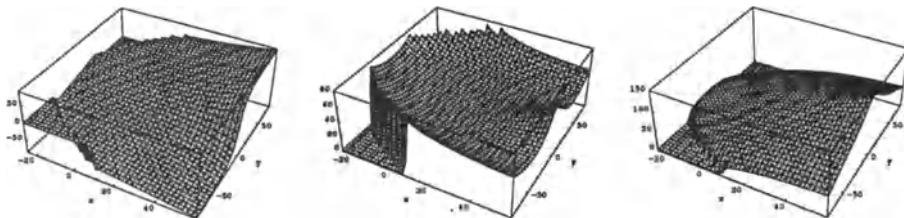


Figure 3 Joint rotation $\Delta\theta_1$ (left), $\Delta\theta_2$ (centre), and $\Delta\theta_3$ (right) necessary to reach the target from the initial state $\theta_2 = -70^\circ$, $\theta_3 = 90^\circ$, $z = 70$ cm. The *x* and *y* axes are the camera input *x* and *y*, respectively.

θ_3 . The x , y , and z values of the input vector determine the target position in the Euclidean space relative to the end-effector. The shown discontinuities are caused by the fact that we set $\tilde{v} = 0$ for unreachable positions.

2 THE NESTED NETWORK METHOD

When approximating a function, the order of the basis functions used in the approximation will, in general, be chosen lower than the order of the function that is approximated. When this function is locally sufficiently smooth with respect to the basis functions, that local approximation can be sufficiently precise. However, there will always be intervals where this match is not good enough, such that basis functions of a higher order must be used, or the subspaces must be chosen smaller. On the other hand, in intervals where the function is sufficiently ‘smooth’ in terms of its basis functions, fewer (i.e., more global) basis functions can be applied over a similarly large interval.

We realise this as follows. The initial representation of the function will be through a single feed-forward neural network, which is thus globally responsible for the whole input space. While learning progresses, to those subspaces in which the approximation is not sufficiently precise, will be assigned a new feed-forward network. This process of splitting up will be repeated until for every part of the input space the desired approximation precision is achieved. This process can be compared with the first part of the well-known *split and merge* process as used in image processing. This method used in image processing was first mentioned in [9].

On the other hand we want to have a very flexible system being able to adjust itself to changes in the system. This means that from the moment that the representation as constructed in the tree does no longer match the measured system (i.e., the learning samples are no longer represented by the existing feed-forward networks), the tree has to be chopped down. This can be achieved by a merge process. During this process learning samples from the ‘old’ environment have to be thrown away together with the corresponding feed-forward networks. In fact, the learning process has to be restarted, depending on how much the system has changed.

The tree consists of three kinds of nodes: *network nodes*, *bin nodes*, and *virtual nodes*. A network node is a node with a feed-forward network, which takes part in the approximation of the function. A bin node is used for storage of the input

patterns, and always resides at the deepest level in the tree (i.e., the leaves). Virtual nodes are created on the path between network nodes and bin nodes. As the approximation of the function is refined, virtual nodes can be changed to network nodes.

When a new learning sample is available, it will be stored in the bin node corresponding to the subspace of the input space from which the input sample is drawn. In case this node does not exist, it will be created, along with the virtual nodes starting at the nearest network node, when necessary.

Network nodes are trained with the learning samples that are found via their child virtual or bin nodes. If the network has no child nodes which are virtual or bin, it will not be trained at all. Figure 4 shows the network nodes of a partly constructed nested network over a two-dimensional input space.

In our nested network approach, the input space of an N -dimensional function that has to be approximated, can be recursively divided in 2^N subspaces until the desired fragmentation is attained for the storage of the input samples. This is done by halving the subspace in each of the N dimensions knowing for each dimension its minimum and maximum value. Now any of the subspaces can now be found in at most d steps with d the recursion depth (the recursion depth is equal to the maximum depth of the tree). We restrict the immediate subspaces of a node to be of equal size for reasons of implementational facility only.

1 Tree structure

Description

The tree for representing a function $f : U \subseteq \mathcal{R}^N \rightarrow \mathcal{R}^M$ consists of nodes with 2^N branches. Each branch of a node at depth d , $1 \leq d \leq D$, is given a value $b_d = [\beta_{N-1}\beta_{N-2}\dots\beta_0]$. We choose $\beta_j \in \{0, 1\}$ such that $[\beta_{N-1}\beta_{N-2}\dots\beta_0]$ is a binary representation of the number of the branch meaning that $b_d = \sum_{j=0}^{N-1} \beta_j 2^j$ (see fig. 4). A node at depth $d + 1$ is represented by the values of b_1, b_2, \dots, b_d of the branches of its ancestors. It represents an input space $U^{(b_1, b_2, \dots, b_d)} \subseteq U^{(b_1, b_2, \dots, b_{d-1})} \subseteq \dots \subseteq U$. At the lowest level each leaf $\langle b_1, b_2, \dots, b_D \rangle$ contains

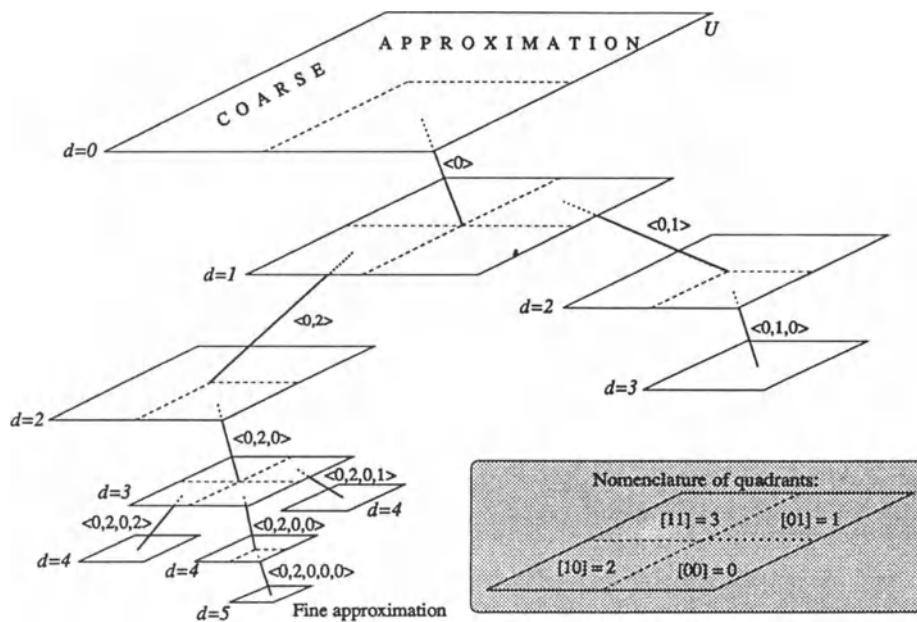


Figure 4 Example of the subdividing of a two-dimensional space U . Each space can be divided into four equally sized parts. If no further subdividing exists the best possible approximation for that specific subspace is reached.

a bin for the storage of patterns of subspace $U^{\langle b_1, b_2, \dots, b_d \rangle}$. By induction, input space $U^{\langle b_1, b_2, \dots, b_d \rangle}$ is spanned by the patterns stored in the leaves of the subtree with root $\langle b_1, b_2, \dots, b_d \rangle$.

Tree descending

To make it easy to descend the tree, with each node a compare vector is associated of the same dimension as the input space. If at a node the decision has to be made which branch must be taken to reach a certain subspace, a comparison with each element of the compare vector must be done. So a subsequent searching through the tree to find a subspace $U_i^{\langle b_1, b_2, \dots, b_d \rangle}$ is dependent of the compare vectors $c^{\text{root}}, \dots, c^{\langle b_1, \dots, b_{d-1} \rangle}$ at nodes root, ..., $\langle b_1, b_2, \dots, b_{d-1} \rangle$. A compare factor determines the value of $b_i = \beta_{N-1}^i \beta_{N-2}^i \dots \beta_0^i$, giving the number of the next branch, as follows:

$$\forall j : \beta_j^i = \begin{cases} 0 & \text{if } U_j^{\langle b_1, b_2, \dots, b_i \rangle} \leq c_j^{\langle b_1, b_2, \dots, b_{i-1} \rangle} \\ 1 & \text{otherwise} \end{cases}$$

with $0 \leq j < N$.

The compare vectors for nodes on different depths are calculated as follows:

1. At depth 1 (the root), the compare vector is calculated according to:

$$\forall i : c_i^{\text{root}} = \frac{M_i + \mu_i}{2}$$

with M_i the maximum and μ_i the minimum value for dimension i .

2. At depth 2, the compare vector is calculated according to:

$$\forall i : c_i^{\langle b_1 \rangle} = \begin{cases} c_i^{\text{root}} - \frac{|c_i^{\text{root}} - \mu_i|}{2} & \text{if } \beta_i^1 = 0 \\ c_i^{\text{root}} + \frac{|c_i^{\text{root}} - M_i|}{2} & \text{otherwise} \end{cases}$$

3. At depth $d > 2$, the compare vector is calculated according to:

$$\forall i : c_i^{\langle b_1, \dots, b_d \rangle} = \begin{cases} c_i^{\langle b_1, \dots, b_{d-1} \rangle} - \frac{|c_i^{\langle b_1, \dots, b_{d-1} \rangle} - c_i^{\langle b_1, \dots, b_{d-2} \rangle}|}{2} & \text{if } \beta_i^d = 0 \\ c_i^{\langle b_1, \dots, b_{d-1} \rangle} + \frac{|c_i^{\langle b_1, \dots, b_{d-1} \rangle} - c_i^{\langle b_1, \dots, b_{d-2} \rangle}|}{2} & \text{otherwise.} \end{cases}$$

Interpolation

In order to obtain good interpolation between the learning samples stored in the tree, each node $\langle b_1, b_2, \dots, b_d \rangle$ can contain a feed-forward network indicated by $\mathcal{N}^{\langle b_1, b_2, \dots, b_d \rangle}$. This network is used to represent the input space $U^{\langle b_1, b_2, \dots, b_d \rangle}$ and make use of the samples stored at the leaves of the subtree $\langle b_1, b_2, \dots, b_d \rangle$. For any pattern $p^{\langle b_1, b_2, \dots, b_D \rangle}$ of subspace $U^{\langle b_1, b_2, \dots, b_D \rangle}$ the finest approximation has to be found. This is done by a search through the tree for the network representing the smallest $U^{\langle b_1, b_2, \dots, b_d \rangle} \supseteq U^{\langle b_1, b_2, \dots, b_D \rangle}$.

In our implementation multi-layered feed-forward networks are used as interpolation methods. Thus if for an input sample the correct network is found it just has to be propagated through the network, resulting in the interpolated vector at the output units of the network. Of course, any other interpolation method can be used as well.

2 Tree creation

Initially the tree consists of one node, the root, with a randomly initialised feed-forward network \mathcal{N} . This network represents the complete input space.

When a new learning pattern is available in $U^{\langle b_1, b_2, \dots, b_{D-1} \rangle}$, the nodes which lead to leaf $\langle b_1, b_2, \dots, b_{D-1} \rangle$, as well as the leaf itself, are created if not existing. Let us denote the deepest node along this path which contains a feed-forward network (initially the root) by $\langle b_1, b_2, \dots, b_a \rangle$; also, let the subtree in which the new learning sample is stored be denoted by $\langle b_1, b_2, \dots, b_a, b_{a+1} \rangle$.

To find the learning samples for $\mathcal{N}^{\langle b_1, b_2, \dots, b_a \rangle}$, proceed as follows. From each leaf of each subtree of node $\langle b_1, b_2, \dots, b_a \rangle$ in which already a feed-forward network exists, select the most recent learning sample. From all other subtrees, select all learning samples. Thus $\mathcal{N}^{\langle b_1, b_2, \dots, b_a \rangle}$ is trained to give a coarse representation in those subspaces which are also represented by child networks, and a fine representation in the remaining subspaces.

After training, for each of the subtrees $\langle b_1, b_2, \dots, b_a, b_{a+1} \rangle$ of node $\langle b_1, b_2, \dots, b_a \rangle$ containing no networks, the maximum of the summed squared errors of all of its patterns $\max_{p^{\langle b_1, b_2, \dots, b_{a+1} \rangle}} E_p$ is calculated. For each subtree $\langle b_1, b_2, \dots, b_a, b_{a+1} \rangle$ for which $\max_{p^{\langle b_1, b_2, \dots, b_{a+1} \rangle}} E_p$ exceeds a constant ϵ , the network at $\langle b_1, b_2, \dots, b_a \rangle$ is copied to node $\langle b_1, b_2, \dots, b_{a+1} \rangle$. This network is subsequently trained with

learning samples from the subspace $U^{\langle b_1, b_2, \dots, b_{a+1} \rangle}$ it represents. This process is repeated until all summed squared pattern errors are less than ϵ .

Determining the value of ϵ depends on the structure of the feed-forward networks that are used, as well as the ‘smoothness’ of the function in the region that is approximated. When the error in the approximation by a network is large due to the fact that this network has been trained on too few learning samples, it is useless to perform a split operation; the choice of ϵ depends on the form of the learning error curve [10].

The feed-forward networks are trained with conjugate gradient optimisation [11].

3 Tree destruction

When the function from which learning samples are generated is changed considerably, the bins containing the new as well as old learning samples will provide ambiguous learning samples. Most likely, those learning samples cannot be represented accurately anymore with the feed-forward networks that feed on those bins. With the above scheme, the result would be that those networks will be split to take care of the error. This is an undesired situation, so a method is needed to clean up the now incorrect feed-forward networks.

In order to detect such an event, we use the following method. Every new sample is used as a learning sample in all the feed-forward networks situated between the root (inclusive) and the leaf in which this sample resides. Therefore, network nodes close to the root see a more recent history of learning samples than network nodes close to the leaves. The distribution method of learning samples ensures that each new learning sample is taught to the root network, as well as to no more than one of its children; to at most one of the children of that child; etc.

Thus, when the function that is being approximated changes while learning progresses, network nodes closer to the root will have a higher ratio of ‘new’ vs. ‘old’ learning samples. Therefore, these networks will, when the change occurs, have a higher summed squared error than networks close to the leaves. This information can be used to decide on a merge operation (see figure 5).

In the application of the camera-robot coordination, we employ a derived method. When the target, which has to be grasped, is approached, the distance

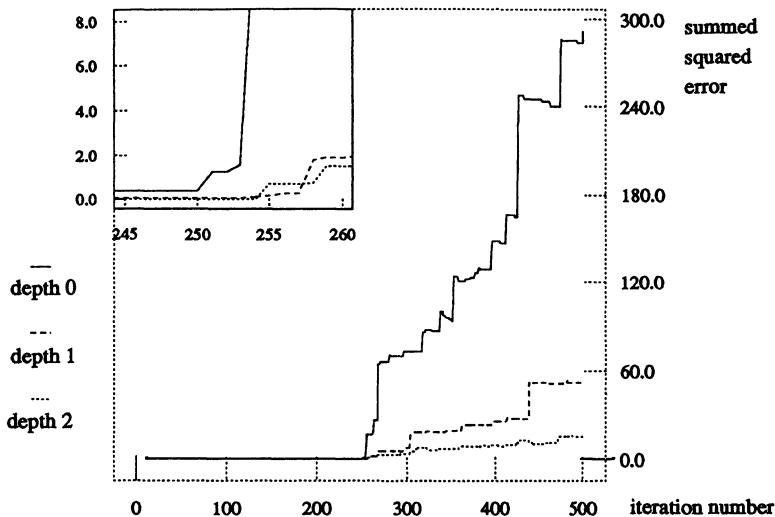


Figure 5 The summed squared error averaged over the network nodes at depth 0 (root), depth 1, and depth 2 during a training session. At iteration 250, the function that is approximated is changed such that the old approximation is not valid anymore. The error at the root increases most dramatically. The inset shows the error increase at iteration 250.

between the robot end-effector and the target should always decrease when a network at a deeper level in the tree generates this move: the deeper networks should provide more accurate moves. When, however, the move makes the manipulator go away from the target, we consider that network to be incorrect and prune it.

Notice that there can be more reasons for such a network to give an imprecise approximation, namely a bad (i.e., not approximately uniform) distribution of the learning set. In such a case, the network will often have to extrapolate instead of interpolate the values, and throwing away such a network is only good riddance.

When a network is pruned, its node will be made into a virtual node. All patterns in its leaves will be cleared out except for the last stored pattern in each leaf; this pattern is considered to be representative for the new situation. If the root is the cause of the bad behaviour, the whole tree will be chopped down, and all the leaves will be removed.

3 SIMULATION RESULTS

To test the method, random target positions are generated within the reach space depicted in figure 1. The tree of maximum depth 6 consisted of feed-forward networks with five hidden units each.

1 Choosing the parameters

There are a few parameters in the learning method which need to be determined.

The values \vec{M} and $\vec{\mu}$ depend on the size of the input space over which the function is approximated. These values are typically user-specified, though could be determined by the algorithm if necessary.

A second choice is that of the number of hidden nodes used in each neural network. It is preferred that this number be kept low, since we favour local over global approximation. Also, our simulation results show that linear feed-forward networks (i.e., no hidden units at all; a method very comparable to [1, 3]) give much worse results than networks with a few hidden units. In most runs we chose five hidden units for each network.

There are a few constants in the conjugate gradient learning process which determine the summed squared error reached in the networks. We choose them according to results published in [11, 10].

The maximum depth that the tree is allowed to grow should, theoretically, be infinity. However, due to the finiteness of computer memory, a value is chosen higher than the maximum depth the tree (i.e., the deepest network node) would reach in a typical run. In our case, we set the maximum depth to six, while a tree deeper than four network nodes is never grown.

Determining the moment of merge is done in accordance with the method described in section 2. We are more lenient than pruning a branch as soon as the robot moves away from the target, since that causes too much merging. A threshold value of 2 cm is chosen rather arbitrarily.

The final parameter is the number of feedback steps that the system is allowed to make. This parameter is not essential for the learning process but is used

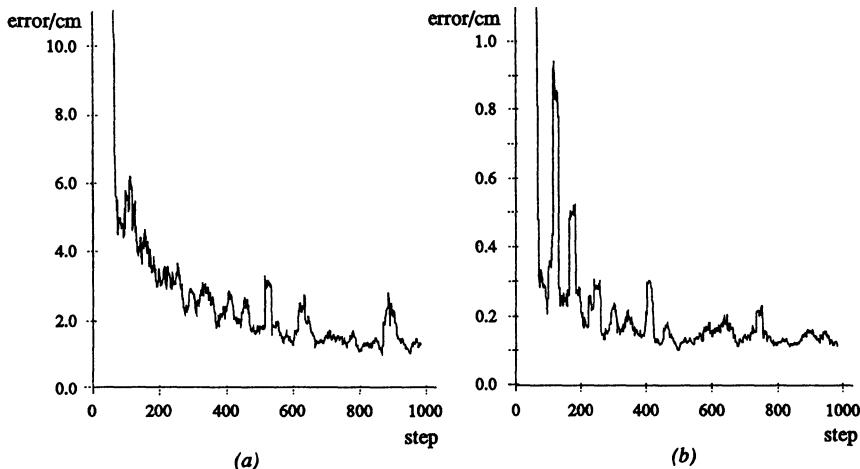


Figure 6 Grasping error in cm. after one step (a) and three steps (b) by the robot. The curves have been smoothed with a moving average filter of width 20. Notice the scale difference between (a) and (b).

to prevent the robot from getting stuck in a part of the workspace where the approximation is not sufficiently good.

2 Exploration phase

First, a form of pre-learning has to be applied to the existing network at the root. If only a few (similar) learning samples are taught to the network, the network is likely to generate a similar output independent of the applied input. If subsequently this move is learned by the network, by reinforcement all next generated displacements will be in the same direction. In this case the input space will not be explored. A solution of this problem is to perform some random movements, and train the network with the corresponding learning samples. The network will now be able to generate movements in different directions and the input space can be explored.

In our test-runs, the robot has to reach its target position within a precision of 1 mm. A maximum of 20 feedback steps is allowed. While the robot is moving, camera and joint encoder values are measured, and stored as learning samples in the tree. Figure 6 shows the grasping error in cm. when the robot is allowed to make one (no feedback) and three (two feedback steps) moves. Figure 7

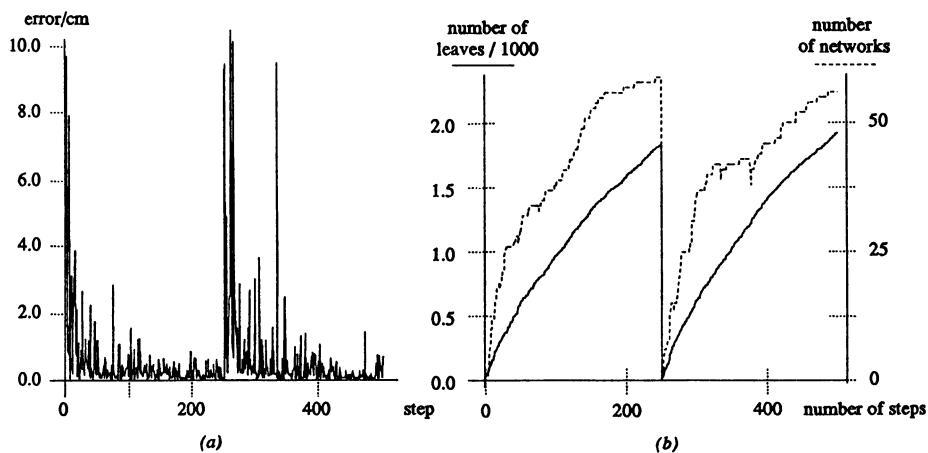


Figure 7 (a) The distance end-effector-object in cm. after 3 steps. At iteration 250 the hand-held camera is given a 90° rotation, such that the robot function is changed. Due to the merge, the network tree will be destroyed and a new tree rebuilt. (b) The number of leaves in the tree, and the number of networks in the tree. At iteration 250, the collapse of the network is clearly visible.

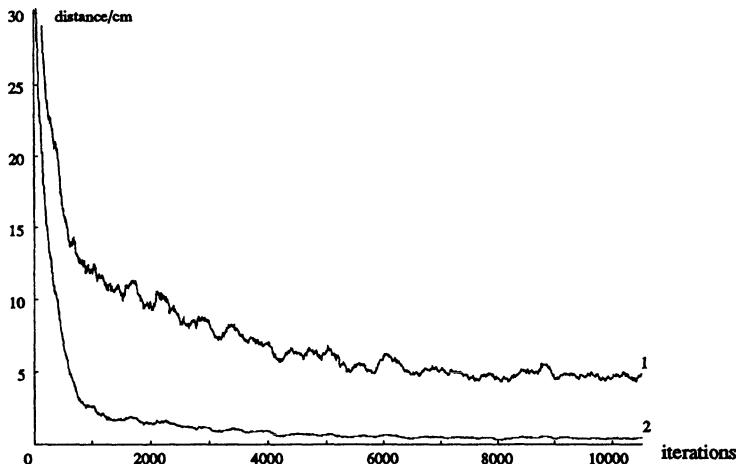


Figure 8 The grasping error for the robot trained with a Kohonen neural network after one and two feedback steps [12]. The curve has been smoothed with a moving average filter of width 20.

show the precision after three feedback steps, as well as how the network grows during learning. At iteration 250, the robot function is changed by rotating the hand-held camera by 90° , which in effect corresponds with swapping the x and y inputs of the nested network. Obviously, the nested network is subsequently destroyed, and a new one is gradually built.

We compare the grasping error attainable with the nested network with those attained with a Kohonen and a single feed-forward network. A Kohonen network reaches a precision of around 0.5 cm with two steps towards the target, but needs many learning steps to get this result (figure 8). Conversely, a single feed-forward network exhibits fast learning but needs more feedback steps to reach the target (figure 9).

4 CONCLUSION

We have developed a method which is able to represent unknown multi-dimensional functions from samples randomly drawn from its input space. Because we start with a global approximation represented by one feed-forward network and splitting up only those parts that are not well approximated, a good generalisation character is obtained.

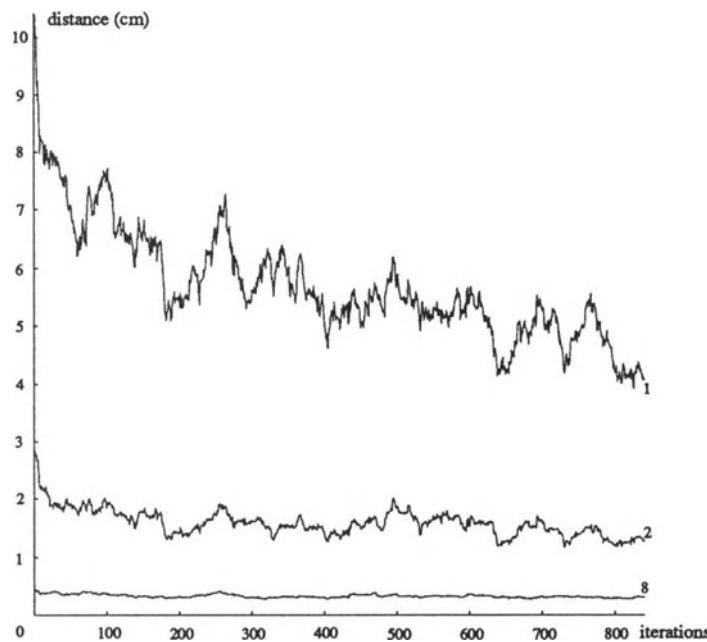


Figure 9 The grasping error for the robot trained with a single feed-forward neural network after one, two, and eight feedback steps [4, 12]. The curve has been smoothed with a moving average filter of width 20.

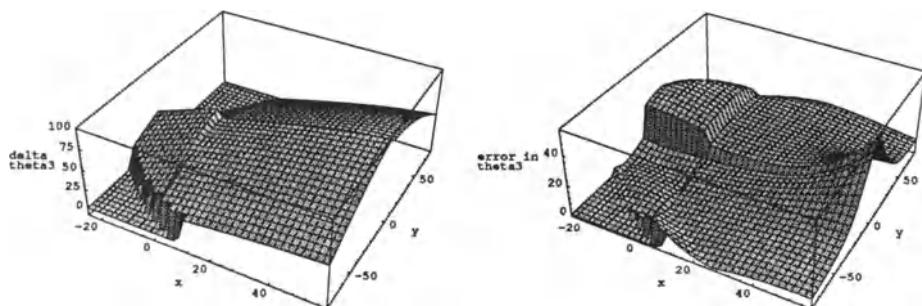


Figure 10 Approximation and error in approximation for joint 3, using the same parameters as in fig. 3, after one step.

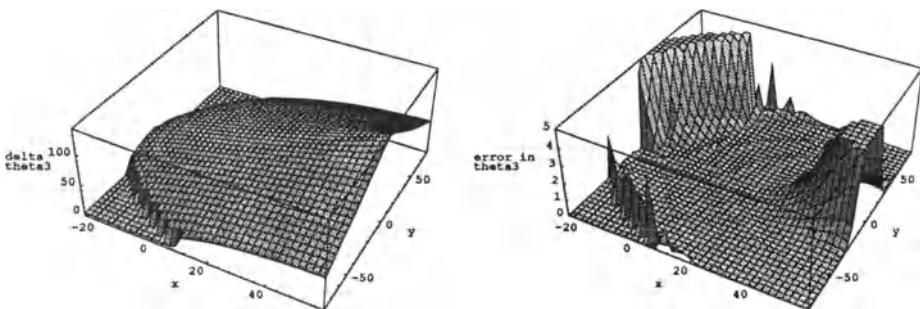


Figure 11 As figure 10, after 3 steps.

The Nested Network method requires a minimum of knowledge of the function that is approximated, thus making it very suitable for controlling time-varying systems. We have applied the method to an eye-hand coordination problem by making use of a robot-simulator with one camera placed in the end-effector. During simulations to tackle this problem with the Nested Network method, it has shown to adapt in few iterations, being able to obtain high precision, i.e., reaching the target within 1 mm using up to three feed-back steps, and having large flexibility during any stage of the learning process.

When compared with other approaches applied to this problem, a Kohonen network and a single feed-forward network, our method has shown to be superior due to its fast learning, high precision results, and superior relearning abilities. For example, our method reaches an average precision of less than 0.5 cm. in the end-effector position with two feedback steps after only 200 learning trials, where a Kohonen network needs 5000 trials, and a feed-forward network cannot reach this precision with two feedback steps only.

The method has a good potential for parallel implementation on coarser-grain computers such as Transputer systems. Since the nodes in the tree are independent in their learning, they can be represented on different nodes, thus gaining a near-to-linear performance increase. Further research is planned in this direction, as well as on the use of different interpolation techniques in the nodes.

5 ACKNOWLEDGMENTS

This research has been carried out with partial financial support from the Dutch Foundation for Neural Networks.

REFERENCES

- [1] T. Martinetz, H. Ritter, and K. Schulten. Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Transactions on Neural Networks*, 1(1):131–136, March 1990.
- [2] T. Martinetz and K. Schulten. A “neural-gas” network learns topologies. In *Proceedings of the 1991 International Conference on Artificial Neural Networks*, volume 1, pages 397–402, Espoo, Finland, June 1991.
- [3] T. Hesselroth, K. Sarkar, P. van der Smagt, and K. Schulten. Neural network control of a pneumatic robot arm. Technical Report UIUC-BI-TB-92-15 (*IEEE Systems, Man, and Cybernetics*, to be published), Theoretical Biophysics, University of Illinois at Urbana/Champaign, August 1992.
- [4] P. van der Smagt and B. Kröse. A real-time learning neural robot controller. In *Proceedings of the 1991 International Conference on Artificial Neural Networks*, pages 351–356, Espoo, Finland, June 1991.
- [5] P. van der Smagt, A. Jansen, and F. Groen. Interpolative robot control with the nested network approach. In *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, pages 475–480. Glasgow, Scotland, U.K., August 1992.
- [6] A. Jansen, P. van der Smagt, and F. Groen. High-precision robot control: The nested network. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks 2*, pages 583–586. North-Holland/Elsevier Science Publishers, September 1992.
- [7] P. van der Smagt, B. Kröse, and F. Groen. Using time-to-contact to guide a robot manipulator. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 177–182. Raleigh, N. C., June 1992.

- [8] D. Psaltis, A. Sideris, and A. Yamamura. A multilayer neural network controller. *Control Systems Magazine*, 8(2):17–21, April 1988.
- [9] C. Brice and C. Fennema. Scene analysis using regions. *Artificial Intelligence*, 1:205–226, 1970.
- [10] V. Vyšniauskas, F. Groen, and B. Kröse. Function approximation with a feedforward network: the optimal number of learning samples and hidden units. Technical report, Department of Computer Systems, University of Amsterdam, Amsterdam, Netherlands, 1992.
- [11] P. van der Smagt. Minimisation methods for training feed-forward networks. Technical Report UIUC-BI-TB-92-17 (submitted to *Neural Networks*), Theoretical Biophysics, University of Illinois at Urbana/Champaign, November 1992.
- [12] Arjen Jansen. *Neural Approaches in the Approximation of the Inverse Kinematics Function: A Comparative Study*. Universiteit van Amsterdam, Faculteit Computer Systemen, April 1992. graduation thesis.

ADAPTIVE EQUALISATION USING NEURAL NETWORKS

Sheng Chen

*Department of Electrical Engineering
University of Edinburgh, Edinburgh EH9 3JL, UK*

ABSTRACT

Adaptive signal processing plays a crucial role in many modern communication systems. A particular example of adaptive signal processing is known as adaptive equalisation, which is an important technique for combatting distortion and interference in communication links. The conventional approach to communication channel equalisation is based on adaptive linear system theory. In the past few years, artificial neural networks have been applied to this important area of signal processing. The results obtained have indicated that the neural network approach offers superior performance over the conventional equalisation approach. This chapter continues this theme and investigates the application of neural networks to adaptive channel equalisation with the emphasis on what a neural network attempts to achieve and why it outperforms a conventional equaliser. This allows us to choose the most appropriate neural network structure for the task of equalisation. Issues of adaptive implementation and computational complexity are also addressed.

10.1 INTRODUCTION

Many digital communication channels are subject to intersymbol interference (ISI) and can be characterised by a finite impulse response (FIR) filter with an additive noise source [1]-[3]. The time-dispersion of a channel is due to the restricted bandwidth allocated to the channel or the presence of multipath

distortion in the transmission medium. At the receiver, the ISI must be compensated in order to reconstruct the transmitted symbols, and this is referred to as channel equalisation. Adaptive equalisers have been in operation for a long time [4] and are playing a significant part in the development of modern telecommunications. An example is digital communications over analogue telephone circuits. Adaptive equalisation techniques have enabled an increase of the data rate from 200 bit/s in the early days to the current 19.2 kbit/s. The digital communication scenario studied in this chapter is depicted in Fig. 10.1.

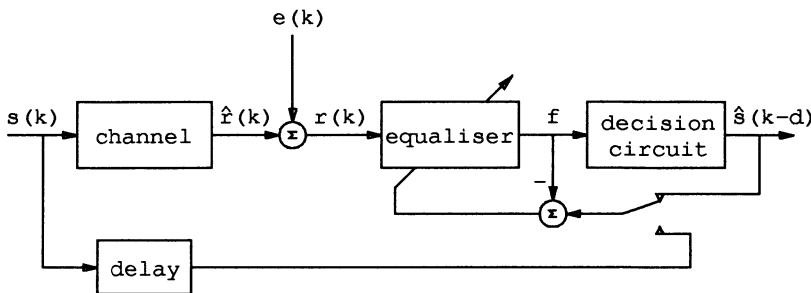


Figure 10.1 Schematic of a digital communication system.

The channel, which is a convolution of the transmitter filter, the transmission medium and the receiver filter, is modelled as a FIR filter with a transfer function

$$A(z) = \sum_{i=0}^{n_a-1} a_i z^{-i}, \quad (10.1)$$

where n_a is the length of the channel impulse response. In general, the transmitted symbol sequence $s(k)$ and the channel taps a_i are complex-valued. In this chapter, unless otherwise stated, the channel and symbols are assumed to be real-valued. This corresponds to the use of multilevel pulse amplitude modulation (M -ary PAM) [2]. Concentration on the simpler real case allows us to highlight the basic concepts. In particular, the case of binary symbols provides a useful geometric visualisation of equalisation process. The symbol sequence is assumed to be independently identically distributed (i.i.d.) with the symbol constellation defined by the set

$$s_i = 2i - M - 1, \quad 1 \leq i \leq M. \quad (10.2)$$

The received signal is given by

$$r(k) = \sum_{i=0}^{n_a-1} a_i s(k-i) + e(k), \quad (10.3)$$

where $e(k)$ is an i.i.d. Gaussian noise source with zero mean and variance $E[e^2(k)] = \sigma_e^2$, and $E[\cdot]$ denotes the expectation operator.

The task of the equaliser is to reconstruct the transmitted symbols as accurately as possible based on noisy channel observations. This is typically done by passing $r(k)$ through an adaptive filter, and the output of this adaptive filter is then quantized into one of the symbol points defined in (10.2) by a decision circuit. The integer d is known as the decision delay and, at sample k , the equaliser estimates the transmitted symbol $s(k-d)$. This decision delay is necessary in the non-minimum phase situation where $A(z)$ has zeros outside the unit circle of the z -plane. Two adaptive modes of equaliser are indicated in Fig.10.1. The lower link marked by the "delay" block represents training mode. During the training period, the transmitter sends out a pre-arranged symbol sequence that is known to the equaliser. The receiver generates this sequence locally, and uses $s(k-d)$ as the desired response to adjust the equaliser parameters. After the initial training period, the parameters of an adaptive equaliser can be continually adjusted in a decision-directed manner. In this mode, equaliser decisions $s(k-d)$ are assumed to be correct and are used to substitute for $s(k-d)$. This decision-directed adaptation allows an equaliser to track slow variations in the channel characteristics during data transmission.

A variety of adaptive equalisers have been developed based on adaptive linear system theory. These existing equalisers can loosely be classified into two categories, namely the linear transversal equaliser (LTE) and the decision feedback equaliser (DFE) [1]. For many practical communication systems, the existing adaptive schemes perform adequately. However, there are strong commercial demands to increase the capacity of existing communication channels. Furthermore, new technology brings new services, which impose much more stringent requirements on adaptive equaliser performance. For example, equaliser design for future public land-mobile telecommunication systems must overcome problems that are much more complex than those encountered in fixed-link communication networks such as telephone channels and microwave line-of-sight radio channels. Existing adaptive structures may not meet these challenges, and this motivates the search for more advanced equalisation schemes.

In the past few years, considerable advances have been achieved in adaptive nonlinear equaliser design based on artificial neural networks [5]-[16]. These studies have demonstrated that the neural network approach can offer significant performance gain over the conventional adaptive linear system approach. On first consideration, if a channel is adequately modelled as a FIR filter, a linear equaliser would be sufficient to accomplish the equalisation objective, and it is not immediately apparent why the nonlinear capability of neural networks is needed in such a situation. This misconception disappears when we view the equalisation process as a classification problem, which seeks to classify an observed channel output vector into one of the finite symbol points in the data constellation. It then becomes clear that the optimal solution for this classification problem is inherently nonlinear and, therefore, neural networks will be superior over a conventional linear scheme.

The LTE and the conventional DFE as well as most of the neural network equalisers belong to the class of symbol-decision structures. A different class of equaliser is based on the maximum likelihood detection of the entire transmitted symbol sequence. The optimal solution for this class of equaliser is the maximum likelihood Viterbi algorithm (MLVA) [17],[18], which determines the estimated symbol sequence $\{s(1), s(2), \dots, s(k), \dots\}$ by minimizing the cost

$$J = \sum_{k=1}^{\infty} \left(r(k) - \sum_{i=0}^{n_a-1} a_i s(k-i) \right)^2. \quad (10.4)$$

The MLVA provides the lowest error rate attainable for any equaliser when the channel is known but is computationally very expensive. In practice, the adaptive MLVA employs a channel estimator to estimate a channel model and, consequently, it inevitably deviates from theoretical performance. The degree of performance loss depends on the nature of channels. For stationary channels, a fairly accurate channel estimate can be obtained and the loss of optimality is negligible. Performance degradation of the adaptive MLVA however becomes very serious for highly nonstationary channels [16]. In contrast a properly designed adaptive neural network equaliser is very robust and suffers a much smaller degradation in the time-varying environment [16].

This chapter is organized as follows. §10.2 introduces the transversal equaliser (TE). The limitations of the adaptive LTE are discussed. A brief summary of using both the multilayer perceptron (MLP) and the Volterra series (VS) filter as an adaptive equaliser is given. By formulating the

equalisation process as a classification problem, the Bayesian equaliser solution is derived in §10.3. It is demonstrated that the Bayesian decision boundary of the equalisation problem is inherently nonlinear. This explains why a neural network equaliser is superior in terms of performance. Moreover the Bayesian solution is shown to have an equivalent structure to the radial basis function (RBF) network and, therefore, the latter is considered to be a most appropriate structure for application to equalisation. §10.4 investigates the DFE and highlights why a neural network DFE is superior in performance to the conventional DFE. A bonus of adopting the Bayesian view is that decision feedback can be utilized to reduce computational complexity. §10.5 compares the Bayesian symbol-decision equaliser with the MLVA. It is demonstrated that, for rapidly time-varying channels, the adaptive MLVA becomes inferior to the adaptive Bayesian equaliser. Some concluding remarks are given in §10.6.

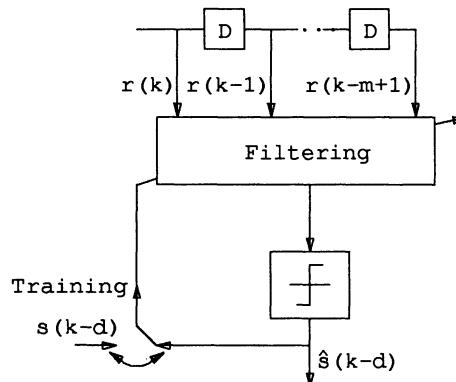


Figure 10.2 Schematic of a generic transversal equaliser.

10.2 TRANSVERSAL EQUALISER

The structure of a generic transversal equaliser is shown in Fig.10.2. Essentially, the equalisation structure defined in Fig.10.2 uses the information present in the observed channel output vector

$$\mathbf{r}(k) = [r(k) \cdots r(k - m + 1)]^T \quad (10.5)$$

to produce an estimate $\hat{s}(k - d)$ of $s(k - d)$ at sample k , hence the name "symbol-decision". The integer m is known as the feedforward order. Traditionally the equalisation process of Fig.10.2 is viewed as an inverse

filter operation where the equaliser forms an approximation to the inverse of the distorting channel [1], and this leads to the LTE

$$f_l(\mathbf{r}(k)) = \mathbf{w}^T \mathbf{r}(k) = \sum_{i=0}^{m-1} w_i r(k-i), \quad (10.6)$$

where $\mathbf{w} = [w_0 \cdots w_{m-1}]^T$ is the filter's weight vector. The thinking behind this approach seems to be rational. The transfer function of the equaliser is

$$W(z) = \sum_{i=0}^{m-1} w_i z^{-i}. \quad (10.7)$$

Assume that the equaliser has an infinite length, that is, $m \rightarrow \infty$. In the absence of noise, if the equaliser weights are chosen to satisfy $W(z)A(z) = z^{-d}$, perfect equalisation is achieved. In practice, allowing $m \rightarrow \infty$ is unnecessary. As long as m is large enough, the ISI will be reduced sufficiently, and the decision quantization will take care of the residual ISI.

When the channel noise is taken into account, however, a problem emerges. The equaliser filter also amplifies the noise, and this has the adverse effect of making the equaliser decisions erroneous. As the number of the equaliser taps m increases, the total noise power on the equaliser input is also increased, and this tends to diminish any advantage gained by increasing m . Therefore the LTE must strike a compromise between eliminating the ISI and enhancing the noise. Because the adaptive LTE converges to the Wiener filter solution, the Wiener filter can be used to predict the performance of the former. The Wiener solution is defined by

$$\hat{\mathbf{w}} = \left(\mathbf{E}[\mathbf{r}(k)\mathbf{r}^T(k)] \right)^{-1} \mathbf{E}[\mathbf{r}(k)s(k-d)]. \quad (10.8)$$

The corresponding mean square error (MSE) is

$$\text{MSE} = \mathbf{E}[s^2(k-d)] - \hat{\mathbf{w}}^T \mathbf{E}[\mathbf{r}(k)s(k-d)]. \quad (10.9)$$

For the case of 2-ary PAM symbols, the bit error rate (BER) of the Wiener filter is given by

$$\text{BER} = \text{Prob}\{\hat{\mathbf{w}}^T \mathbf{r}(k) < 0 | s(k-d) = 1\}. \quad (10.10)$$

Monte Carlo simulation can be used to evaluate (10.10). In the simulation, the signal to noise ratio (SNR) is defined as $\text{SNR} = \mathbf{E}[r^2(k)]/\sigma_e^2$.

Assume the simple case of 2-ary PAM symbols and consider the channel with a transfer function

$$A(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}. \quad (10.11)$$

For the equaliser delay $d = 1$, Fig.10.3 shows the relationship between the MSE and the Wiener filter order under several SNR conditions while Fig.10.4 depicts the graph of the BER versus the Wiener filter order under the same SNRs. The results are better for $d = 2$ but the trends are identical to those shown in Figs. 10.3 and 10.4. For this example, there is no advantage in employing an adaptive LTE having an order $m > 4$. Clearly the noise enhancement severely limits the performance of the LTE.

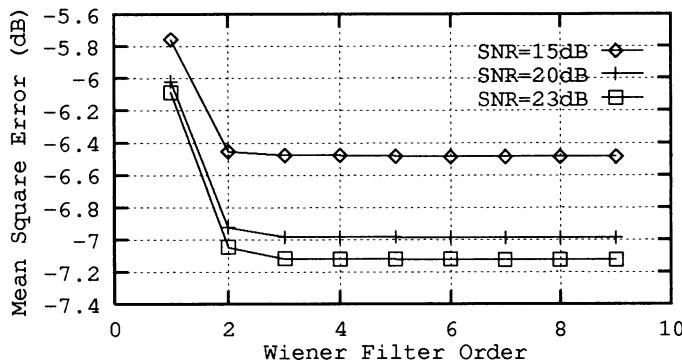


Figure 10.3 Mean square error versus Wiener filter order. Channel $0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$ and equaliser delay $d = 1$.

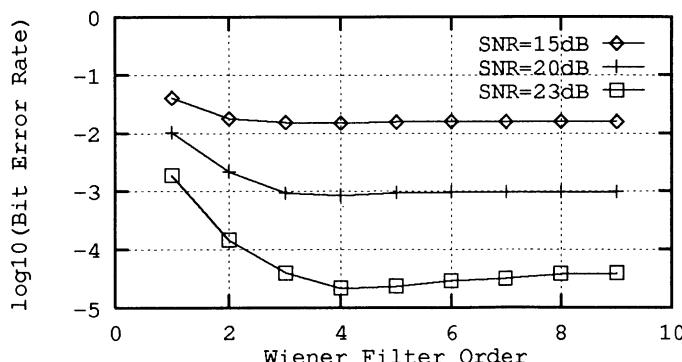


Figure 10.4 Bit error rate versus Wiener filter order. Channel $0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$ and equaliser delay $d = 1$.

Referring to Fig.10.2, if a more complex processing method is used instead of a linear filter, better performance can be achieved. A particular artificial neural network known as the MLP [19] provides a nonlinear processing capability. The network structure used for equalisation usually has two hidden layers [5]. Let the numbers of nodes in the first and second layers be n_1 and n_2 respectively. For simplicity, assume the 2-ary PAM case. A single output node is then sufficient, and the overall network response is given as

$$f_n(\mathbf{r}(k)) = \phi \left(\sum_{i=1}^{n_2} w_i^{(3)} \phi \left(\sum_{j=1}^{n_1} w_{ij}^{(2)} \phi \left(\sum_{l=1}^m w_{jl}^{(1)} r(k-l+1) + \mu_j^{(1)} \right) + \mu_i^{(2)} \right) + \mu_i^{(3)} \right), \quad (10.12)$$

where the superscripts indicate in which layer the nodes are, w and μ are the weights and thresholds respectively. The node's nonlinearity $\phi(\cdot)$ can be chosen as

$$\phi(x) = \tanh(x) = (1 - \exp(-2x))/(1 + \exp(-2x)). \quad (10.13)$$

Extension to the general case of M -ary PAM symbols can be obtained by using a suitably chosen $\phi(\cdot)$ [10]. The error signal

$$\varepsilon(k) = \begin{cases} s(k-d) - f_n(\mathbf{r}(k)), & \text{in training mode,} \\ \hat{s}(k-d) - f_n(\mathbf{r}(k)), & \text{in decision-directed mode,} \end{cases} \quad (10.14)$$

is used for network adaptation based on, for example, the backpropagation algorithm [20]. Although there are better learning algorithms for the MLP [21],[22], computational considerations favour a simple gradient algorithm.

Another nonlinear structure known as the VS filter can also be employed to form a nonlinear equaliser [7]. To avoid an exponentially increasing filter dimension, the VS equaliser is often truncated to the third order

$$\begin{aligned} f_v(\mathbf{r}(k)) = & \sum_{i=1}^m w_i r(k-i+1) + \sum_{i=1}^m \sum_{j=i}^m w_{ij} r(k-i+1) r(k-j+1) \\ & + \sum_{i=1}^m \sum_{j=i}^m \sum_{l=j}^m w_{ijl} r(k-i+1) r(k-j+1) r(k-l+1). \end{aligned} \quad (10.15)$$

This VS equaliser is linear in the parameters, and the least mean square (LMS) algorithm [3] can be employed to adjust these parameters. The expanded input vector consists of monomials of $r(k-i)$, $0 \leq i \leq m-1$. In

the 2-ary PAM case, the autocorrelation matrix of this input vector often becomes ill-conditioned and has a large eigenvalue spreading ratio. This numerical difficulty can be alleviated by passing the VS filter (10.15) through the sigmoid function (10.13) [7]. The resulting equaliser, however, is no longer linear-in-the-parameters.

Previous research [5]-[8] has demonstrated that the use of the MLP or the VS filter as equalisers offers significant performance improvement over the LTE. Before investigating why a nonlinear capability should enhance performance, some drawbacks of the MLP and VS equalisers are pointed out. An obvious drawback of these two nonlinear equalisers is that they require a much longer training period compared with an adaptive LTE. A less noticed disadvantage is that it is not known how to specify parameters of these two equalisers even when the channel statistics are given. Their parameters can only be set by experiment in an ad hoc manner. This is in contrast to the LTE whose parameters are completely specified by the Wiener solution when the channel statistics are provided. Recent research [12],[13] has shown how to overcome these two drawbacks by employing a different network structure called the RBF network [23],[24].

10.3 BAYESIAN TRANSVERSAL EQUALISER

The equalisation process depicted in Fig.10.2 can be viewed as a classification problem, which seeks to classify an observation vector $\mathbf{r}(k)$ into one of the known symbol points s_i , $1 \leq i \leq M$. The transmitted symbols that influence the equaliser decision at k are $\mathbf{s}(k) = [s(k) \cdots s(k - m - n_a + 2)]^T$. This channel input sequence has $n_s = M^{m+n_a-1}$ combinations. In the absence of noise, therefore, the noise-free channel output vector

$$\hat{\mathbf{r}}(k) = [\hat{r}(k) \cdots \hat{r}(k - m + 1)]^T \quad (10.16)$$

has n_s outcomes. The set of these states, denoted as $R_{m,d}$, can be partitioned into M subsets according to the value of $s(k - d)$

$$R_{m,d} = \bigcup_{1 \leq i \leq M} R_{m,d}^{(i)}, \quad (10.17)$$

where

$$R_{m,d}^{(i)} = \{\hat{\mathbf{r}}(k) | s(k - d) = s_i\}, \quad 1 \leq i \leq M. \quad (10.18)$$

The number of states in each $R_{m,d}^{(i)}$ is $n_s^{(i)} = n_s/M$. Because of the additive noise, the observed $\mathbf{r}(k)$ is a stochastic vector.

Bayes decision rule [25],[26] provides the optimal solution to the general decision problem and is applicable here. Compute M Bayesian decision variables

$$\eta_i(k) = \sum_{j=1}^{n_s^{(i)}} p_j^{(i)} p_e(\mathbf{r}(k) - \mathbf{r}_j^{(i)}), \quad 1 \leq i \leq M, \quad (10.19)$$

where $\mathbf{r}_j^{(i)} \in R_{m,d}^{(i)}$, $p_j^{(i)}$ are the a-priori probabilities of $\mathbf{r}_j^{(i)}$ and $p_e(\cdot)$ is the probability density function (p.d.f.) of $\mathbf{e}(k) = [e(k) \cdots e(k - m + 1)]^T$. Each $\eta_i(k)$ is the conditional p.d.f. of $\mathbf{r}(k)$ given $s(k - d) = s_i$. The minimum-error-probability decision is defined by

$$\hat{s}(k - d) = s_i^* \text{ if } \eta_{i^*}(k) = \max\{\eta_i(k), 1 \leq i \leq M\}. \quad (10.20)$$

The Bayesian decision procedure effectively partitions the m -dimensional observation space into M decision regions. When $\mathbf{r}(k)$ is within the i th region the decision $\hat{s}(k - d) = s_i$ is made. Since all the channel states can be assumed to be equiprobable, all the $p_j^{(i)}$ are equal. In particular, for $M = 2$, the Bayesian solution can be expressed as

$$\hat{s}(k - d) = \text{sgn}(f_B(\mathbf{r}(k))) = \text{sgn}(\eta_2(k) - \eta_1(k)), \quad (10.21)$$

where $\text{sgn}(\cdot)$ is the signum function and $f_B(\cdot)$ is called the Bayesian decision function. The set $\{\mathbf{r} | f_B(\mathbf{r}) = 0\}$ defines the Bayesian decision boundary.

Consider again the channel (10.11) with 2-ary PAM symbols. For the purpose of graphical display, assume $m = 2$ and let $d = 1$. All the combinations of $s(k)$ and the channel states are listed in Table 10.1. The states of $R_{2,1}^{(2)}$ and $R_{2,1}^{(1)}$ are plotted in Fig.10.5 using \square and \times respectively. The Bayesian decision boundaries for two given SNRs are depicted in Fig.10.5. This geometric picture can obviously be generalized to the case of higher dimension and more classes. In general, the Bayesian decision boundary is a hypersurface in the m -dimensional observation space. This geometric visualisation explains why a nonlinear capability is required, and it becomes clear that the MLP and VS equalisers attempt to realize this nonlinear decision boundary implicitly. The geometric interpretation of the equalisation process also highlights the limitation of the LTE. For this given channel and equaliser structure, the

Wiener solution for SNR=15dB is $[\hat{w}_1 \hat{w}_2] = [0.9856 - 0.2415]$, and the decision boundary of this Wiener filter is shown in Fig.10.6. The LTE can only realize a hyperplane decision boundary and, therefore, a performance gap inevitably exists between the optimal TE and the LTE.

$s(k)$	$s(k-1)$	$s(k-2)$	$s(k-3)$	$\hat{r}(k)$	$\hat{r}(k-1)$
1	1	1	1	1.5668	1.5668
1	1	1	-1	1.5668	0.8704
1	1	-1	1	0.8704	-0.1740
1	1	-1	-1	0.8704	-0.8704
-1	1	1	1	0.8704	1.5668
-1	1	1	-1	0.8704	0.8704
-1	1	-1	1	0.1740	-0.1740
-1	1	-1	-1	0.1740	-0.8704
1	-1	1	1	-0.1740	0.8704
1	-1	1	-1	-0.1740	0.1740
1	-1	-1	1	-0.8704	-0.8704
1	-1	-1	-1	-0.8704	-1.5668
-1	-1	1	1	-0.8704	0.8704
-1	-1	1	-1	-0.8704	0.1740
-1	-1	-1	1	-1.5668	-0.8704
-1	-1	-1	-1	-1.5668	-1.5668

Table 10.1 Binary symbol and channel states for channel $0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$. $d = 1$, $m = 2$ and $n = 2$; boldfaced: $\mathbf{s}_{f,j} = [1 \ 1]^T$.

The Bayesian TE solution can be realized explicitly using the RBF network. The RBF network [23],[24] is a one-hidden-layer neural network. Each hidden node in the RBF network has a radially symmetric response around a node parameter vector called a centre, and an output node is a linear combiner with weights. The overall response of the RBF network with m inputs and a single output node is defined as

$$f_r(\mathbf{r}(k)) = \sum_{i=1}^{n_h} w_i \psi(\|\mathbf{r}(k) - \mathbf{c}_i\|^2 / \rho_i), \quad (10.22)$$

where n_h is the number of hidden nodes, \mathbf{c}_i are the m -dimensional RBF centres, $\|\cdot\|$ denotes the Euclidean norm, ρ_i are the positive scalars known as the widths, w_i are the weights, and the node's nonlinearity $\psi(\cdot)$ has a radially symmetric shape.

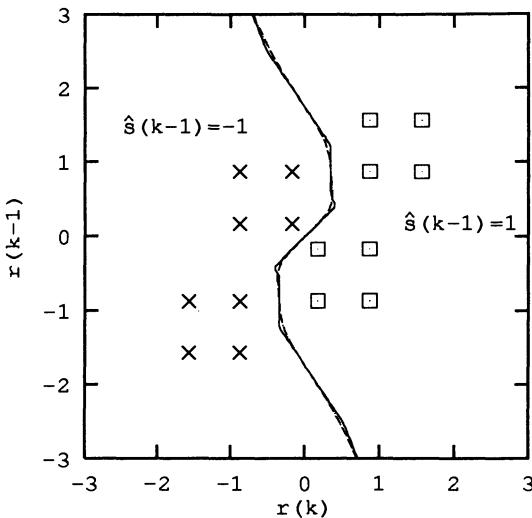


Figure 10.5 Bayesian decision boundaries for channel $0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$. Dashed: SNR=10dB, solid: SNR=20dB.

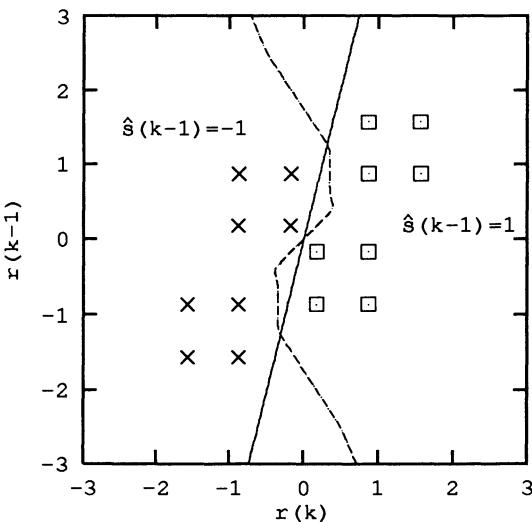


Figure 10.6 Decision boundaries for channel $0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$. SNR=15dB, dashed: Bayesian, solid: Wiener.

The Bayesian TE in the 2-ary PAM case has an identical structure to the above RBF network. Given channel statistics, it is known exactly how to specify all the parameters of the RBF network. The number of hidden nodes

is equal to the number of channel states and RBF centres are placed at these states. The node's nonlinearity is obviously chosen as the p.d.f. of the channel noise, which is generally Gaussian, and all the widths are set to $\rho = 2\sigma_e^2$. The weights are related to the a-priori probabilities of channel states and, because the states are equiprobable, the weights can be fixed to either 1 or -1 according to (10.21). Each hidden node implements a conditional p.d.f. for a given state, and the network response realizes precisely the Bayesian decision function $f_B(\cdot)$. The general Bayesian TE for the case of M -ary PAM can be realized by an extended RBF network. The network has two layers. The first layer consists of M subnets, each realizing a Bayesian decision variable in (10.19). The structure of a subnet is depicted in Fig.10.7. The second layer is a MAXNET [19], which selects the maximum Bayesian decision variable and thus implements the Bayesian decision (10.20).

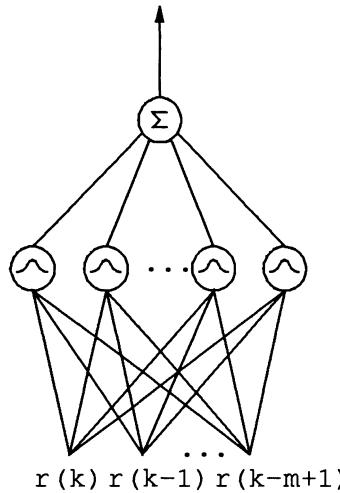


Figure 10.7 Subnet for computing a Bayesian decision variable.

The performance of a RBF equaliser does not depend crucially on the width ρ . This is evident in Fig.10.5, which shows two Bayesian decision boundaries corresponding to noise variances 0.1 and 0.01. In practice, an estimated noise variance is sufficient to set the width parameters. The realization of the Bayesian performance depends on knowing the channel states. Two adaptive schemes are available to obtain these vector states [12],[13]. The first method estimates the channel model $\mathbf{a} = [a_0 \cdots a_{n_a-1}]^T$ based on the LMS algorithm. When the channel model is known, it is straightforward to calculate the channel states $R_{m,d}$. The second method

estimates the channel states based on a clustering algorithm. The number of combinations of $\mathbf{s}_a(k) = [s(k) \cdots s(k - n_a + 1)]^T$ is $n_{a,s} = M^{n_a}$. Denote these input states as $\mathbf{s}_{a,i}$ and the corresponding one-dimensional channel states as r_i , where $1 \leq i \leq n_{a,s}$. Each r_i is the conditional mean of $r(k)$ given $\mathbf{s}_a(k) = \mathbf{s}_{a,i}$. Therefore, a clustering procedure can be used to update the scalar channel states. At sample k , if $\mathbf{s}_a(k) = \mathbf{s}_{a,i}$, r_i is adjusted according to

$$r_i(k) = r_i(k-1) + \mu * (r(k) - r_i(k-1)), \quad (10.23)$$

where $0 < \mu < 1$ is an adaptive gain. Once the scalar states r_i , $1 \leq i \leq n_{a,s}$, are obtained, it is straightforward to expand them into the set of the vector states $R_{m,d}$. Because the number of channel taps n_a is much smaller than that of the scalar channel states, the channel-estimator approach requires a shorter training sequence than the clustering approach. The former is therefore better suited for time-varying channels. The latter, however, does not assume a linear channel model and is immune from nonlinear channel distortion.

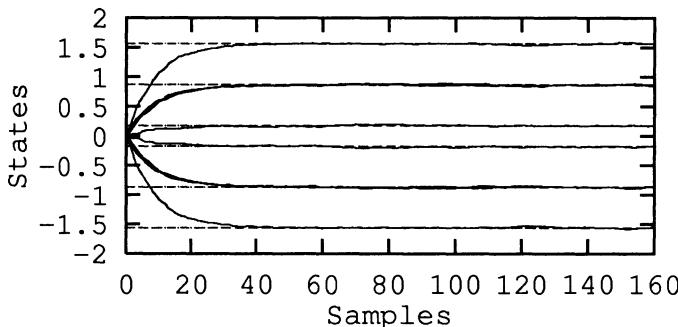


Figure 10.8 State trajectories obtained using channel-estimator scheme. Channel (10.11) with binary symbols and SNR=15dB.

The LMS channel estimator is a conventional adaptive algorithm. The properties of the LMS and the choice of adaptive gain are well-known and widely available in literature. The number of training samples for this adaptive scheme is the order of $10 \times n_a$, which is usually less than 100. Similarly the number of training samples for the clustering scheme is the order of $10 \times n_{a,s}$. For the binary case, therefore, a few hundred samples are sufficient for the clustering scheme. The behaviours of the two adaptive schemes are illustrated using the channel (10.11) with binary symbols and SNR=15dB. The number of scalar states is $n_{a,s} = 8$ but there are only 6 distinct scalar states. The trajectories of the estimated scalar channel states

obtained using the LMS channel estimator with an adaptive gain 0.1 are plotted in Fig.10.8 while Fig.10.9 shows the trajectories of the estimated scalar states obtained using the clustering algorithm with an adaptive gain 0.5. In the both cases, the dashed lines indicate the true values of the channel states and the trajectories were averaged over ensembles of 50 different runs.

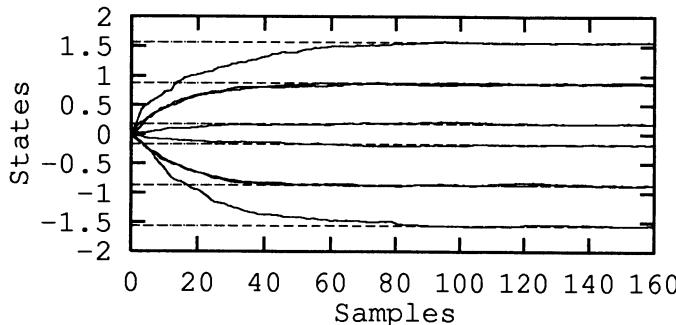


Figure 10.9 State trajectories obtained using clustering scheme. Channel (10.11) with binary symbols and SNR=15dB.

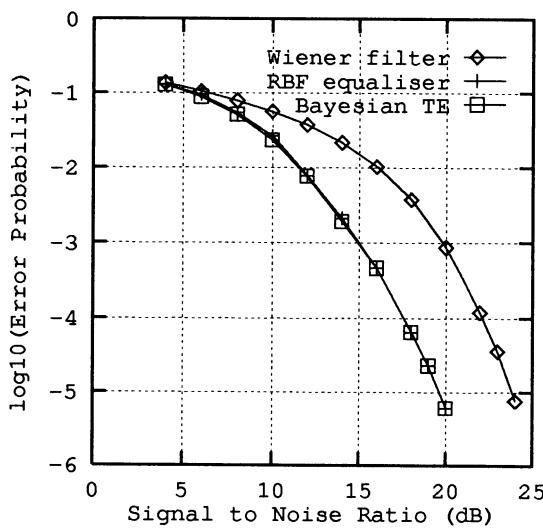


Figure 10.10 Performance comparison for linear channel (10.11). RBF equaliser uses the LMS with 90 training samples and adaptive gain 0.05.

The performance of the RBF equaliser is now compared with that of the LTE using 2-ary PAM symbols. In the first example, the channel is given in (10.11), and the equaliser has a structure of $d = 1$ and $m = 4$. The BER curves

of the Wiener solution and the Bayesian solution are plotted in Fig.10.10. The BER of the Wiener filter provides a lower bound for an adaptive LTE. The third curve in Fig.10.10 is obtained by a RBF equaliser, which employs a LMS channel estimator with 90 training samples and an adaptive gain 0.05. From Fig.10.10, it can be seen that the adaptive RBF equaliser realizes the Bayesian performance. In the second example, the channel output is distorted by a third-order Volterra nonlinearity, and the channel is defined by

$$\left. \begin{aligned} x(k) &= 0.3482s(k) + 0.8704s(k-1) + 0.3482s(k-2), \\ r(k) &= 1.1530x(k) + 0.2306x^2(k) - 0.1153x^3(k) + e(k). \end{aligned} \right\} \quad (10.24)$$

The equaliser structure is chosen as $d = 1$ and $m = 4$. Fig.10.11 shows the BERs of the adaptive LTE using the LMS with 120 training samples and an adaptive gain 0.02, the adaptive RBF equaliser using the clustering with 150 training samples and an adaptive gain 0.1, and the Bayesian TE.

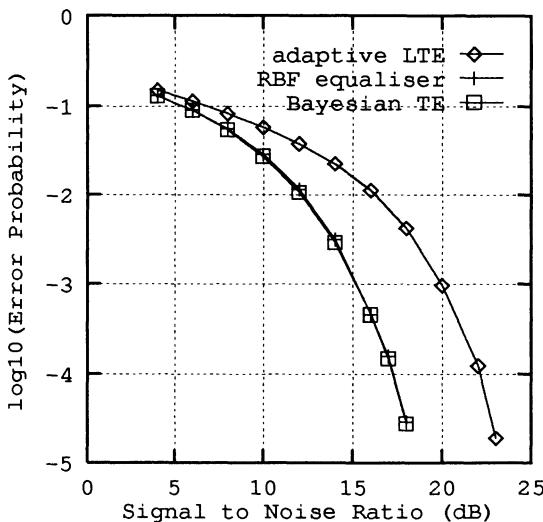


Figure 10.11 Performance comparison for nonlinear channel (10.24). LTE uses the LMS with 120 training samples and adaptive gain 0.02, RBF equaliser uses the clustering with 150 training samples and adaptive gain 0.1.

Compared with the MLP and VS equalisers, the RBF network design has several advantages. It realizes the Bayesian performance explicitly and requires a much shorter training period. Unlike the other two nonlinear equalisers, the structure and the parameters of the RBF network are

completely specified by the channel. A nonlinear TE offers significant performance improvement over the LTE at the cost of a considerable increase in computational complexity. A useful technique to improve equalisation performance is to use decision feedback [1]. It has been shown that decision feedback can be incorporated into the Bayesian approach, not only to improve performance, but also to minimize processing complexity [14]-[16].

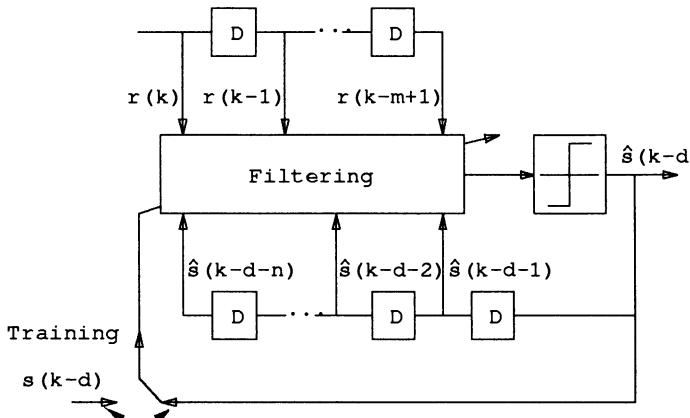


Figure 10.12 Schematic of a generic decision feedback equaliser.

10.4 DECISION FEEDBACK EQUALISER

The structure of a generic DFE is depicted in Fig.10.12, where the integer n is known as the equaliser feedback order. This structure is obtained by expanding the inputs of the TE to include past detected symbols

$$\hat{s}_f(k-d) = [\hat{s}(k-d-1) \cdots \hat{s}(k-d-n)]^T. \quad (10.25)$$

The conventional DFE [1] is based on a linear filtering of this expanded equaliser input vector and is defined by

$$f_l(r(k), \hat{s}_f(k-d)) = \mathbf{w}^T r(k) + \mathbf{b}^T \hat{s}_f(k-d), \quad (10.26)$$

where $\mathbf{b} = [b_1 \cdots b_n]^T$ are the feedback filter's coefficients. Under the assumption that the equaliser decisions are correct, $\mathbf{b}^T \hat{s}_f(k-d)$ can be designed to eliminate a large proportion of the ISI without enhancing noise. The feedforward filter $\mathbf{w}^T r(k)$ takes care of the remaining ISI. A nonlinear DFE based on the MLP has been shown to offer better performance over the conventional DFE [9],[10].

The equalisation process defined in Fig.10.12 can also be considered as a classification problem as in the case of TE, and Bayes decision rule can be used to derive its optimal solution. First notice that $\mathbb{S}_f(k-d)$ has $n_f = M^n$ states. Denote these feedback states as $\mathbf{s}_{f,j}$, $1 \leq j \leq n_f$. A subset of the channel states $R_{m,d}^{(i)}$ defined in (10.18) can further be partitioned into n_f subsets according to the feedback state

$$R_{m,d}^{(i)} = \bigcup_{1 \leq j \leq n_f} R_{m,d,j}^{(i)}, \quad (10.27)$$

with

$$R_{m,d,j}^{(i)} = \{\mathbf{r}(k) | s(k-d) = s_i \cap \mathbb{S}_f(k-d) = \mathbf{s}_{f,j}\}, \quad 1 \leq j \leq n_f. \quad (10.28)$$

The number of states in each $R_{m,d,j}^{(i)}$ is $n_{s,j}^{(i)} = n_s^{(i)}/n_f$. Under the assumption that correct decisions are fed back, the M Bayesian decision variables given $\mathbb{S}_f(k-d) = \mathbf{s}_{f,j}$ are

$$\eta_i(k | \mathbb{S}_f(k-d) = \mathbf{s}_{f,j}) = \sum_{l=1}^{n_{s,j}^{(i)}} p_l^{(i)} p_e(\mathbf{r}(k) - \mathbf{r}_l^{(i)}), \quad 1 \leq i \leq M, \quad (10.29)$$

where $\mathbf{r}_l^{(i)} \in R_{m,d,j}^{(i)}$. Comparing (10.29) with (10.19), it is obvious how decision feedback reduces computational complexity. Without feedback, all of the $n_s^{(i)}$ channel states are required to compute a decision variable. As a result of feedback, only a fractional number of these states are needed to compute a decision variable. The reduction factor in computational complexity owing to decision feedback is actually larger than n_f because a DFE requires a smaller feedforward order m than that which is needed without decision feedback.

How decision feedback improves performance has a simple geometric explanation. Any group of the M conditional subsets $R_{m,d,j}^{(i)}$, $1 \leq i \leq M$, is related to other groups of conditional subsets by some linear transformations. It is therefore sufficient to consider just a group of the M conditional subsets when examining the error rate of the Bayesian DFE. It is apparent that the minimum distance among the M conditional subsets $R_{m,d,j}^{(i)}$, $1 \leq i \leq M$, is larger than that among the M full subsets $R_{m,d}^{(i)}$, $1 \leq i \leq M$. This geometric property of DFE is illustrated using the channel and the equaliser structure

specified in Table 10.1, where the subset states given feedback $s_{f,j} = [1 \ 1]^T$ are emphasized in boldface. These subset states are depicted in Fig.10.13 using $+$. The Bayesian decision boundaries without feedback and with feedback $[1 \ 1]^T$ are also plotted in Fig.10.13. Because of feedback, fewer states are used in computing $f_B(\cdot)$, and the nonlinearity of the optimal decision boundary is milder and thus easier to realize. Fig.10.13 also shows that decision feedback greatly increases the minimum distance between the two classes of channel states.

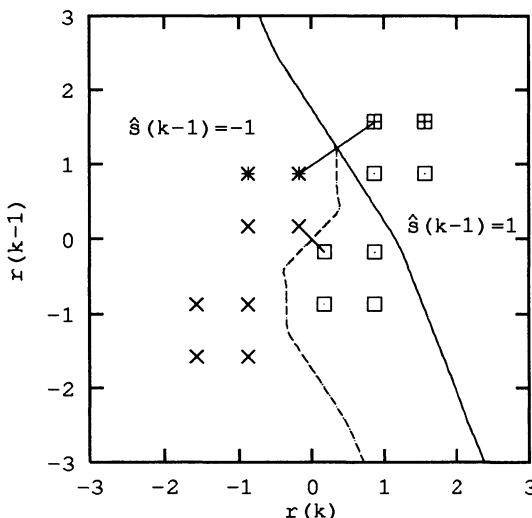


Figure 10.13 Comparison of decision boundaries for channel (10.11) with binary symbols and SNR=15dB. Dashed curve: Bayesian boundary without feedback, solid curve: Bayesian boundary with feedback $[1 \ 1]^T$, solid lines: minimum distances with and without feedback.

The structure of the Bayesian DFE is specified by equaliser delay d , feedforward order m and feedback order n . The basic structure parameter is d , which determines computational complexity (number of states required for decision variables). A pragmatic rule, which is often adopted in the conventional DFE, is to set $d = n_a - 1$. An explanation of this rule is that equaliser performance depends on energy of the channel taps a_0 to a_d covered by the decision delay [15]. Increasing d improves performance but there is no need to use $d > n_a - 1$. The maximum delay $d = n_a - 1$ is sufficient to achieve the full performance potential. However reducing d in turn reduces equaliser complexity, and the minimum complexity is achieved

when $d = 0$. In practice, if most of the channel energy is contained in the channel taps a_0 to a_q , where $q \leq n_a - 1$, the equaliser delay is set to $d = q$. Given d , it can be proven that $m = d + 1$ is sufficient, that is, a Bayesian DFE with $m = d + 1$ has the same performance as those with $m > d + 1$ [15]. The feedback order is then given by $n = n_a + m - d - 2 = n_a - 1$.

The theoretical performance of the Bayesian DFE is compared with that of the conventional DFE using the channel (10.11) and 4-ary PAM symbols. The Bayesian and conventional DFEs both have a structure of $d = 2$, $m = 3$ and $n = 2$. The coefficients of the conventional DFE are set to the Wiener solution. Fig.10.14 depicts the symbol error rates of these two DFEs using detected symbols as feedback. Adaptive implementation of the Bayesian DFE is similar to that of the Bayesian TE.

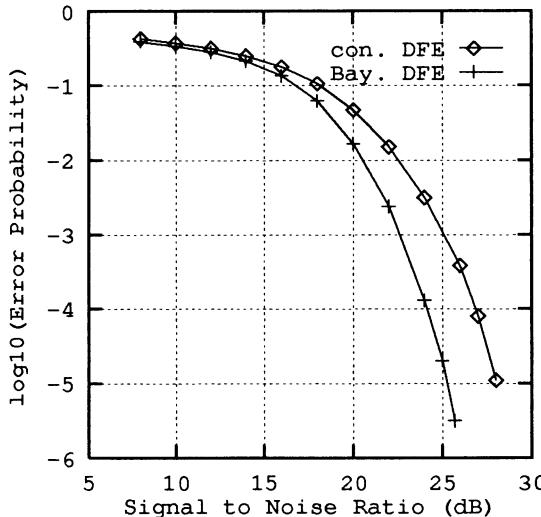


Figure 10.14 Performance comparison for channel (10.11) with 4-ary PAM symbols. Bayesian and conventional DFEs: $d = 2$, $m = 3$ and $n = 2$.

10.5 A COMPARISON WITH THE MLVA

The theoretical performance of the Bayesian equaliser is inferior to that of the MLVA since the former is only a symbol-decision solution. The MLVA is optimal because it makes decisions based on long-term likelihood functions, and this has an important implication for adaptive implementation. For highly

nonstationary channels, deviation of the adaptive MLVA from its theoretical performance can be very serious due to accumulation of tracking errors in likelihood functions [16]. By contrast the adaptive Bayesian DFE is very robust in a time-varying environment and, as a consequence, has superior performance over the adaptive MLVA. These results are illustrated using a simulation of a time-dispersive mobile radio channel.

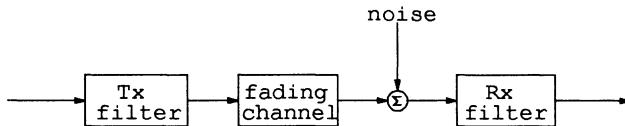


Figure 10.15 Baseband mobile radio channel simulator.

The structure of a baseband mobile radio channel simulator is depicted in Fig.10.15. Analogue signals are simulated and the channel is complex-valued. The modulation scheme is the quadrature amplitude modulation with 4 symbol points (4-QAM) [2], and the symbol rate is 300 kHz. The transmission pulse has a raised-cosine characteristic with a rolloff factor of 0.5 and is split equally between the transmitter and receiver filters. Multipath Rayleigh fading channels are simulated with a Doppler frequency of 100 Hz. Data are organized as blocks. A block consists of 20 symbols for training and 60 information symbols. The sampled receiver outputs are passed to the equaliser. When the sample rate is equal to the symbol rate, the equaliser feedforward section consists of symbol-spaced (SS) samples. When receiver outputs are sampled faster than symbol rate, the feedforward section can include fractional-spaced (FS) samples, and the resulting equaliser is called a FS equaliser [1]. Details of this computer simulator are given in [16].

A multipath mobile fading channel is simulated using this computer simulator. The root mean powers of the transmission paths are exponentially decaying, and the equivalent SS sampled channel model has the form

$$A(z) = z^{-\tau}(a_0(k) + a_1(k)z^{-1} + \dots + a_4(k)z^{-4}), \quad (10.30)$$

where τ is the transmission delay. The adaptive MLVA and the adaptive Bayesian DFE employ a LMS channel estimator to track the channel (10.30). In this nonstationary fading environment, the decision-directed adaptation is necessary and, therefore, the decision delay of the adaptive MLVA cannot be very large. The delay that gives the best result for the adaptive MLVA is

found to be 6 in this case. The adaptive Bayesian DFE has a structure of $d = 2$, $m = 3$ and $n = 4$. The conventional DFE is FS with 9 half SS feedforward terms and 4 SS feedback terms, and uses the LMS algorithm to adapt its coefficients. The symbol error rates of these three adaptive equalisers are depicted in Fig.10.16, where it can be seen that the adaptive Bayesian DFE is far superior in terms of performance over the other two adaptive equalisers. A more extensive simulation study is presented in [16].

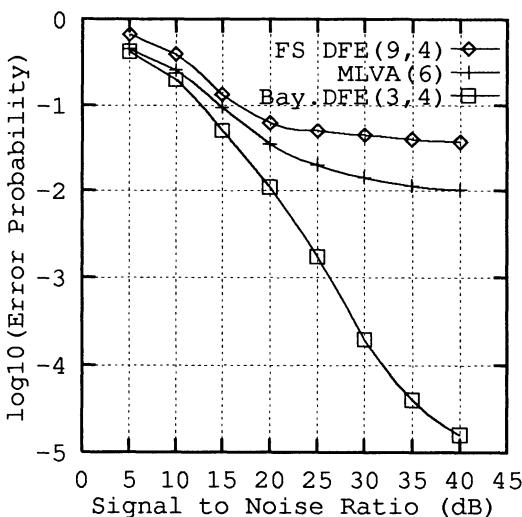


Figure 10.16 Performance comparison for a mobile radio channel.

10.6 CONCLUSIONS

In the recent years, artificial neural networks have been applied to adaptive channel equalisation with enhanced performance over traditional equalisation schemes. By adopting a Bayesian approach, it has been explained why the nonlinear ability of artificial neural networks is desired for equalisation process. Furthermore, this geometric understanding of equalisation process allows us to choose a most appropriate network structure for equalisation application. The RBF network has been shown to realize precisely the Bayesian equalisation solution. A further advantage of the Bayesian approach is that decision feedback can be utilized to reduce processing complexity.

The adaptive Bayesian equaliser based on the RBF network offers significant

performance improvement over the traditional adaptive linear scheme at the cost of a small increase in computational complexity. Typically, several dB improvement in SNR can be obtained by the former at the error probability level of 10^{-4} . Moreover training time for the adaptive Bayesian equaliser is comparable to that for the adaptive linear scheme. Although the MLVA achieves the best theoretical performance when the channel is known, it suffers serious performance degradation in a highly nonstationary environment. In contrast the adaptive Bayesian equaliser is very robust and can outperform the adaptive MLVA in this practical situation.

ACKNOWLEDGMENTS

This work was supported by the UK Science and Engineering Research Council. The author gratefully acknowledges the contributions of Professor P.M. Grant, Dr. B. Mulgrew and Dr. S. McLaughlin to the work described in this chapter.

REFERENCES

- [1] S.U.H. Qureshi, "Adaptive equalization," *Proc. IEEE*, Vol.73, No.9, pp.1349-1387, 1985.
- [2] J.G. Proakis, *Digital Communications*. New York: McGraw-Hill, 1983.
- [3] C.F.N. Cowan and P.M. Grant, *Adaptive Filters*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
- [4] R.W. Lucky, "Automatic equalization for digital communication," *Bell Syst. Tech. J.*, Vol.44, pp.547-588, 1965.
- [5] G.J. Gibson, S. Siu and C.F.N. Cowan, "Application of multilayer perceptrons as adaptive channel equalisers," in *Proc. ICASSP* (Glasgow, Scotland), 1989, pp.1183-1186.
- [6] S. Chen, G.J. Gibson, C.F.N. Cowan and P.M. Grant, "Adaptive equalization of finite non-linear channels using multilayer perceptrons," *Signal Processing*, Vol.20, pp.107-119, 1990.

- [7] S. Chen, G.J. Gibson and C.F.N. Cowan, "Adaptive channel equalisation using a polynomial-perceptron structure," *Proc. IEE, Pt.I*, Vol.137, No.5, pp.257-264, 1990.
- [8] G.J. Gibson, S. Siu and C.F.N. Cowan, "The application of nonlinear structures to the reconstruction of binary signals," *IEEE Trans. Signal Processing*, Vol.39, No.8, pp.1877-1884, 1991.
- [9] S. Siu, G.J. Gibson and C.F.N. Cowan, "Decision feedback equalization using neural network structures," in *Proc. 1st IEE Int. Conf. Artificial Neural Networks* (London), 1989, pp.125-128.
- [10] M. Peng, C.L. Nikias and J.G. Proakis, "Adaptive equalization with neural networks: new multilayer perceptron structures and their evaluation," in *Proc. ICASSP* (San Francisco), 1992, Vol.II, pp.301-304.
- [11] S. Chen, G.J. Gibson, C.F.N. Cowan and P.M. Grant, "Reconstruction of binary signals using an adaptive radial-basis-function equalizer," *Signal Processing*, Vol.22, No.1, pp.77-93, 1991.
- [12] S. Chen and B. Mulgrew, "Overcoming co-channel interference using an adaptive radial basis function equaliser," *Signal Processing*, Vol.28, No.1, pp.91-107, 1992.
- [13] S. Chen, B. Mulgrew and P.M. Grant, "A clustering technique for digital communications channel equalisation using radial basis function networks," *IEEE Trans. Neural Networks*, to appear, 1992.
- [14] S. Chen, B. Mulgrew and S. McLaughlin, "Adaptive Bayesian decision feedback equaliser based on a radial basis function network," in *Proc. ICC* (Chicago), 1992, pp.343.3.1-343.3.5.
- [15] S. Chen, S. McLaughlin and B. Mulgrew, "Complex-valued radial basis function network, Part II: application to digital communications channel equalisation," submitted to *Signal Processing*, 1992.
- [16] S. Chen, S. McLaughlin, B. Mulgrew and P.M. Grant, "Adaptive Bayesian decision feedback equaliser for dispersive mobile radio channels," submitted to *IEEE Trans. Communications*, 1992.
- [17] G.D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Information Theory*, Vol. IT-18, No.3, pp.363-378, 1972.
- [18] M.V. Eyuboglu and S.U.H. Qureshi, "Reduced-state sequence estimation with set partitioning and decision feedback," *IEEE Trans.*

Communications, Vol.36, No.1, pp.13-20, 1988.

- [19] R.P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, Vol.4, 1987.
- [20] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D.E. Rumelhart and J.L. McClelland, Eds., Cambridge, MA: MIT Press, 1986, pp.318-362.
- [21] S. Chen, C.F.N. Cowan, S.A. Billings and P.M. Grant, "Parallel recursive prediction error algorithm for training layered neural networks," *Int. J. Control*, Vol.51, No.6, pp.1215-1228, 1990.
- [22] S. Chen and S.A. Billings, "Neural networks for non-linear dynamic system modelling and identification," *Int. J. Control*, Vol.56, No.2, pp.319-346, 1992.
- [23] D.S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, Vol.2, pp.321-355, 1988.
- [24] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, Vol.78, No.9, pp.1481-1497, 1990.
- [25] D.F. Specht, "Generation of polynomial discriminant functions for pattern recognition," *IEEE Trans. Electronic Computers*, Vol. EC-16, No.3, pp.308-319, 1967.
- [26] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*. New York: John Wiley, 1973.

TD-GAMMON: A SELF-TEACHING BACKGAMMON PROGRAM

Gerald Tesauro

*IBM Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598 USA
(tesauro@watson.ibm.com)*

ABSTRACT

This chapter describes TD-Gammon, a neural network that is able to teach itself to play backgammon solely by playing against itself and learning from the results. TD-Gammon uses a recently proposed reinforcement learning algorithm called TD(λ) (Sutton, 1988), and is apparently the first application of this algorithm to a complex nontrivial task. Despite starting from random initial weights (and hence random initial strategy), TD-Gammon achieves a surprisingly strong level of play. With zero knowledge built in at the start of learning (i.e. given only a “raw” description of the board state), the network learns to play the entire game at a strong intermediate level that surpasses not only conventional commercial programs, but also comparable networks trained via supervised learning on a large corpus of human expert games. The hidden units in the network have apparently discovered useful features, a longstanding goal of computer games research.

Furthermore, when a set of hand-crafted features is added to the network’s input representation, the result is a truly staggering level of performance: TD-Gammon is now estimated to play at a strong master level that is extremely close to the world’s best human players. We discuss possible principles underlying the success of TD-Gammon, and the prospects for successful real-world applications of TD learning in other domains.

1 INTRODUCTION

Most of the successful real-world applications of neural networks to date have employed *supervised* learning procedures, in which the learner is given a “teacher signal” which indicates the correct output for every input pattern seen during training. In contrast, this chapter focuses on training networks by *reinforcement* learning meth-

ods. Reinforcement learning provides less information to work with than supervised learning: instead of a detailed teacher signal giving the correct output for every input, the learner is only given a “reward” or “reinforcement” signal indicating the quality of the learner’s output. Usually this is only a scalar signal, and in many cases, it is just a single binary bit indicating whether the learner’s output is right or wrong. Reinforcement learning is also often studied under conditions of delay of the reward signal, in which case the learner produces a sequence of outputs and is rewarded at the end of the sequence. In this type of situation, the learner has to solve a *temporal credit assignment* problem, i.e., it must apportion credit and blame at different points in time, as well as the usual *structural credit assignment* problem of figuring out which weights to change.

As one might expect, reinforcement learning is a more difficult and challenging learning paradigm than supervised learning. As a result, our theoretical understanding of reinforcement learning lags behind that of supervised learning, and there are fewer successes in the area of practical applications. Yet the appeal of a machine being able to learn on its own, without the aid of an intelligent teacher, continues to draw considerable interest in the subject. This is not only of great intellectual interest, but could also have numerous practical applications in real-world prediction and control tasks in which it would be difficult or impossible to define an appropriate teacher signal.

A common approach to reinforcement learning problems involves Dynamic Programming (DP) methods (Bertsekas, 1987). The basic idea is to construct a lookup table for every state in the state space, and by working backwards from the terminal states, to fill the table with exact expected reward values for every possible state. DP is well understood theoretically, and has achieved some practical successes, but it can run into problems for very large state spaces. For many real-world problems, the process of computing the values of every possible state, and separately storing all the results, can easily exceed by many orders of magnitude the CPU and storage capacity of available computers.

A neural network approach to reinforcement learning, on the other hand, offers the potential to overcome this limitation of DP. Multilayer neural networks have a demonstrated ability to approximate nonlinear functions and to generalize well for inputs not seen during training. This means that conceivably, a neural network learning a complex task by reinforcement learning might only need to see during training an infinitesimal fraction of all possible states. The nonlinear function learned by the network might then generalize acceptably well to the rest of the space. Unfortunately there are no theoretical guarantees of this. Most of the existing theoretical analyses of connectionist reinforcement learning algorithms, such as the Adaptive Heuristic Critic (Sutton, 1984) and the more recent $TD(\lambda)$ algorithm (Sutton, 1988), do not address

the situation of multilayer networks that learn nonlinear functions by generalization. Instead, the analyses focus on the much simpler situation of single-layer nets learning a linear target function by visiting every state in the state space infinitely often.

In the absence of theoretical guarantees, the approach taken in this chapter is to apply a connectionist reinforcement learning approach to a specific nonlinear test problem, in an attempt to gain empirical insight into the behavior and capabilities of the learning procedure. This is along the lines of the work of Barto, Sutton and Anderson (1987), who applied the Adaptive Heuristic Critic algorithm to a relatively small-scale cart-pole balancing problem.

The algorithm studied in this chapter is Sutton's $\text{TD}(\lambda)$ procedure. This algorithm has the following form:

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^t \lambda^{t-k} \nabla_w P_k \quad (1.1)$$

where P_t is the network's output upon observation of input pattern x_t at time t , w is the vector of weights that parameterizes the network, and $\nabla_w P_k$ is the gradient of network output with respect to weights. Equation 1.1 basically couples a temporal difference method for temporal credit assignment with a gradient-descent method for structural credit assignment; thus it provides a way to adapt supervised learning procedures such as back-propagation (Rumelhart et al., 1986) to solve temporal credit assignment problems. The λ parameter interpolates between two limiting cases: $\lambda = 1$ corresponds to an explicit supervised pairing of each input pattern x_t with the final reward signal, while $\lambda = 0$ corresponds to an explicit pairing of x_t with the next prediction P_{t+1} .

By applying $\text{TD}(\lambda)$ to a complex, large-scale test problem in which generalization is essential, we hope to discover empirically whether multilayer neural networks can in fact learn complex nonlinear functions by reinforcement learning. We also hope to address basic algorithmic issues such as convergence, scaling, and the possibility of overtraining or overfitting. $\text{TD}(\lambda)$ has been proven to converge only for a linear network and a linearly independent set of input patterns (Sutton, 1988; Dayan, 1992). In the more general case, the algorithm may not converge even to a locally optimal solution, let alone to a globally optimal solution.

Regarding scaling, no results are available to indicate how the speed and quality of TD learning will scale with the temporal length of sequences to be learned, the dimensionality of the input space, the complexity of the task, or the size of the network. Intuitively it seems likely that the required training time might increase very dramatically, possibly exponentially, with the sequence length. The training time might

also scale poorly with the network or input space dimension, e.g., due to increased sensitivity to noise in the teacher signal. Another potential problem is that the quality of solution found by gradient-descent learning relative to the globally optimal solution might get progressively worse with increasing network size.

Overtraining and overfitting are important factors in the practical use of supervised learning procedures, and could be equally important in reinforcement learning. Overtraining occurs when continued training of the network results in poorer performance. Overfitting occurs when a larger network does not do as well on a task as a smaller network. In supervised learning, both of these problems are believed to be due to a limited data set. In the TD approach, training takes place on-line using patterns generated *de novo*, thus one might hope that these problems would not occur. But both overtraining and overfitting may occur if the error function minimized during training does not correspond to the performance function that the user cares about. For example, in a combined prediction-control task, the user may care only about the quality of control signals, not the absolute accuracy of the predictions. Overtraining and overfitting might also occur if the distribution of training and testing patterns are different: the network could adapt very well to one distribution, but perform poorly on other distributions. Again no theoretical results are available concerning either of these issues in the context of reinforcement learning in the general nonlinear case.

2 A CASE STUDY: TD LEARNING OF BACKGAMMON STRATEGY

We have seen that existing theory provides little indication of how $TD(\lambda)$ will behave in practical applications. In the absence of theory, we now examine empirically the above-mentioned issues in the context of a specific application: learning to play the game of backgammon from the outcome of self-play. Some of the details of the experimental protocol and early results are described more fully in (Tesauro, 1992).

Complex board games such as checkers, chess, Othello and backgammon have been widely studied as testing grounds for various machine learning procedures (Samuel, 1959; Samuel, 1967; Griffith, 1974; Quinlan, 1983; Mitchell, 1984; Frey, 1986; Christensen and Korf, 1986; Lee and Mahajan, 1988; Tesauro and Sejnowski, 1989; Tesauro, 1990). Several of these studies have employed temporal difference learning methods.

Unlike checkers, chess, and Othello, backgammon is a nondeterministic game in which the players take turns rolling dice and moving their pieces around the board as allowed

by the dice roll. The first player to move all of his pieces around and off the board wins the game. The game is complicated because it is possible to “hit” opponent pieces and send them to the far end of the board, and to form blocking configurations that impede the forward movement of opponent pieces. These facts lead to a number of subtle and complex strategies and tactics at the highest levels of play (Magriel, 1976).

Additional complexity is introduced in the scoring in that it is possible to win more than one point per game. There are two ways in which this happens: (a) There are bonus points awarded if one player gets all of his checkers off before the opponent removes any checkers; this is called winning a “gammon” and the point value awarded is double the normal value. (b) There is also a rule that allows either player to offer to double the stakes at any point during the course of a game. The opponent can either accept the double, in which case he gets exclusive right to make the next double, or he can refuse, in which case the game is over and he loses the current stake.

Due to its considerable complexity and stochastic nature, backgammon is an attractive test vehicle for studying TD learning approaches. In fact, due to the stochastic dice rolls, the evolution of board states can be characterized as an absorbing Markov process, which is precisely the type of situation for which $TD(\lambda)$ was designed. It is also possible to make a detailed comparison of TD learning with the alternative approach of supervised learning from expert examples, which was used previously in the development of the Neurogammon (Tesauro, 1990), a program that convincingly won the backgammon championship at the 1989 International Computer Olympiad.

The TD backgammon learning system is set up as follows: the network observes a sequence of board positions x_1, x_2, \dots, x_f leading to a final reward signal z . In the simplest case the reward signal is $z = 1$ if White wins and $z = 0$ if Black wins. In this case the network’s output P_t is an estimate of White’s probability of winning from board position x_t . In the actual experiments reported here, the networks had four output units and the reward signal was a four-component signal in which two units represented the condition of a regular win for Black and White, and two units represented a bonus “gammon” win for Black and White. Thus the network’s four output units learn to estimate the probabilities of each of the four possible outcomes. (No constraint was imposed that the network’s four outputs should always sum to 1.) These games were played without doubling, thus the network did not learn anything about doubling strategy.

It seems plausible that the TD learning system described so far might work, provided that the playing strategy for the two sides is kept fixed. In other words, the TD network might be able to learn to predict, after observing a large enough number of samples, the expected outcome of any given position assuming that White plays a fixed strategy A and Black plays a fixed strategy B. However, the experiments presented here study

the more interesting question of whether a network can learn from its own play. This is accomplished by selecting the move at each time step that offers the highest expected number of points according to the network's four output units. (The expected number of points, also called the "equity" of the position, is given by: $\text{prob(regular win)} + 2 * \text{prob(gammon win)} - \text{prob(regular loss)} - 2 * \text{prob(gammon loss)}$, and is thus a linear combination of the four network outputs.) Since the network's strategy varies during learning, this is a more challenging learning task than simply learning to estimated expected outcome for a fixed strategy.

This procedure of letting the network learn from its own play was used even at the very start of learning, when the network's initial weights are random, and hence its initial strategy is a random strategy. From an *a priori* point of view, this methodology appeared unlikely to produce any sensible learning, because random strategy is exceedingly bad, and because the games end up taking an incredibly long time: with random play on both sides, games often last several hundred or even several thousand time steps. In contrast, in normal human play games usually last on the order of 50-60 time steps.

The input representation scheme used here contained only a simple encoding of the raw board description, i.e. the number of White or Black checkers at each location, and did not utilize any additional pre-computed features relevant to good play, such as, e.g. the probability of being hit or the strength of a blockade. Since the input encoding scheme contains no built-in knowledge about useful features, and since the network only observes its own play, we may say that this is a "knowledge-free" approach to learning backgammon. While it's not clear that this approach can make any progress beyond a random starting state, it at least provides a baseline for judging other approaches using various forms of built-in knowledge.

The approach described above is similar in spirit to Samuel's approach to learning checkers from self-play, but in several ways it is a more challenging learning task. One important difference is that Samuel's board description was in terms of a number of hand-crafted features, several of which were designed in consultations with human checkers experts. However, the networks studied here use only a raw board description and had no knowledge built into the input features. The evaluation function learned in Samuel's study was a linear function of the input variables, whereas multilayer networks learn more complex nonlinear functions. Also, the final reward signal in backgammon is noisy due to the dice rolls; this presumably makes the learning task more difficult than in noise-free games such as checkers. The branching ratios in backgammon are so large that look-ahead methods cannot be employed, whereas Samuel used search both in move selection and in calculation of the learning algorithm's error signal. Finally, Samuel found that it was necessary to give the learning system

at least one fixed intermediate goal, material advantage, as well as the ultimate goal of the game. The proposed backgammon learning system has no such intermediate goals.

The networks had a feedforward structure with full connectivity from one layer to the next. Two architectures were examined: a single-layer architecture with direct input-output connections and no hidden units, and a multilayer architecture containing a single hidden layer with 10-40 hidden units. The single-layer architecture can only represent linearly separable functions (Minsky and Papert, 1969), while the multilayer architecture, given sufficient hidden units, is capable of universal function approximation (Hornik, Stinchcombe and White, 1989). Both the hidden units and the output unit utilized a sigmoidal transfer function $y = 1/(1 + e^{-x})$. The learning algorithm parameters were set, after a certain amount of parameter tuning, at $\alpha = 0.1$ and $\lambda = 0.7$.

Learning was assessed primarily by testing the networks in actual game play against Sun Microsystems' Gammontool program. Gammontool is representative of the playing ability of typical commercial programs, and provides a decent benchmark for measuring game-playing strength: in contests without doubling, human beginners can win about 40% of the time against it, decent intermediate-level humans would win about 60%, and human experts would win about 75%. (The random initial networks before training win only about 1%.)

Networks were trained on the entire game, starting from the standard opening position and going all the way to the end. This is an admittedly naive approach which was not expected to yield any useful results other than a reference point for judging more sensible approaches. However, the rather surprising result was that a significant amount of learning actually took place. Results are shown in figure 1. For comparison purposes, networks with the same input coding scheme were also trained on a massive human expert data base of over 15,000 engaged positions, following the training procedure described in (Tesauro, 1989). These networks were also tested in game play against Gammontool.

Given the complexity of the task, size of input space and length of typical sequences, it seems remarkable that the TD nets can learn on their own to play at a level substantially better than Gammontool. Perhaps even more remarkable is that the TD nets surpass the EP nets trained on a massive human expert data base: the best TD net won 66.2% against Gammontool, whereas the best EP net could only manage 59.4%. This was confirmed in a head-to-head test in which the best TD net played 10,000 games against the best EP net. The result was 55% to 45% in favor of the TD net. This confirms that the Gammontool benchmark gives a reasonably accurate measure of relative game-playing strength, and that the TD net really is better than the EP net. In fact, the TD net with no features appears to be as good as Neurogammon 1.0, backgammon champion

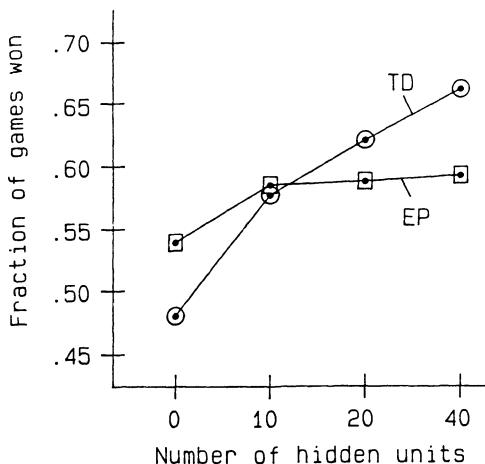


Figure 1 Plot of game performance against Gammontool vs. number of hidden units for networks trained using TD learning from self-play (TD), and supervised training on human expert preferences (EP). Each data point represents the result of a 10,000 game test, and should be accurate to within one percentage point.

of the 1989 Computer Olympiad, which does have features, and which wins 65% against Gammontool. A 10,000 game test of the best TD net against Neurogammon 1.0 yielded statistical equality: 50% for the TD net and 50% for Neurogammon.

The TD net's performance against these three opponents indicates that it has reached a significant level of playing ability. This violates a widely held belief in computer games and machine learning research that significant levels of performance in game-playing programs can only be achieved through the use of hand-crafted features in the evaluation function. Apparently the hidden units in the TD net have discovered useful features through self-play. When one looks at the pattern of weights learned by the TD net, one can see a great deal of spatially organized structure, and some of this structure can be interpreted as useful features by a knowledgeable backgammon player. Figure 2 shows the weights from the input layer to two of the hidden units in the best TD net. Both hidden units contribute positively to the estimation of Black's chances of winning and gammoning, and negatively to the estimation of White's chances of winning and gammoning. The first hidden unit appears to be a race-oriented feature detector, while the second hidden unit appears to be an attack or blitz-oriented feature detector.

The ability of the TD network to discover on its own useful features for game play provides an answer to the longstanding "feature discovery" problem in computer games research, which was recently stated in (Frey, 1986) as follows: "Samuel was

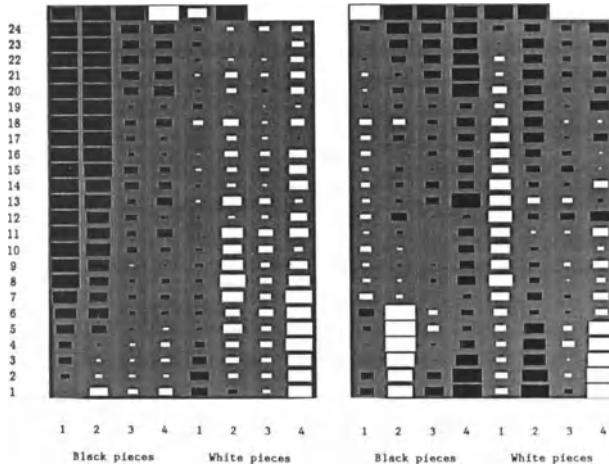


Figure 2 Weights from the input units to two hidden units in the best TD net. Black squares represent negative weights; white squares represent positive weights; size indicates magnitude of weights. Rows represent spatial locations 1-24, top row represents no. of barnmen, men off, and side to move. Columns represent number of Black and White men as indicated. The first hidden unit has two noteworthy features: a linearly increasing pattern of negative weights for Black blots and Black points, and a negative weighting of White men off and a positive weighting of Black men off. These contribute to an estimate of Black's probability of winning based on his racing lead. The second hidden unit has the following noteworthy features: strong positive weights for Black home board points, strong positive weights for White men on bar, positive weights for White blots, and negative weights for White points in Black's home board. These factors all contribute to the probability of a successful Black attacking strategy.

disappointed in his inability to develop a mechanical strategy for defining features. He thought that true machine learning should include the discovery and definition of features. Unfortunately, no one has found a practical way to do this even though more than two and a half decades have passed.”

The training times needed to reach the levels of performance shown in figure 1 were on the order of 50,000 training games for the networks with 0 and 10 hidden units, 100,000 games for the 20-hidden unit net, and 200,000 games for the 40-hidden unit net. Since the number of training games appears to scale roughly linearly with the number of weights in the network, and the CPU simulation time per game on a serial computer also scales linearly with the number of weights, the total CPU time thus scales quadratically with the number of weights: on an IBM RS/6000 model 320 workstation, the smallest network was trained in several hours, while the largest net required two weeks of simulation time.

Some qualitative observations on the styles of play learned by the TD and EP nets are worth noting. The TD nets have developed a style emphasizing running and tactical play. For example, they prefer to immediately split the back men rather than bringing down builders or slotting home board points. They are good in running game situations and in tactical situations such as blot-hitting contests and blitz attacks. The EP nets, however, favor a more quiescent positional style of play emphasizing blocking rather than racing. This is more in line with human expert play, but it often leads to complex prime vs. prime and back-game situations that are hard for the network to evaluate properly. This suggests one possible advantage of the TD approach over the EP approach: by learning to imitate an expert teacher, the learning system may get itself into situations that it doesn't know how to handle. With the alternative approach of learning from experience, the learner may not reproduce the expert strategies, but at least it will learn to handle whatever situations are brought about by its own strategy.

It's also interesting that TD nets ended up playing well in the early engaged phase of play, whereas play in the late racing phase was poor by comparison. This is contrary to the intuitive notion that in temporal credit assignment learning, states far from the end of the sequence will be harder to learn than states near the end. There are a number of possible explanations for this. One possibility is that the inductive bias due to the representation scheme and network architecture is more important than temporal distance to the final outcome. Another possible factor is that early positions always look similar to each other, since every game starts from an identical opening position, whereas there are a wide variety of different kinds of endgame position and thus the network's experience with any particular kind of endgame is correspondingly reduced.

3 TD LEARNING WITH BUILT-IN FEATURES: TD-GAMMON 1.0

We have seen that TD networks with no built-in knowledge are able to reach computer championship levels of performance for this particular application. It is then natural to wonder whether even greater levels of performance might be obtained by adding hand-crafted features to the input representation. This is certainly the case for training on expert preferences (Tesauro and Sejnowski, 1989; Tesauro, 1990): the hand-designed features in Neurogammon enable it to improve from 60% against Gammontool to 65%.

In a separate series of experiments, TD nets containing all of Neurogammon's features were trained from self-play as described in the previous section. Once again it was found that the performance improved monotonically by adding more hidden units to the network, and training for longer training times. The best performance was obtained

with a network containing 80 hidden units and over 25,000 weights. This network was trained for over 300,000 training games, taking over a month of CPU time on an RS/6000 model 320 workstation. (It is interesting to note by comparison that over the course of a 20-year professional career, a human master might expect to play a total of about 100,000 games.) The resulting level of performance was 73% against Gammontool and nearly 60% against Neurogammon. This is very close to a human expert level of performance, and appears to clearly surpass any previous computer program. This TD network constitutes the basis of version 1.0 of "TD-Gammon."

In addition to testing against other computer programs, TD-Gammon 1.0 was tested in late 1991 and early 1992 against three world-class human grandmasters: Bill Robertie and Paul Magriel, both noted authors and former World Champions, and Malcolm Davis, the 11th-highest rated player in the world in 1991. For the tests against humans, a heuristic doubling algorithm was added to the program which took TD-Gammon's equity estimates as input, and tried to apply somewhat classical formulas developed in the 1970's (Zadeh and Kobliska, 1977) to determine proper doubling actions.

The results of these tests against human grandmasters were very exciting and encouraging for the program in all three cases. In the match against Robertie, TD-Gammon put up a good fight but lost overall, winning 13 of the 31 games played to Robertie's 18 games. (Due to weaknesses in the doubling algorithm, which Robertie thought was significantly weaker than the checker play, the total point score showed a more lopsided victory for Robertie— he won 39 points to TD-Gammon's 20.) Robertie used a careful positional style of play reminiscent of the way Anatoly Karpov plays chess— he slowly accumulates small positional advantages until the opponent's position becomes untenable. TD-Gammon did well overall against this strategy, but the positional style of play did on occasion throw the program off. In contrast, Davis employs an aggressive attacking strategy, which turns out to be remarkably similar to the network's own style developed during self-training. Thus it seemed to be on very familiar ground playing against him, and turned in a very strong showing, winning 9 out of 13 games. (Again the point score including doubling was less in favor of the program— TD-Gammon won 13 points to 11 points for Davis.)

The match against Magriel was equally interesting— Magriel's strategy was to try to trick the program by steering the game into unusual lines of play where the program would be likely to make mistakes. While the program did in fact make some errors in these unusual situations, they may have been compensated by the weaknesses Magriel had to introduce into his own position in order to set up these situations. In any event, the plan did not result in overall victory for Magriel— TD-Gammon won 5 out of the 7 games played, and the total point score was 8-4 in favor of the program. Hence this may not be the best strategy to use against the program.

The cumulative result of all three test sessions was an overall victory for TD-Gammon in terms of the number of games won by each side. Of a total of 51 games played, TD-Gammon won 27 to the grandmasters' 24. In terms of total points scored, TD-Gammon lost by 13 points in 51 games, which represents an average loss of about 0.25 points per game. All three masters were impressed with the overall level of play of the program. They agreed that the strongest part of the program's game was its checker play in most normal early and middle game positions. (This covers the vast majority of positions that actually occur in real games.) The program generally plays these positions at expert level. The program's major weakness was its doubling algorithm, which on several occasions was substantially off target. It also made numerous minor technical errors in its endgame play, and it also significantly misunderstood back-game positions and complex positions where both sides have several men back. The latter two categories of positions, however, occur only rarely, and thus this defect is not expected to be very costly.

Taking all these factors into account, the overall assessment of the program's playing ability was that it would play at a decent advanced human level that is close to expert-level play. It would have a good chance to win in local or regional Open tournaments that are regularly attended by advanced and expert humans. This level of play would make TD-Gammon significantly better than any previous computer program, as explained in detail by Bill Robertie in an article published in *Inside Backgammon* magazine (Robertie, 1992). His estimate was that the program would lose on average in the range of 0.2 to 0.25 points per game against world-class human play. (This is consistent with the results of our 51-game sample.) In contrast, no commercial program had ever scored better than -0.66 points per game on this scale, and in fact most commercial programs lose well over one point per game. The best previous program of any sort was probably Hans Berliner's BKG program, which in its only public appearance in 1979 won a short match against the World Champion at that time (Berliner, 1980). BKG was about equivalent to a very strong intermediate or weak advanced player, and would have scored in the range of -0.3 to -0.4 points per game.

4 CURRENT STATUS OF TD-GAMMON

In 1992, TD-Gammon underwent considerable improvement from the level of play exhibited by version 1.0 of the program. One major improvement came about from further experimentation with different values of the λ parameter in the $\text{TD}(\lambda)$ algorithm. Experiments on small networks revealed that smaller values of λ work better— more training games are required to reach peak asymptotic performance, but as a result a

higher asymptotic performance is achieved. The best results seem to be obtained simply by setting $\lambda = 0$. This is somewhat surprising since in terms of the tradeoff between bias and variance, TD(0) is at the extreme end corresponding to maximum bias and minimum variance. Nevertheless, the experiments for small networks clearly indicated that $\lambda = 0$ gave an improvement of about 2-3% on the Gammontool benchmark compared to the previous networks trained with $\lambda = 0.7$, and this trend continued to hold for larger networks.

The other improvement was an increase in running speed of the program, both in learning mode and in performance mode. The latest generation of RS/6000 workstations are 2-3 times faster than the original model 320's, and furthermore, setting $\lambda = 0$ enables the learning program to run about twice as fast. The result is that substantially more training games can be played in a given amount of training time. Also, when the program is playing in performance mode, it is now fast enough to perform 2-ply search in real time. In other words, instead of just scoring every legal move and picking the one with the highest score, it also considers all possible opponent responses for all possible dice rolls. This gives a noticeable improvement in playing ability, although it does seem that the stronger the network already is at 1-ply, the smaller the improvement becomes by going to 2-ply search.

Both improvements went into the development of TD-Gammon 2.0, which made its public debut at the 1992 World Cup of Backgammon tournament held in Dallas. TD-Gammon 2.0 had a network with 40 hidden units which was trained for a total of 800,000 training games. Playing in 2-ply search mode, the program had a 53-47 edge over version 1.0 of the program. A total of 38 exhibition games were played against several world-class players in attendance at the tournament, including such luminaries as Kent Goulding, Kit Woolsey, Wilcox Snellings, former World Cup Champion Joe Sylvester, and former World Champion Joe Russell. The cumulative result of the exhibition games was a 19-19 tie in terms of the number of games won. In terms of the total points scored, the cumulative result was Humans 48, TD-Gammon 41, a net loss of 7 points in 38 games, or an average loss of -0.18 points per game. Most of the humans were favorably impressed with the program, and the general assessment of its play was a decent-to-strong Open level play, although a few humans were quite critical of the program's technical endgame errors and its inability to play back-games.

The current version of the program, version 2.1, was training back at the laboratory while version 2.0 was on exhibit at the World Cup. Version 2.1 has a network with 80 hidden units, and was trained for a total of 1.5 million training games, which represents about five times the experience of the original version 1.0. In 2-ply search mode, version 2.1 beats version 1.0 by a 55-45 margin. Its equity estimates have greatly improved in accuracy, resulting in a much stronger doubling algorithm, and its

ability to play and evaluate complex positions and back-game positions has improved dramatically.

As an example of this improvement, consider the following back-game position, shown in figure 3, taken from Bill Robertie's textbook *Advanced Backgammon* (Robertie, 1991). This is a well-timed back-game which is strong for White. Human expert rollouts indicate that White wins in this position over 60% of the time. When TD-Gammon 1.0 is given this position, its evaluation is way off: it thinks that White has only a 25% chance to win. TD-Gammon 2.1, however, realizes that White is the favorite in this position, and estimates White's winning chances at 54%. This represents an enormous improvement in the accuracy of the evaluation, and indicates that some genuinely significant learning took place during the network's million and a half training games. The raw position description and elementary features that TD-Gammon has to work with all indicate a very strong position for Black: Black has an enormous lead in the race, he has a strong four-point home board, he has five White men trapped behind a strong blockade, he has completely escaped White's blockade, and he has a nice safe position with no blots and no awkward stacks of checkers. Yet somehow TD-Gammon knows, as a result of its massive experience, that over the next several rolls, Black will have a great deal of difficulty bringing his checkers in safely, and will probably have to leave a repeated series of shots. Meanwhile, White will have several rolls to build up a menacing blockade that will securely trap any Black checker that happens to be hit, and enable White to win easily.

When Bill Robertie heard about the improvements in the latest version of the program, he was eager to return to the lab for a rematch. The rematch took place in January 1993, during which Robertie played a total of 40 games over a two-day period. The battle was a classic, hard-fought confrontation in which TD-Gammon acquitted itself rather well, especially in its ability to handle complex positions. The program jumped out to an early lead and held it until the very last game, when Robertie managed to claw his way back and achieve a narrow one-point victory by the score of 40-39. In terms of number of games won, Robertie had the edge by 23-17, but he attributed this to the luck of the dice more than anything else.

Robertie was very impressed with the new program, to say the least. He was particularly impressed with the accuracy of the doubling algorithm, and with the program's handling of complex positions. He thinks he may have actually learned some novel strategies from the program that improve upon strategies currently used by top human players.

Based on the 40-game sample, Robertie's overall assessment is that TD-Gammon now plays at a strong master level that is extremely close to equaling the world's best human players. In fact, due to the program's steadiness (it never gets tired or careless, as even the best of humans inevitably do), he thinks TD-Gammon would actually be

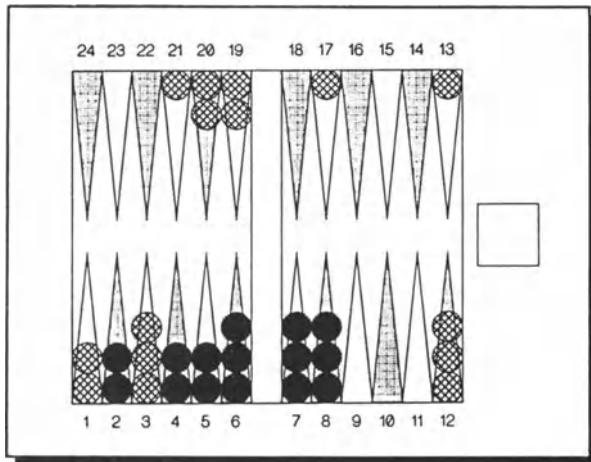


Figure 3 A well-timed back-game position showing a dramatic improvement in TD-Gammon's position evaluation. With human expert play, White wins this position over 60% of the time. TD-Gammon 1.0 estimates White's winning chances at only 25%. TD-Gammon 2.1, however, estimates White's winning chances at 54%, indicating an enormous improvement in its understanding of back-game positions.

the favorite against any human player in a long money-game session or in a grueling tournament format such as the World Cup competition.

The only thing which prevents TD-Gammon from genuinely equaling world-class human play is that it still makes minor, practically inconsequential technical errors in its endgame play. One would expect these technical errors to cost the program on the order of .05 points per game against top humans. Robertie thinks that there are probably only two or three dozen players in the entire world who, at the top of their game, could expect to hold their own or have an advantage over the program. This means that TD-Gammon is now probably as good at backgammon as the grandmaster chess machine Deep Thought is at chess. Interestingly enough, it is only in the last 5-10 years that human play has gotten good enough to rival TD-Gammon's current playing ability. If TD-Gammon had been developed 10 years ago, Robertie says, it would have easily been the best player in the world at that time. Even 5 years ago, there would have been only two or three players who could equal it.

Bill Robertie is not the only human master to give such a favorable assessment to the new program. Malcolm Davis has also seen the latest version, and he agrees that in a long session, no human could maintain the level of concentration needed to beat the program. Paul Magriel is also convinced that TD-Gammon represents a technology

that if developed further can easily surpass any human's ability to play the game. This is a rather dramatic conversion for him: before he saw TD-Gammon, he was quoted as saying that no computer program would play world-caliber backgammon in his lifetime.

5 CONCLUSIONS

The development of TD-Gammon can only be described as a smashing success for TD learning that greatly exceeds any possible expectations one could have had beforehand. Although there are no theoretical guarantees, the TD(λ) learning algorithm has performed magnificently by every conceivable standard: it has successfully trained a multilayer network on a complex, large-scale problem, it demonstrated reasonable convergence and scaling behavior, and the quality of solution has always improved with more hidden units and more training experience. It does so despite the fact that the strategy is initially random and varies during the course of learning. Significant learning takes place even when no specialized knowledge is built into the input representation.

For this particular application, self-teaching via reinforcement learning appears to be the methodology of choice for producing the highest quality computer program. No other approach, including conventional AI techniques for hand-crafting evaluation functions or supervised training on human expert examples, appears to offer the potential even to equal expert performance, let alone surpass it. TD-Gammon, however, is already very close to world-class human play, and it seems quite likely that with further increases in network size, training experience, and refinements of the learning algorithm, it will soon get to the point where no human will be able to contest it for the title of world's best backgammon player.

The most important research direction at this point, however, is not to push this particular approach until it surpasses human performance, but rather to extract the principles underlying its success, and to determine what kinds of other applications may also produce similar successes. At this point we are still largely ignorant as to why TD-Gammon is able to learn so well. However, one plausible conjecture is that the stochastic nature of the task is critical to the success of TD learning. One possibly very important effect of the stochastic dice rolls in backgammon is that during learning, they enforce a certain minimum amount of exploration of the state space. In contrast, a network teaching itself a deterministic task could quite conceivably end up exploring only a very narrow portion of the state space. It could even end up performing the exact same control actions over and over again, in a horribly bad endless loop. By

stochastically forcing the system into regions of state space that the current evaluation function wants to avoid, it is possible that improved evaluations and new strategies can be discovered.

Another important effect of the dice rolls could be that the ideal target prediction function (i.e. the true game-theoretic value of a position given perfect play) is a continuous, real-valued function representing an average over all possible sequences of dice rolls. In contrast, in deterministic games the game-theoretic value of a position is a binary function (win, lose), or a three-valued integer function in games where draws are allowed. It may well be the case that continuous prediction functions are easier to learn than binary prediction functions. This may be especially true if one is trying to learn from the results of imperfect play. The average results of an imperfect backgammon player, even a very bad one, might reasonably approximate the true equity of a position—this is why experts are willing to trust computer rollouts of positions. On the other hand, the results of an imperfect chess program could give very large errors with respect to the true game-theoretic values of positions.

The success of TD-Gammon suggests that it is at least worth trying out a TD learning approach for other large-scale, complex, real-world prediction and control problems. A natural assumption from the above discussion is that some sort of stochastic noise should be intrinsically present or externally injected during the learning process. One class of real-world tasks which has much in common with backgammon is in the area of learning to predict and develop trading strategies for financial markets such as stocks, bonds, and commodities. Another application area that is already receiving much attention from the reinforcement learning community is control tasks such as robot motor control, path planning, and navigation, as well as numerous kinds of manufacturing process control problems. Since reinforcement learning is often a slow process, it would be advantageous for these control applications if the network could be trained in simulation using a computer model of the plant, rather than the actual physical plant itself. This was an important factor in the success of the backgammon application— if the network had to use a physical board, dice and pieces, the learning times would have been many orders of magnitude longer. Finally, since modern military strategies are often studied and developed in computerized simulations of different kinds of war scenarios, it seems possible that TD learning systems could improve upon human strategies or develop novel strategies through massive self-training in computer-simulated war environments.

References

H. Berliner, “Computer backgammon.” *Scientific American* 243:1, 64-72 (1980).

- D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*. Englewood Cliffs NJ: Prentice Hall (1987).
- J. Christensen and R. Korf, "A unified theory of heuristic evaluation functions and its application to learning." *Proc. of AAAI-86*, 148-152 (1986).
- P. Dayan, "The convergence of $TD(\lambda)$ for general λ ." *Machine Learning* 8, 341-362 (1992).
- P. W. Frey, "Algorithmic strategies for improving the performance of game playing programs." In: D. Farmer et al. (Eds.), *Evolution, Games and Learning*. Amsterdam: North Holland (1986).
- A. K. Griffith, "A comparison and evaluation of three machine learning procedures as applied to the game of checkers." *Artificial Intelligence* 5, 137-148 (1974).
- K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators." *Neural Networks* 2, 359-366 (1989).
- K.-F. Lee and S. Majahan, "A pattern classification approach to evaluation function learning." *Artificial Intelligence* 36, 1-25 (1988).
- P. Magriel, *Backgammon*. New York: Times Books (1976).
- M. L. Minsky and S. A. Papert, *Perceptrons*. Cambridge MA: MIT Press (1969). (Republished as an expanded edition in 1988).
- D. H. Mitchell, "Using features to evaluate positions in experts' and novices' Othello games." Master's Thesis, Northwestern Univ., Evanston IL (1984).
- J. R. Quinlan, "Learning efficient classification procedures and their application to chess end games." In: R. S. Michalski, J. G. Carbonell and T. M. Mitchell (Eds.), *Machine Learning*. Palo Alto CA: Tioga (1983).
- B. Robertie, *Advanced Backgammon*. Arlington MA: Gammon Press (1991).
- B. Robertie, "Carbon versus silicon: matching wits with TD-Gammon." *Inside Backgammon* 2:2, 14-22 (1992).
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation." In D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing*, Vol. 1. Cambridge MA: MIT Press (1986).

- A. Samuel, "Some studies in machine learning using the game of checkers." *IBM J. of Research and Development* **3**, 210-229 (1959).
- A. Samuel, "Some studies in machine learning using the game of checkers, II – recent progress." *IBM J. of Research and Development* **11**, 601-617 (1967).
- R. S. Sutton, "Temporal credit assignment in reinforcement learning." Ph. D. Thesis, Univ. of Massachusetts, Amherst MA (1984).
- R. S. Sutton, "Learning to predict by the methods of temporal differences." *Machine Learning* **3**, 9-44 (1988).
- G. Tesauro and T. J. Sejnowski, "A parallel network that learns to play backgammon." *Artificial Intelligence* **39**, 357-390 (1989).
- G. Tesauro, "Connectionist learning of expert preferences by comparison training." In D. Touretzky (Ed.), *Advances in Neural Information Processing* **1**, 99-106. San Mateo, CA: Morgan Kauffmann (1989).
- G. Tesauro, "Neurogammon: a neural network backgammon program." *IJCNN Proceedings* **III**, 33-39 (1990).
- G. Tesauro, "Practical issues in temporal difference learning." *Machine Learning* **8**, 257-277 (1992).
- N. Zadeh and G. Kobliska, "On optimal doubling in backgammon." *Management Science* **23**, 853-858 (1977).

TEMPORAL DIFFERENCE LEARNING: A CHEMICAL PROCESS CONTROL APPLICATION

Scott Miller and Ronald J. Williams

*College of Computer Science
Northeastern University
Boston, MA 02115*

1 INTRODUCTION

1.1 Control and Reinforcement Learning

Learning to control can be considered a trial-and-error process in which the controlling agent explores the consequences of various actions. Actions that produce good results become reinforced while those that produce bad results are suppressed. Eventually, the best control actions become dominant over all others, resulting in an optimal solution to the control problem. Central to this approach is the existence of an appropriate performance measure, or *reinforcement function* that can distinguish good from bad consequences among possible control actions. Often, the control objective is specified as an operating setpoint, suggesting a simple reinforcement function based on distance to setpoint. Actions that result in states closer to the setpoint are assigned relatively higher reinforcement values, while those that result in states further from the setpoint are assigned relatively lower reinforcement values. Control of dynamical systems is complicated by time lags between control actions and their eventual consequences. In such systems, it is sometimes undesirable to move too rapidly toward the setpoint. Because of time lags between actions and consequences, it may be impossible to slow down in time once the controlled variable is moving rapidly toward the setpoint. The result is to overshoot. A controller that relies on a reinforcement function based only on distance to setpoint may never learn to control at all. Rather, it will approach the setpoint from one side, overshoot, approach from the other side, overshoot again, and thus forever oscillate. The problem is that the reinforcement function considers only local, short-term consequences of the controller's actions, but we really want the controller to choose actions based on their long-term consequences.

1.2 Learning to Predict Long-Term Consequences

There are two possible approaches to the problem of predicting long-term consequences. We illustrate both with a simplified example in which we are concerned only with making long-term predictions, temporarily ignoring the problem of control. We imagine a process that runs in batch mode over a predetermined time interval. The process can start in any of large number of initial states and proceed toward one of many possible final states. Our problem is to predict, at regular intervals as the process proceeds, the final state of the process. That is, we make an initial prediction at the beginning of the run, a second prediction a short time later, repeating until the end of the run. At each time step the objective is to predict as accurately as possible based on the currently available measurements. One approach is to save both predictions and measurements over the entire length of the run. When the final state becomes known, supervised learning can be used to correct each of the predictions made along the way. A drawback to this approach is its demand for storage; longer running times or more frequent predictions require proportionately more storage capacity. A second drawback involves computational requirements; no corrections are applied until the end of the run, at which time all predictions are corrected at once. It would obviously be desirable to spread out the computational load over the entire batch run rather than performing all computation upon completion. The second approach, known as *temporal differencing* [10], addresses both these issues. Rather than correcting each prediction based on the final state at the end of the run, each prediction is corrected based on the immediately succeeding prediction. That is, the second prediction is used to correct the first, the third is used to correct the second, repeating until the actual final state is used to correct the last prediction. Predictions and measurements need only be stored until the next prediction is made, and the computational load is spread over the entire run. As experience is gained over successive batch runs, the predictions produced by temporal differencing converge to the same values as performing all corrections at the end of each run. Furthermore, as demonstrated by Sutton [10], this technique can actually converge to these correct values faster in many situations than when all predictions are corrected toward the final outcome.

1.3 Infinite-Horizon Discounted Prediction

We next show that the method of temporal differencing is useful for prediction in processes that continue indefinitely as well as those with a definite final outcome. Once again, we illustrate with a simplified example. Imagine a process coupled to a fixed control system. The objective is to learn to predict how well the closed-loop system will regulate to a specified setpoint over all future measurements. Specifically, we wish to predict the sum of all future tracking errors. Since the process has no

definite ending time, the sum may be infinite. The solution is to introduce discounting, or a lower weighting, for tracking errors that occur further into the future. For example, we can assign a weight of 1 to the current tracking error, $1/2$ to the next, $1/4$ to the next, continuing on indefinitely. Although all future errors contribute to the weighted sum, only the first few contribute significantly.

More abstractly, let $r(t)$ denote *immediate reinforcement* received at time t . In this example it might represent some measure of proximity to desired setpoint. Then we define the *cumulative discounted reinforcement* at time t to be

$$\sum_{i=0}^{\infty} \gamma^i r(t+i), \quad (1.1)$$

where γ is a *discount factor* chosen to lie between 0 and 1. A sensible control objective that takes into account long-term consequences is to choose at time t whatever action maximizes this cumulative reinforcement, as we consider below, but for now we examine the more limited question of trying to determine this quantity at time t . This is a prediction problem since its actual value depends on immediate reinforcement received at all future times.

Let $P(t)$ denote a prediction of (1.1). Note that if $P(t)$ and $P(t+1)$ are both correct, then it follows that

$$P(t) = r(t) + \gamma P(t+1). \quad (1.2)$$

This shows how such predictions can be learned by a temporal difference method. Each such prediction is corrected using only currently available information and the value of the immediately succeeding prediction, by correcting the left-hand side of (1.2) to more nearly match the right-hand side.

The parameter γ essentially determines how many samples into the future are considered significant, whether for control or prediction, with values closer to 1 putting relatively more emphasis on longer-term effects. A reasonable guide to selecting γ can be based on using $1/(1 - \gamma)$ as a measure of the number of time steps into the future considered significant.

1.4 Q-Learning and Control

Learning to control is reducible to learning to predict. An optimal control action is one that leads to the most favorable predicted outcome. Q-learning [14] is an algorithm for learning to predict the long-term consequences among alternative actions. The Q-value of a state and an action represents an infinite-horizon discounted prediction

of the expected outcome of performing that action in that state, assuming that all subsequent control decisions select the best possible action. Let x denote the current state, a the current action, r the resulting immediate reinforcement, and y the resulting successor state. Then these Q-values satisfy the recursive relationship

$$Q(x, a) = r + \gamma \max_b Q(y, b).$$

Note that if we knew the correct Q-values, it would be straightforward to choose actions optimizing the cumulative discounted reinforcement since an optimal action for state x is any action a that maximizes $Q(x, a)$.

Learning estimates of these Q-values by temporal differencing is integrated with control as follows:

1. Given state x , select a control action a . Save x and a .
2. Apply control action a . This results in immediate reinforcement r and new state y .
3. Among all possible actions b , find one that maximizes $\hat{Q}(y, b)$ (where \hat{Q} denotes current Q-value estimates).
4. Correct the prediction mechanism so that $\hat{Q}(x, a)$ more nearly matches the current value of $r + \gamma \max_b \hat{Q}(y, b)$.
5. Loop to step 1.

1.5 Selecting Control Actions

How should control actions be selected? One answer is to always select actions with the highest estimated Q-values. This policy would produce optimal results except that true Q-values are never available to the controller. Instead, the controller must rely on estimates of Q-values which are updated incrementally during each control cycle. An estimate $\hat{Q}(x, a)$ is updated only when action a is actually performed in state x . Thus it is generally necessary to experiment with every action in every state to obtain sufficiently accurate estimates so that the true optimal action can eventually be identified in each state. Furthermore, the estimate $\hat{Q}(x, a)$ is updated according to $\hat{Q}(y, b)$, where y is the successor state to x under action a and b represents the best action in state y . Until a correct estimate $\hat{Q}(y, b)$ is available (which includes the knowledge of which action b is best), the estimate $\hat{Q}(x, a)$ may not be updated correctly even when action a is taken. Rather, it may be necessary to perform the same action in the same state many times before its Q-value is updated correctly.

This leads to a choice between exploration and quality of control. In order to achieve high quality control, the controller should select actions associated with maximal Q-estimates. However, in order to improve its Q-estimates, the controller should occasionally select actions with consequences not currently believed to be best. A simple compromise is to select actions with maximal Q-estimates most of the time, while occasionally selecting actions at random. A reasonable frequency for random actions is one in every ten control cycles.

1.6 Dyna Architecture

An interesting feature of the Q-learning algorithm is that it can learn an optimal control policy without constructing an internal model of the process it is controlling. That is, given a process state and an action, the controller is incapable of predicting the next state, although it is fully capable of predicting the long-term utility of applying the action given the state. However, the cost of not using an internal model is that learning can be slow. As just described, the controller must explore the same states and repeat the same experiences until the Q-values finally converge.

Dyna [11] addresses this issue by the addition of a world model component. As experience is gained during control, the controller learns an internal model of the external process. The controller then divides its time between performing real control actions on the external process and performing practice control actions on the internal world model. Q-learning is applied during practice cycles just as during real control cycles. Because repeated simulated experiences are substituted for repeated real experiences, less real experience is required for the Q-values to converge. Thus, learning is much faster.

2 THE APPLICATION

2.1 Bioreactor Benchmark Problem

We now describe simulation experiments in which reinforcement learning techniques were used to obtain an effective, near-optimal control policy for a challenging bioreactor benchmark problem suggested by Ungar [13]. The equations governing the simulation dynamics [1] are:

$$c_1(t+1) = c_1(t) + \Delta t \left[-c_1(t)w(t) + c_1(t)(1 - c_2(t))e^{c_2(t)/\gamma_1} \right]$$

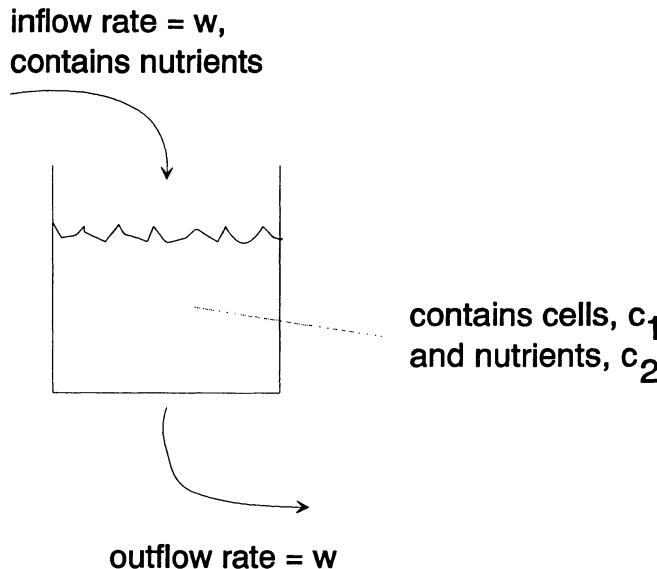


Figure 1 Schematic diagram of bioreactor.

$$c_2(t+1) = c_2(t) + \Delta t \left[-c_2(t)w(t) + c_1(t)(1 - c_2(t))e^{c_2(t)/\gamma_1} \frac{1 + \beta}{1 + \beta - c_2(t)} \right],$$

where c_1 represents the concentration of cells in the reactor, c_2 is related to the concentration of nutrients in the reactor, and w is the flow rate through the reactor. The dynamics of this system are highly nonlinear and involve substantial delays between control actions and system responses. Following Ungar, parameter values of $\beta = 0.02$, $\gamma_1 = 0.48$, and $\Delta t = 0.01$ were used, while the control interval is set to 50 samples. The control objective is to maintain a cell mass $c_1 = 0.2107$ by adjusting the flow rate w . One challenging aspect of the problem is that no open-loop stable solution exists for this target cell concentration level.

2.2 Relationship of Bioreactor Control Problem to a Maze Navigation Problem

An early demonstration of Dyna involved learning to solve a maze navigation problem [11]. One way to think about the present control problem is that the controller must learn to solve a bioreactor state-space navigation problem. There are many similarities

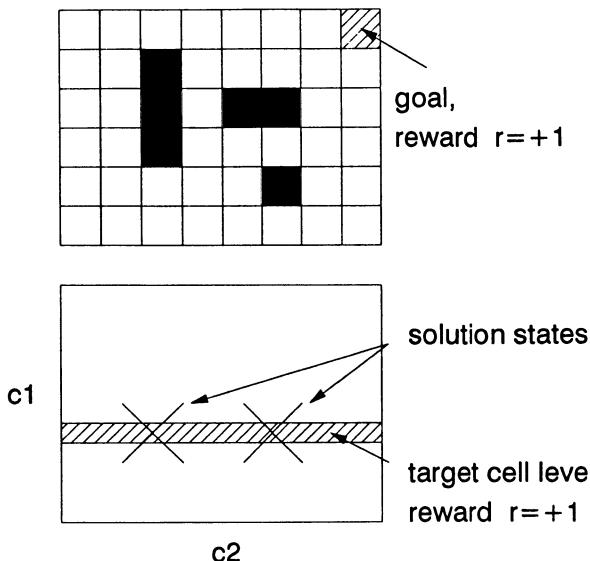


Figure 2 Top: for the maze problem, receipt of reward = +1 indicates that the maze is solved. Bottom: for the bioreactor, receipt of reward = +1 usually does not indicate a solution state.

between the problems, as well as some significant differences. For the maze problem, the goal is clearly defined. Upon entering the goal state, the controller receives an immediate reward of +1. For all other transitions, the reward is 0. The controller needs only to learn to navigate the maze so as to receive the reward as quickly as possible. Similarly, the bioreactor problem can be arranged so that a reward of +1 occurs whenever the concentration of cells in the reactor is sufficiently close to the target level. In the bioreactor case, however, state is defined by two variables: cell mass c_1 , and nutrient level c_2 . The controller must solve two distinct problems. First, it must identify a nutrient level c_2 such that the target cell mass c_1 can be maintained. Second, it must learn to navigate the state-space so as to quickly reach the identified solution state. The problem is further complicated by the fact that there are two distinct values for c_2 which are satisfactory.

Like the maze problem, the bioreactor exhibits significant time delays between control actions and responses. For the maze problem, no reward whatsoever is received until the final transition to the goal state. For the bioreactor problem, changes in cell mass lag significantly behind control actions. In fact, it is sometimes necessary for the controller to plot a course through state-space that initially moves away from the target cell mass in order to ultimately reach it.

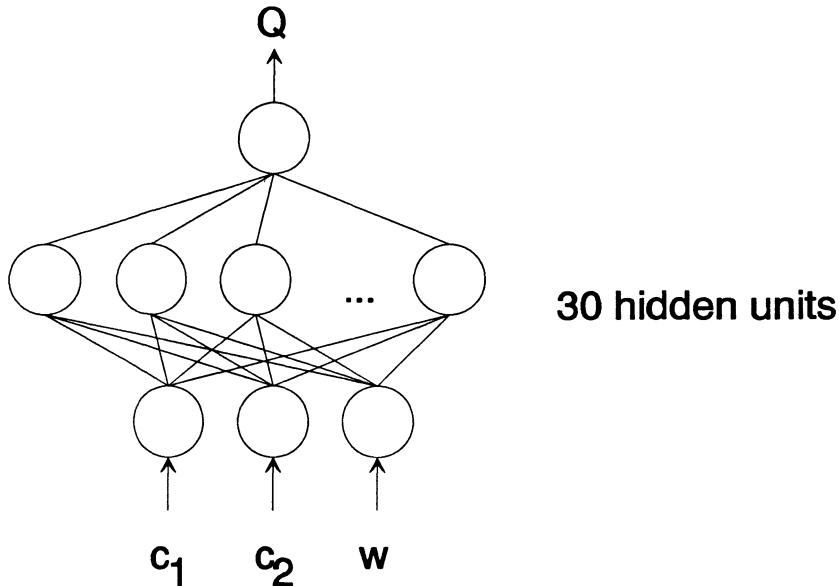


Figure 3 Q-values are learned by a backpropagation network.

Data structure requirements differ greatly between the maze and bioreactor problems. The entire search space of the maze problem is easily maintained in a tabular form, and is small enough to be fully explored. In contrast, the bioreactor state-space has a continuous, highly non-linear structure requiring some form of interpolation and generalization.

2.3 Representing and Learning the Q-function

For the bioreactor problem we used a backpropagation network with one hidden layer of 30 units to learn and maintain the Q-value estimates. This network has a single output unit and three input units, two for the state variables and one for the action.

To see how the network is trained, assume that the plant is in state (c_1, c_2) and that control action w is selected. Further, assume that the result of action w is a transition into state (c'_1, c'_2) and an immediate reward r . Then the training target for the network, when presented with input (c_1, c_2, w) is

$$Q_{\text{target}} = r + \gamma \max_a Q(c'_1, c'_2, a).$$

For this problem, setting $\gamma = 0.98$, corresponding to a horizon of 50 control cycles, gave good results. Training the network is somewhat difficult due to the complex shape that the bioreactor imposes on the Q-function, and because of the long time delays between control actions and changes in cell concentrations. A variety of techniques must be combined in order to overcome these problems.

2.4 States and Control Actions

State variables c_1 and c_2 are treated as continuous real numbers in the range [0..1]. Unfortunately, actions must be treated differently. Q-learning requires finding actions w such that $Q(c_1, c_2, w)$ is maximized for given values of c_1 and c_2 . In the present work, this was accomplished by exhaustively testing actions from the fixed set

$$W = \{0.50, 0.52, 0.54, \dots, 1.50\}.$$

2.5 Immediate Reward Function

A natural reinforcement function is based on the square of the error between the actual concentration of cells and the target level, or $r = -(0.2107 - c_1)^2$. Surprisingly, we found that learning was always unsuccessful when this reward function was used. Suspecting that this may have been due to the fact that this function did not delineate the target region sharply enough, we were led to use instead the following reinforcement function, a graph of which appears in Figure 4.

$$r = \begin{cases} 0 & \text{if } c_1 < 0.18 \\ 50c_1 - 9 & \text{if } 0.18 \leq c_1 < 0.2 \\ 1 & \text{if } 0.2 \leq c_1 < 0.22 \\ -50c_1 + 12 & \text{if } 0.22 \leq c_1 < 0.24 \\ 0 & \text{if } 0.24 \leq c_1 \end{cases}$$

This function was indeed found to give good results. There is some temptation to narrow the reinforcement region still further in order to achieve more accurate control. However, if the region is too narrow, high rewards are not sufficiently frequent and training fails.

2.6 World Model

In applying Dyna to a maze-navigation problem, the world model component occupies a useful though not essential role. By applying imaginary actions to the world model,

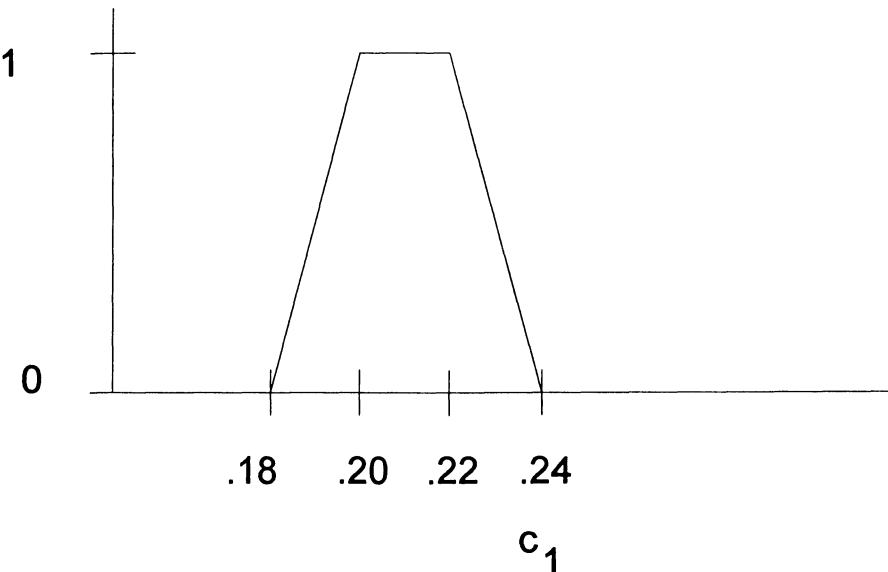


Figure 4 Reinforcement function.

Q-values are backed up throughout the search space without repeatedly visiting the same states many times. The result is a drastic reduction in the time required to learn the maze. For the bioreactor problem, the world model occupies a more essential role. Backpropagation networks can suffer from a fixation problem wherein the network focuses on recent training data while forgetting previous training inputs. In the case of the bioreactor, the problem is severe enough to prevent any useful learning. By randomly training on stored previous experiences in addition to new inputs, the fixation problem is overcome. In its present form, the world model is simply a table of 1500 remembered experiences of the form (state, action, next state), combined with a heuristic storage mechanism to maintain a reasonably uniform distribution over states

2.7 Exploring the State Space

All Q-learning systems rely upon exploration. Training the backpropagation network requires some extra care in selecting an exploration strategy. Early in the training, a great deal of exploration is required in order to have enough experience to begin shaping the Q-function. Later however, we have found that if too much exploration occurs the Q-values will not converge to correct values. In the present work, an annealing process was used to control exploration. During each control cycle, either

the control action with the highest Q-value or a random control action is selected. The probability of selecting a random action during control cycle t is given by

$$P_{\text{randomaction}}(t) = 0.5e^{t/1000}$$

2.8 Experimental Results

We conducted 25 separate trials of the learning system. In each trial, a successful controller was learned after approximately 10,000 control cycles, at which time exploration was disabled and training turned off to permit testing. Once trained, the controller always found and maintained one of the two solution states for the problem. Of the two, it exhibited a definite preference for $(c_1, c_2) = (0.2107, 0.2974)$, maintained by control actions close to $w = 1.306$. Occasionally, it found the other solution $(c_1, c_2) = (0.2107, 0.7226)$, maintained by control actions close to $w = 1.25$. Actual levels for c_1 ranged between 0.2 and 0.22, depending on differences in training sessions. Control within the target region was characterized by a small limit cycle. When control was disabled, and the system ran open-loop, large oscillations occurred, as shown in Figure 5. The simulation could be started from any region within the state-space except the extreme margins and it quickly and consistently reached a solution state. Figure 6 illustrates the behavior of the learned controller when started in various initial states.

3 DISCUSSION

There are several alternative ways that one may combine neural network and Q-learning techniques beyond those we have investigated here that are worth pointing out. Among these are the use of networks in which the hidden layer uses radial basis functions (e.g., [7]) instead of the sigmoid units we have used. Because such units respond strongly to a more limited region in the state \times action space, training tends to be more localized and changes made in one part of the space have little effect on other parts of the space. In contrast, adjustments to the sigmoid functions we have used has effects across large portions of this space. The potential advantage of using radial basis functions is thus that there is less spatial interference as learning proceeds. Although we have not tried them on this task, it is possible that our strategy of retraining the network on past transitions would not be necessary if radial basis functions were used.

Also, while we have performed the maximization over actions by explicitly limiting the actions to a discrete set, it is possible to search for this maximum over a continuous interval use gradient techniques. In particular, when the Q-function is computed by

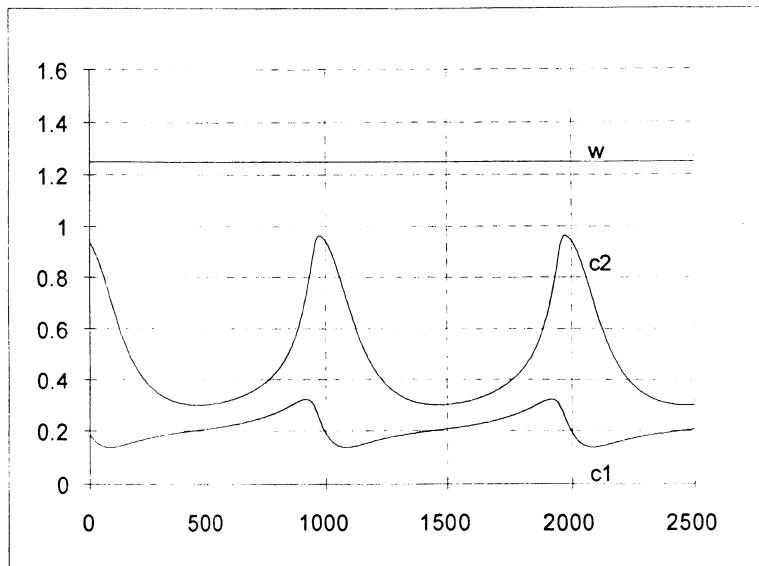


Figure 5 Open loop, uncontrolled response using a constant flow rate $w = 1.25$. Run length = 2500 samples.

a network like that we have used, it is possible to compute the gradient of the Q-value with respect to the action input using backpropagation. This is just a particular application of a general approach described by Williams [16], Munro [8], and Jordan and Jacobs [5], among others.

We note that it would be interesting to try in this application to maximize the cell concentration c_1 on average instead of trying to achieve a given target concentration. We do not know what would happen if this were done, but it would seem to be a natural thing to investigate. In this case the immediate reinforcement at each instant could be taken simply to be c_1 itself.

4 CONCLUSION

The experiments described here have demonstrated that reinforcement learning techniques can be successful at finding near-optimal control policies for interesting non-linear learning control problems. However, our experience has also demonstrated to us that successful application of such methods may require careful tuning of learning

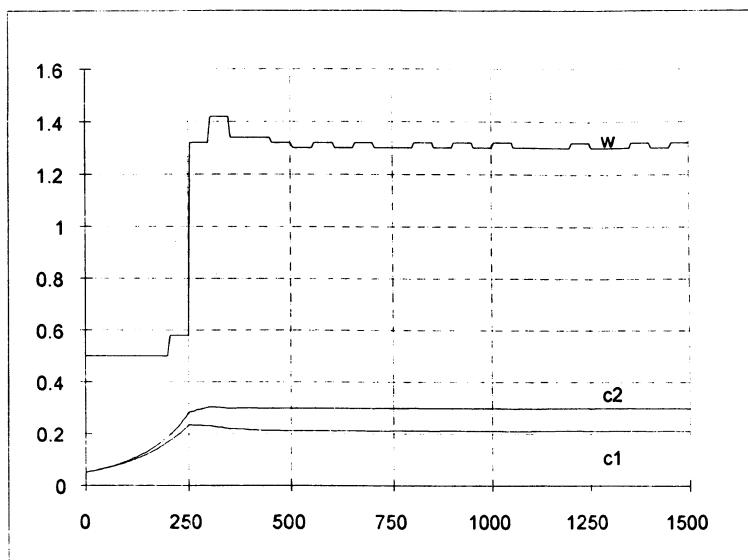
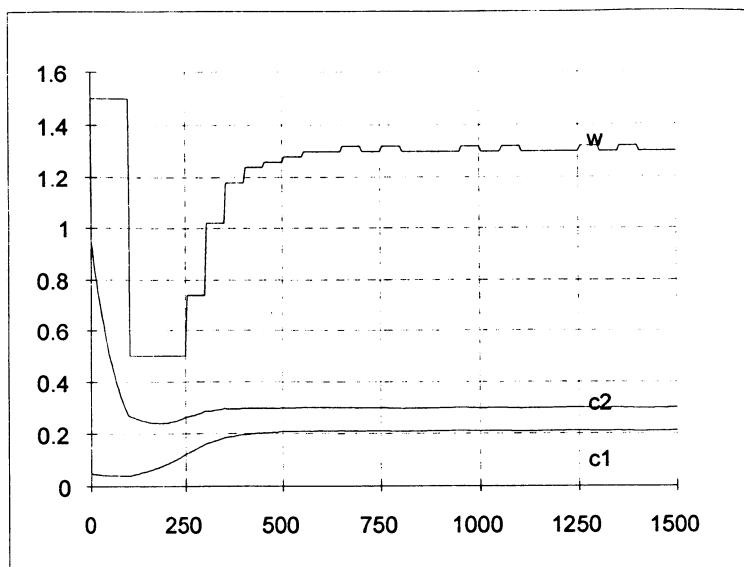
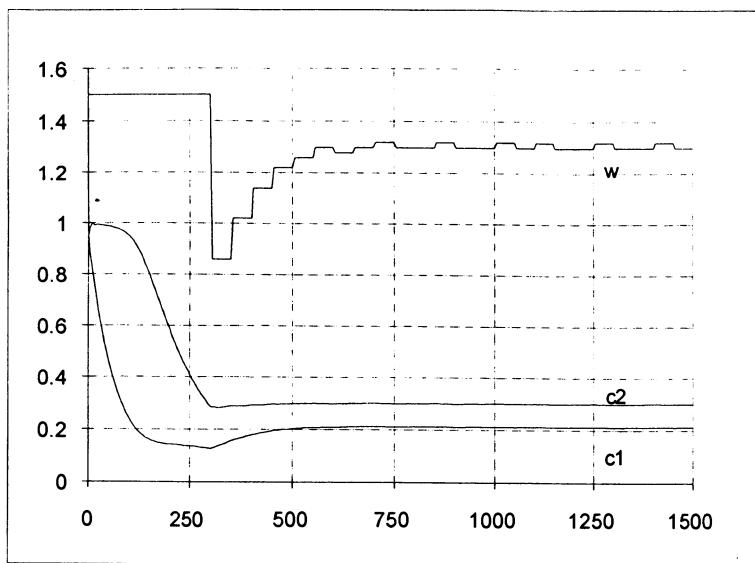


Figure 6 Closed loop response after training. Each figure shows a run of 1500 samples starting from a different initial state. a. Initial state: $c_1=.05$, $c_2=.05$. b. Initial state: $c_1=.05$, $c_2=.95$. c. Initial state: $c_1=.95$, $c_2=.95$.





rates, exploration strategies, and short-term reward functions. It would obviously be desirable to be able to predict in advance to what extent these factors determine success or failure in specific learning control problems. Further empirical and analytic studies are clearly necessary before such an understanding is available.

5 ACKNOWLEDGMENTS

This work was partially supported by Grant IRI-8921275 from the National Science Foundation.

References

- [1] Agrawal, P., Lee, C., Lim, H. C., & Ramkrishna, D. (1982). Theoretical investigations of dynamic behavior of isothermal continuous stirred biological reactors. *Chemical Engineering Science*, 37, 453.
- [2] Barto, A. G. (1990). Connectionist learning for control: an overview. In: W. T. Miller, R. S. Sutton, & P. J. Werbos (Eds.) *Neural Networks for Control*. Cambridge, MA: MIT Press.
- [3] Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13, 835-846.
- [4] Brody, C. (1992). Fast learning with predictive forward models. In: J. E. Moody, S. J. Hanson, & R. P. Lippmann (Eds.) *Advances in Neural Information Processing Systems 4*, San Mateo, CA: Morgan Kaufmann.
- [5] Jordan, M. I. & Jacobs, R. A. (1990). Learning to control an unstable system with forward modeling. In: D. S. Touretzky (Ed.) *Advances in Neural Information Processing Systems 2*, 324-331. Cambridge, MA: MIT Press.
- [6] Lin, Long-Ji. (1991). Self-improving reactive agents: Case studies of reinforcement learning frameworks. *Proceedings of the International Conference on the Simulation of Adaptive Behavior*, MIT Press.

- [7] Moody, J. & Darken, C. J. (1989). Fast learning in networks of locally tuned processing units. *Neural Computation*, 1, 281-294.
- [8] Munro, P. (1987) A dual back-propagation scheme for scalar reward learning *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, 165-176.
- [9] Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1, 4-27.
- [10] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3, 9-44.
- [11] Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning*, 216-224.
- [12] Sutton, R. S., Barto, A. G., & Williams, R. J. (1991). Reinforcement learning is direct adaptive optimal control. *Proceedings of the American Control Conference*, June 26-28, Boston, MA, 2143-2146.
- [13] Ungar, L. H. (1990). A bioreactor benchmark for adaptive network-based process control. In: W. T. Miller, R. S. Sutton, & P. J. Werbos (Eds.) *Neural Networks for Control*. Cambridge, MA: MIT Press.
- [14] Watkins, C. J. C. H. (1989). Learning from delayed rewards. Ph.D. Dissertation, Cambridge University, Cambridge, England.
- [15] Werbos, P. J. (1987). Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17, 7-20.
- [16] Williams, R. J. (1986). Inverting a connectionist network mapping by back-propagation of error. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, 859-865.

AUTOMATED SLEEP EEG ANALYSIS USING AN RBF NETWORK

Stephen Roberts & Lionel Tarassenko

*Department of Engineering Science
University of Oxford, UK*

0.1 INTRODUCTION

There are many examples of expert systems which have been developed in the last twenty years in an attempt to solve medical diagnostic problems automatically (see, for example [1]). There are, however, a number of medical problems which do not lend themselves very well to the expert system's approach. In this chapter, we focus on one such problem, namely the analysis of the electroencephalogram (EEG) during sleep. At present, a set of rules proposed more than twenty years ago [15] is still being used by human experts to classify successive 30-second segments of the EEG sleep record into one of six major categories (wake, dreaming sleep and four stages of progressively deeper sleep) but the rules are notoriously difficult to apply and inter-observer correlation can be as low as 51% for some sections of data [8]. The lack of agreement amongst trained human experts on all but very typical data segments has made the automation of the "sleep scoring" process an almost impossible task.

It has been suggested that neural networks might offer a promising alternative for problems such as these where the extraction and application of rules are tasks which cannot be performed very reliably. Other chapters in this book do indeed show that neural network classifiers, such as the multi-layer perceptron, can be trained from examples to analyse complex signals. Unfortunately, we cannot here use the supervised learning régime required to train a multi-layer perceptron because of the lack of *reliable* labelled data. If we attempt to build a training database from a number of subjects and restrict ourselves only to EEG segments for which all human experts are in agreement, we find that we can only assign a label to a very small proportion of the input patterns. The scarcity of labelled training data means that we cannot construct a multi-layer perceptron with a sufficient number of hidden units to capture the complexity of the problem.

We describe in this chapter the hybrid approach which we have developed for this type of problem. It is based on a radial-basis function (RBF) classifier architecture and combines unsupervised and supervised learning. In the initial unsupervised learning phase, we attempt to learn as much as possible about the distribution of the data in input space using the *whole* database of input patterns, which we use to form a hidden layer representation of input space. The results obtained from this are then used in a subsequent supervised learning phase during which a linear classifier is trained on the hidden unit representation using the small fraction of patterns which is known to be reliably labelled.

The Radial-Basis-Function (RBF) classifier is by now a well-known two-layer neural network architecture [2, 13, 14]. The first layer's non-linear mapping is constructed using a set of *basis functions* whose centres correspond to prototype vectors in input space. These basis functions are normally chosen to be Gaussians and the output activity of the j th hidden unit is then given by:

$$\phi_j = \exp\left(-\frac{|\mathbf{x} - \mathbf{c}_j|^2}{2\sigma_j^2}\right) \quad (0.1)$$

where \mathbf{x} is the input feature vector, \mathbf{c}_j is the prototype vector or centre for the j th unit and σ_j is the width of the Gaussian for that unit. The widths σ_j are chosen (see section 0.4.3) so that the value of ϕ_j only rises significantly above zero when the input vector belongs to the same region in input space as the prototype vector \mathbf{c}_j (localised response). The key aspect of training the first layer of the RBF network is the determination of the location of the centres \mathbf{c}_j and we shall show how our choice of unsupervised clustering procedure allows us to learn as much as possible about the input patterns without having to make any *a priori* assumptions.

In an RBF network, the number of hidden units N_H is usually at least an order of magnitude greater than the dimensionality of the input feature vectors. The hidden-unit space will, therefore, be of relatively high dimension and class separation should be much easier in that space. If a linear network is used to combine the contributions from each hidden unit in order to generate the output classification vector \mathbf{y} we can exploit the properties of linear least mean square (LMS) optimization methods in the final layer of the RBF network [12]. Thus:

$$y_i = \sum_{j=1}^{N_H} w_{ji} \phi_j + w_{0i} \quad (0.2)$$

where y_i is the i th component of the classification vector, w_{ji} is the synaptic weight vector connecting the j th hidden unit to it and w_{0i} is a constant bias term. The most common form of coding scheme for the classification vector \mathbf{y} and the one used in all our work is the 1-out-of- n coding, whereby the target value for y_i is 1 if the input feature vector \mathbf{x} belongs to class i and 0 otherwise. The linear classifier is either trained using the LMS algorithm [19] or pseudo-

inverse matrix inversion techniques [16].

0.2 EEG DATABASE

The database which we used in our studies was built up from 9 whole-night sleep records (total sleep time = 71 hours), acquired from adults with no history of sleep disorders. During each sleep study, the following information was recorded continuously onto tape using the 8-channel Medilog recorder (Oxford Medical Ltd): the standard four channels of EEG (C4-A1), two electro-oculograms (EOGs) and the chin electromyogram (EMG). Three human experts, all trained in the same clinical department, used the standard Rechtschaffen and Kales rules to “score” each sleep record using a sub-set of this data: a single channel of EEG (the frontal EEG), the two EOGs which give an indication of eye movement and the chin EMG since chin muscle inhibition is also a feature of rapid-eye movement (REM) sleep. (As the classification of consecutive 30-second segments into one of six categories is required, it takes between two and five hours for each expert to score an 8-hour record.) 30-second sections which were assigned the same category by all three experts could be identified easily enough but this information was ignored until the final phase of the development of the RBF classifier. For our purposes, we selected the same channel of EEG from which to extract feature vectors (see section 3 below) as inputs to the first layer of the RBF classifier. The initial unsupervised clustering procedure described in section 4 below was carried out on 2 of the 9 sleep records, chosen at random. As this corresponds to a total of 13 hours of sleep EEG, this provided more than enough feature vectors to sample the input space more than adequately and select the free parameters of the first layer.

0.3 DATA REPRESENTATION

The RBF classifier, like other feedforward networks with no recurrent connections, can only be used with *static* input patterns. The EEG is a time-varying signal and must, therefore, be segmented into “frames”¹ during which the signal properties can be deemed to be stationary. Adaptive segmentation of the EEG is an active research area in itself [5], but the EEG is usually considered to be quasi-stationary over intervals of the order of one second, as this is the characteristic time of key transient features such as sleep spindles [9]. This is obviously less than the 30-second intervals used by human experts for sleep scoring and this will create a problem when attempting to use the reliably labelled data as target outputs for the RBF classifier – an issue to which we shall

¹to use the terminology more usually associated with speech processing

return later in this chapter.

The signal was sampled from tape at a rate of 128 Hz with 8-bit accuracy, no analogue pre-filtering being required since the Medilog recorder has a bandwidth of 0.5 to 40 Hz. The digitised signal was low-pass filtered with a linear-phase digital filter with a cut-off frequency of 30 Hz and then parameterized. If the Fast Fourier Transform was used to obtain a frequency-domain representation of the signal, information about slow waves (for example, the 0.5 – 2.5 Hz, or delta, activity) could be lost. The Kalman filter [6, 7], like other auto-regressive modelling techniques, does retain information regarding partial and transient waves present in the segment being analysed; furthermore, the Kalman filter also gives an estimation error which can be used to indicate whether the assumption of quasi-stationarity has been met for that particular one-second segment. For these reasons, we decided to use the coefficients from a 10th order Kalman filter to parameterize the EEG signal [17, 18]. The 10 Kalman filter coefficients computed with every sample were averaged over one second to give a 10-dimensional feature vector for every one-second segment.

0.4 UNSUPERVISED LEARNING

0.4.1 Algorithm

There are a number of unsupervised clustering methods for determining the position of the centres, the most popular being the K -means algorithm [3], where K refers to the number of centres. We chose instead Kohonen’s self-organizing feature map algorithm [10, 11], not because it is the closest “neural network equivalent” but because of its topology preservation properties. In Kohonen’s feature map, the K centres are arranged on a one- or two-dimensional grid such that nearest neighbours in the grid are prototype vectors which are close (again using Euclidean distance as a metric) in input space. This ensures that, as successive input vectors are presented, the movement from one cluster to the next on the feature map is smooth, unless two consecutive input vectors are far apart in input space.² Thus the feature map (especially in its 2-D version) is a powerful *visualisation aid* to track the motion of time-varying signals.

With Kohonen’s algorithm, the number of centres is fixed *a priori*. We chose to use 100 centres distributed on a 10×10 square grid. The 100 centres c_j are given initial random values and the set of input feature vectors $\{\mathbf{x}\}$ is presented

²Discontinuities in the motion from one cluster to the next can also occur if one of the clusters on the trajectory lies on the edge of the map or if the map “folds” back upon itself.

repeatedly and in random order. For each presentation, the Euclidean distance d_j in input space between \mathbf{x} and \mathbf{c}_j is computed. Thus:

$$d_j = \sum_k (x_k - c_{kj})^2 \quad (0.3)$$

where x_k is the k th element of the input feature vector \mathbf{x} and c_{kj} the k th element of centre \mathbf{c}_j . The centre \mathbf{c}_j* which lies closest to the input vector \mathbf{x} (i.e. the one for which d_j is a minimum) is identified. In Moody and Darken's (1989) version of the adaptive K -means algorithm, the prototype vector \mathbf{c}_j* is adapted towards the vector \mathbf{x} ; in Kohonen's self-organizing algorithm, not only is \mathbf{c}_j* adapted but also those centres inside a neighbourhood \mathcal{N} defined around \mathbf{c}_j* on the feature map. The update equation is as follows:

$$\mathbf{c}_j(t+1) = \mathbf{c}_j(t) + \alpha(t) \beta(t) [\mathbf{x}(t) - \mathbf{c}_j(t)] \quad (0.4)$$

where $\alpha(t)$ is a gain parameter which decays exponentially with time ($0 < \alpha(0) \leq 1$). The size of the neighbourhood \mathcal{N} starts from a value which includes most of the centres in the feature map (regardless of the position of \mathbf{c}_j*) so that topological ordering occurs during the initial stages of learning. The size of \mathcal{N} decreases linearly with time until it includes \mathbf{c}_j* only, at which point the algorithm reverts to being the adaptive K -means algorithm. The $\beta(t)$ term ensures that not all the centres which lie within the neighbourhood \mathcal{N} at any one time are adapted by the same amount. $\beta(t)$ is a centre-surround or "Mexican-hat" type of function³ of the distance *in the feature map* between \mathbf{c}_j and \mathbf{c}_j* . With such a function, centres at the periphery of \mathcal{N} receive a small *negative* adjustment, i.e. they are adapted *away* from \mathbf{x} .

There are, therefore, 5 parameters to be determined before the unsupervised clustering procedure can be initialized: the initial gain, $\alpha(0)$, the time constant for the exponential decay of $\alpha(t)$, the initial value of the neighbourhood $\mathcal{N}(0)$ and its rate of change $\frac{d\mathcal{N}}{dt}$, and finally the shape parameter for $\beta(\mathcal{N})$. The choice of these is not critical and Figure 1 shows the functions $\alpha(t)$, $\mathcal{N}(t)$ and $\beta(\mathcal{N})$ which we used in our study. The value of $\frac{d\mathcal{N}}{dt}$ is usually set so that global ordering takes place in the first 10 to 20% of iterations. Beyond this, the centres continue to be adapted one at a time until $\alpha(t)$ has decayed sufficiently for the weight change in each update to be negligible.

³in our implementation, we used a linearised version of this function in order to speed up computation

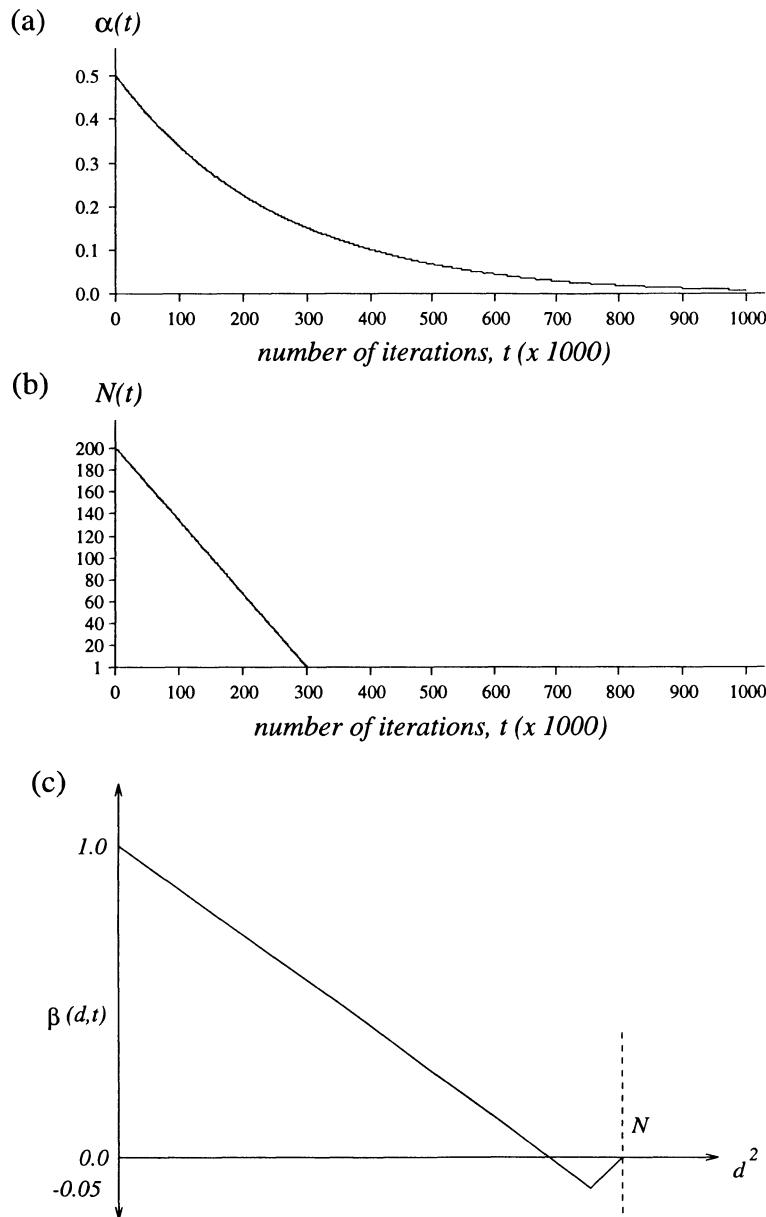


figure 1

0.4.2 Uneven prior probabilities

All of the above analysis assumes that the different types of feature vectors are found in approximately equal numbers in the training set. This is manifestly not true in the case of the sleep EEG, as periods of wakefulness will normally be few and far between in an 8-hour sleep record. Thus the feature vectors generated when the subject is awake (and this includes micro-arousals) are overwhelmingly under-represented in the training set. If the training algorithm is not modified to take this into account, there will not be any centres to cover the region of input space to which those rarely-occurring feature vectors belong. There is, furthermore, no possibility of weighting the training set in their favour since, as we have already said, only a small proportion of the data set is reliably labelled.

We have, therefore, introduced a further modification to Kohonen's algorithm, whereby the neighbourhood \mathcal{N} defined around c_j* not only decreases with time, but also decreases in proportion to the number of times that this centre has previously been identified as c_j* . This has the effect of preventing the most frequently occurring types of feature vectors from occupying the whole of the feature map to the detriment of the rarer vectors. The exact details of the function used to achieve this frequency-dependent neighbourhood shrinkage are not critical (see [18] for details of the linear function which we used).

0.4.3 Cluster formation and display

The unsupervised clustering procedure described above was applied to the training database of 36,000 feature vectors. As a result of 30 iterations through this database, a set of 100 synthesized prototype vectors representative of the whole database was created.

If new (test) data is then applied at the input of the trained feature map, patterns of activity can be seen on the map. These can be displayed as the outputs ϕ_j of the Gaussian kernels centred on each c_j . Before this can be done, however, the width σ_j of each centre must be set. The aim is to achieve a degree of response overlap between a centre and its nearest neighbours so that they can provide a smooth and contiguous interpolation over the region of input space for which they are prototype vectors [13]. There are a number of heuristic methods for achieving this but the simplest method, in which the width of each centre is made equal to the Euclidean distance to the nearest centre, works as well as any of the other nearest-neighbour heuristics [4].

When the outputs ϕ_j are displayed as patterns of activity on the trained feature map, a localised response is observed because of the topology preservation properties of the map: for any input vector \mathbf{x} , the centres with the highest “activities” will be c_j* and its nearest neighbours. The actual number of centres highlighted on the map then depends on the threshold chosen for the value of ϕ_j below which centres are considered to be inactive for display purposes. One alternative is to adopt a hard threshold instead, whereby only c_j* is displayed (“winner-take-all” display). We have used both forms of visualisation but the results reported in this chapter were obtained with the winner-take-all thresholding. With this strategy, the existence of clusters in the feature map can be determined using the following simple procedure: plots are constructed showing, in the z -dimension, the number of times each centre is activated as the winner with the indices of the grid serving as discrete x, y coordinates. An example of such a plot is shown in Figure 2 in which the presence of 3, or possibly 4, clusters is clearly shown for a section of sleep lasting for several minutes. When the whole test database is submitted as input to the trained map, a total of eight clusters is clearly visible (see Figure 3, in which they are labelled from A to H) and this for each test subject, as determined automatically by a 2-D peak detection algorithm.

0.4.4 Motion between clusters

Observation of the patterns of activity in the feature map showed that the 8 clusters of Figure 3 correspond to “halting-states” in an otherwise continuous movement of the EEG’s signal state; the time spent at the 8 cluster sites is greater than the time taken in motion between them. Further examination of the trajectories between clusters, however, also revealed that there were only 3 primary types of transition trajectories, as shown in Figure 4 which shows typical trajectories (each line segment in the figure represents the trajectory over 30 seconds of EEG data). Type 1 has a trajectory from $A \rightarrow A$; with Type 2, there are regular trajectories between states B, E and G and Type 3 shows mainly a trajectory $D \rightarrow D$ with excursions $D \rightarrow F \rightarrow D$ and a short excursion into Type 1. The fourth plot shows a more complex set of trajectories, but closer observation reveals it to be some linear combination of the 3 other types of trajectories.

These observations directly from the map led to more rigorous mathematical analysis. The elements of the state transition matrices were computed assuming a Markov model (i.e. the probability of transition from one cluster to the next is assumed to be independent of prior system states). Figure 5 shows the transition probabilities calculated from the training data set and these are a

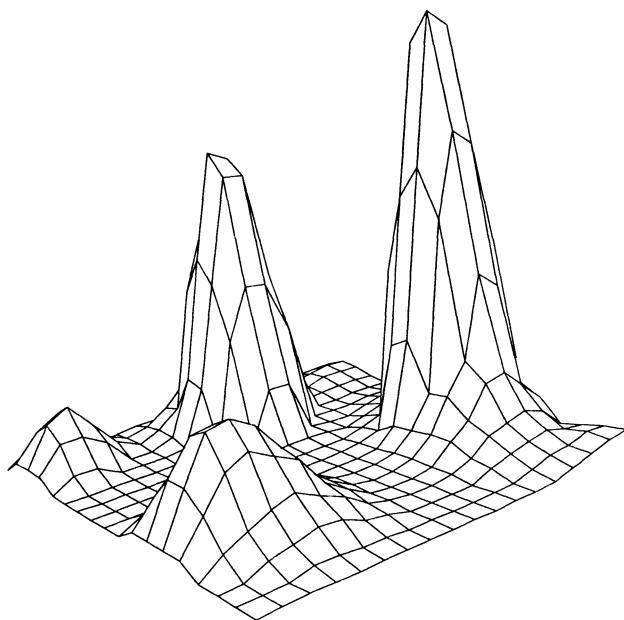


figure 2

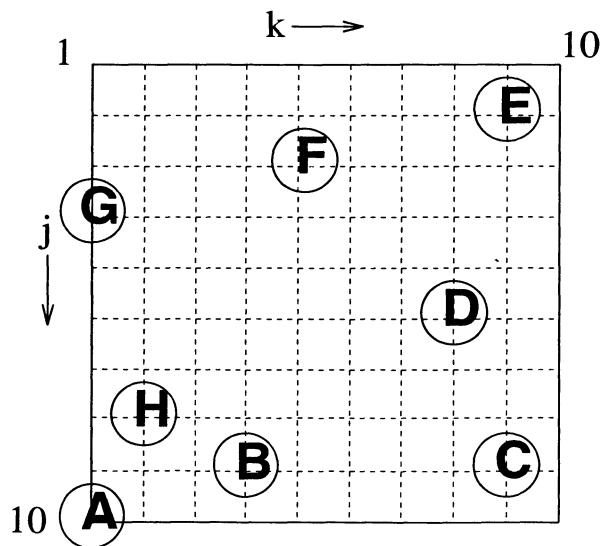


figure 3

clear confirmation of the qualitative data of Figure 4.

The three types of trajectories are related to the physiological “states” of wakefulness (process W), dreaming (REM) sleep (process R) and deep sleep (process S). It must be stressed, however, that the existence of the halting-sates and of the smooth trajectories between them was discovered, with no *a priori* assumptions, as a result of unsupervised clustering and primarily because of the excellent visualisation properties of Kohonen feature maps.

0.5 SUPERVISED LEARNING

In order to quantify the relative importance of processes W, R and S at any one time, the hidden layer representation, consisting of the activities ϕ_j of each centre, was used as the input to a linear classifier with 3 output units. Training sets each consisting of 8 minutes of wakefulness, dreaming and deep sleep were assembled to represent the 3 processes W, R and S, respectively. For each of these, only sections for which there was 100% agreement in the experts’ scoring were used; the classification was stage wake for set W, stage REM for set R and stage 4 sleep for set S. Hence we are making use of the input-output pairs which are known to be more *reliably* labelled but which are relatively few and far between. The data can be assembled with relative ease as the EEG is clearly distinguishable between wakefulness and deep sleep and dreaming sleep is associated with clear non-EEG events, i.e. Rapid Eye Movements (REMs) and reduction of muscle tension. It is usually stages 1, 2 and 3 which are the most difficult to score as they are intermediate stages. These will now be indicated by the RBF classifier by output values y_i which lie anywhere *between* 0 and 1, but whose sum $\sum_i y_i$ will always be equal to 1.0, since, during training, the target value for y_i is 1 if the input belongs to class i and 0 otherwise.

The linear classifier is trained using the LMS algorithm to minimize the mean square output error. If \mathbf{y} represents the vector of three probability values output by the classifier, and \mathbf{d} the desired probability vector⁴ (from the labelled training set) then:

$$w_{ji}(t+1) = w_{ji} + \gamma(y_i - d_i)\phi_j$$

where γ is a learning parameter for which $0 < \gamma < 1$.

⁴ \mathbf{d} here can only be {0,0,1}, {0,1,0} or {1,0,0}

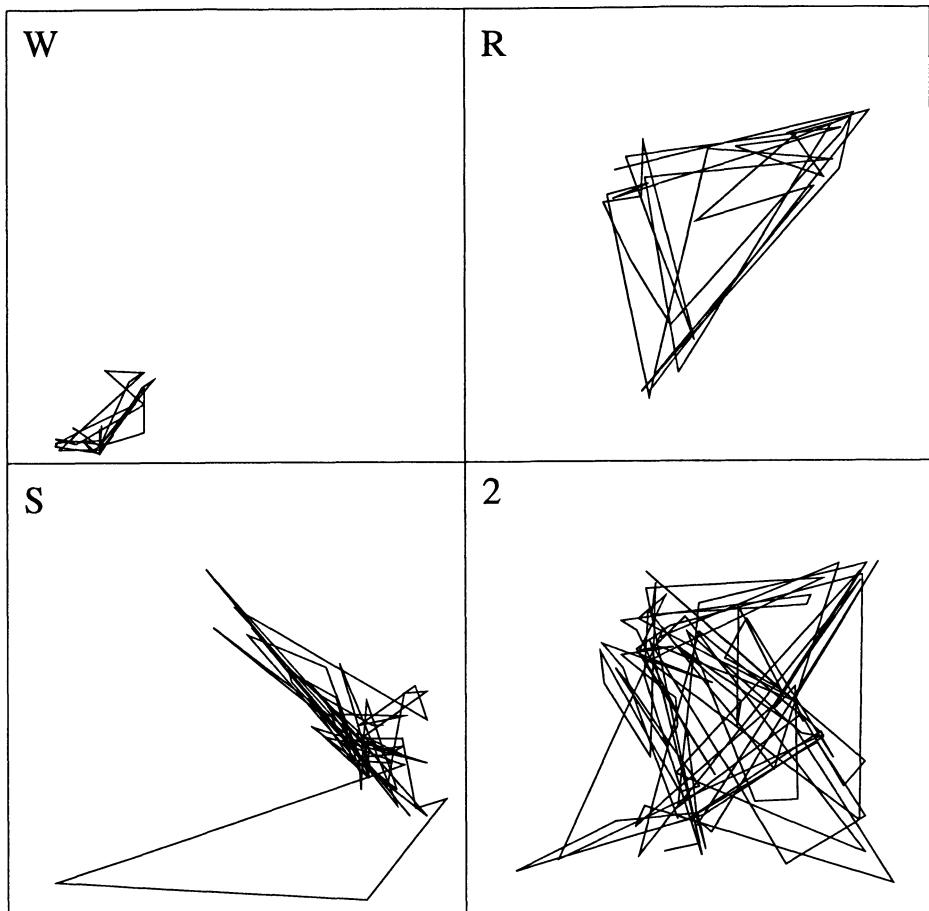


figure 4

0.5.1 Results from original sleep studies

The time course of the 3 probability values provides a quantitative analysis of sleep patterns, as is shown in Figure 5 for one of the 7 subjects in our test set. The following observations can be made:

- there is a clear oscillatory pattern in the values of y_S (deep sleep) and y_R (REM sleep) as sleep waxes and wanes during the night.
- as the night progresses, the mean value of y_S decreases whilst the mean value of y_R increases. It is known that the amount of REM sleep increases and the amount of deep sleep decreases as the night progresses.
- Arousals to wakefulness (typically caused by body movements during sleep) are shown in y_W as spikes from the baseline value.

Figure 5 shows that the output units of the RBF network provide a much more accurate description of the sleep process than the arbitrary six stage classification derived from the Rechtschaffen and Kales rules (see the 4th. plot of figure 6 for a typical “hypnogram”). In particular, the depth of sleep can be assessed with precision; for example, the well-known sleep cycle (with a period of about $1\frac{1}{2}$ hours in Figure 5) is clearly evident in the time course of either y_S or y_R .

0.5.2 Analysis of severely disturbed sleep

The network trained on the 2 normal subjects was also used in an attempt to quantify severely disturbed sleep which is unscorable with the conventional rule-based system. The example given here is the EEG during the sleep of a subject with periodic arousals due to respiratory malfunction. Figure 6 shows respiratory effort plotted along with the time course of y_W for 8 minutes of disturbed sleep. The trace of respiratory effort shows the subject to be alternating between periods of *apnoea* (cessation of breathing) and *hyperventilation* (abnormally rapid breathing). Note how the y_W trace gives a clear indication of the cyclical nature of the arousals caused by the abnormal respiration; note also its phase correspondence with breathing – the subject is arousing during hyperventilation, and falling asleep when breathing ceases. This type of pattern has been known about by clinicians for some time but has never been analysed quantitatively before. (The R & K rules break down because changes are occurring on a time scale which is much less than 30 seconds.) The EEG

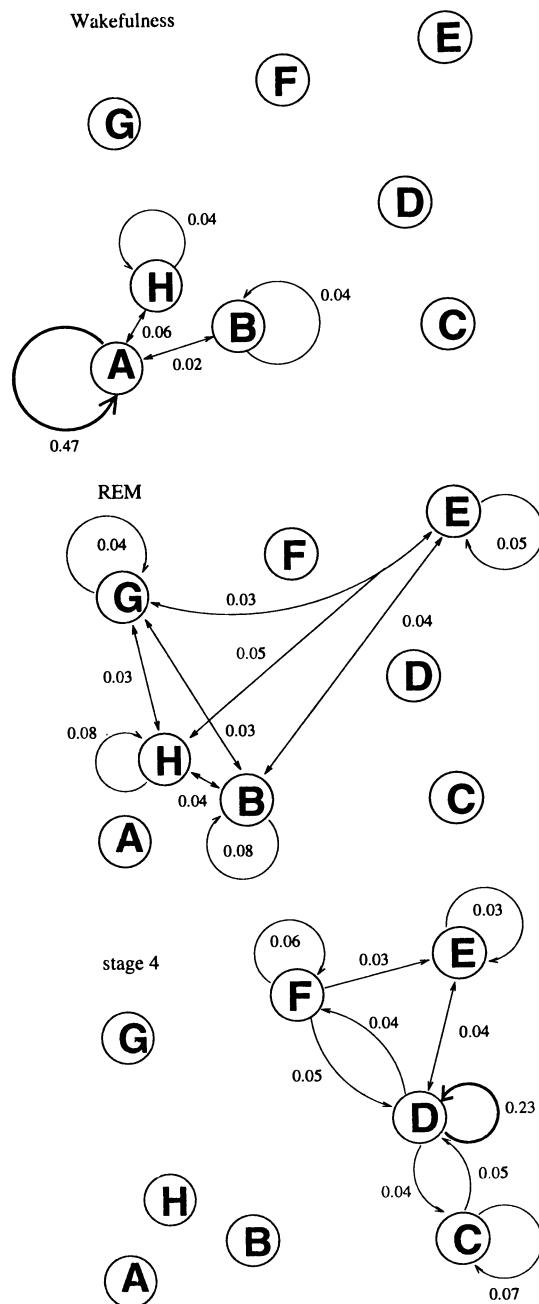


figure 5

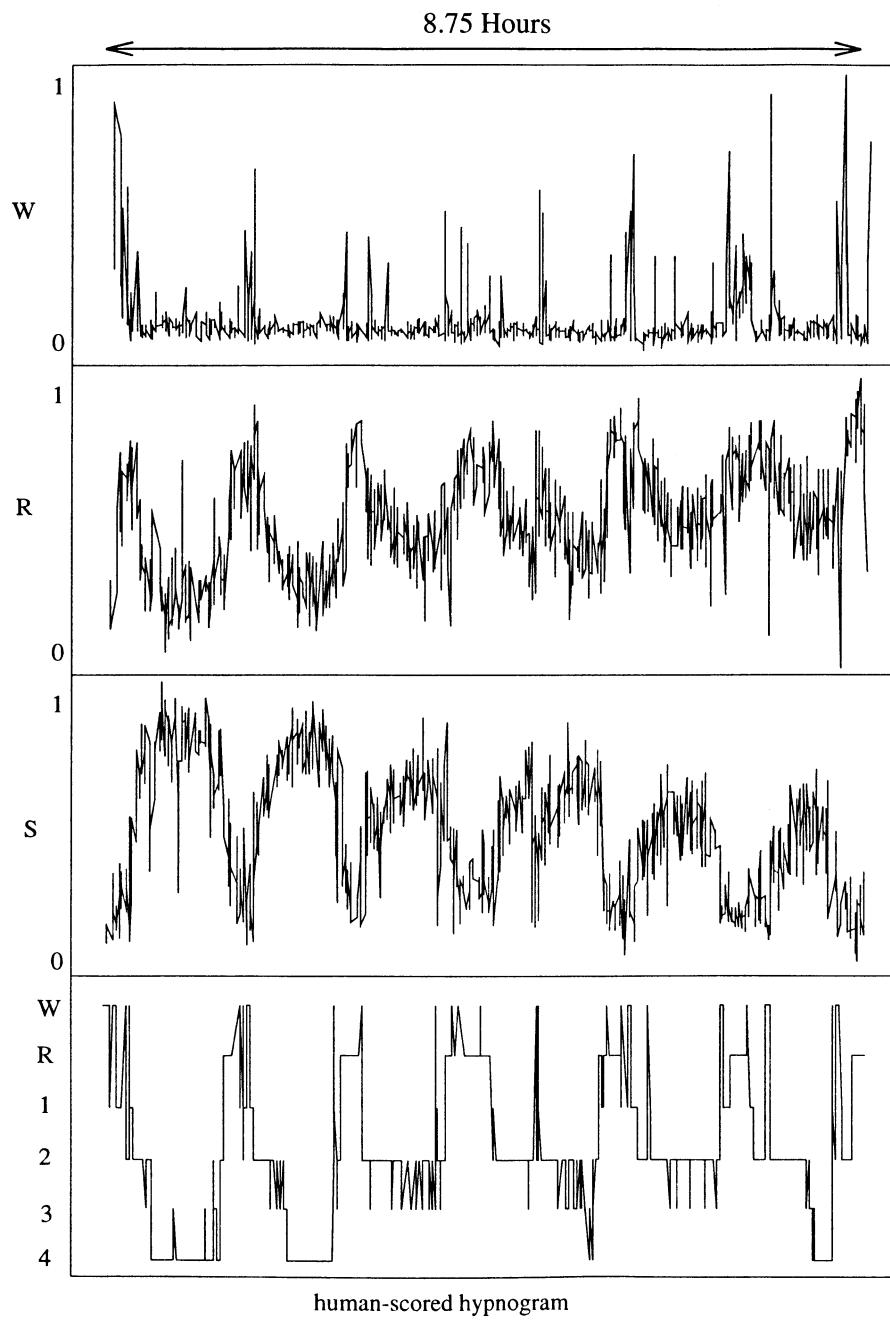


figure 6

patterns of subjects with disturbed sleep of the type shown in Figure 6 are the same as those found in the training database of normal subjects but occur in different sequences.

0.6 CONCLUSION

We have described a neural network approach to a medical diagnostic problem for which the existing rule-based system does not give entirely satisfactory results. The precise formal rules of such a system do not lend themselves to the analysis of a signal as variable and complex as the sleep EEG. Starting with no *a priori* assumptions about the number of states which may exist within the EEG during sleep, we learnt as much as we could about the nature of the input data during an unsupervised learning phase. This process was greatly aided by utilizing the excellent *visualisation* properties of Kohonen's feature map.

Our analysis of the patterns of activity observed on the feature map was then made *quantitative* by assigning the map's centres to be the hidden units of an RBF classifier. The activities of these units, computed using Gaussian kernels, are used as the input values to a linear classifier. The linear classifier can be trained on a sub-set of the input-output pairs, using only those which are known to be reliably labelled. By using a 1-out-of- n coding and LMS optimization, the output values of the linear classifier are made to represent the posterior probabilities of belonging to the 3 classes identified from observation of the feature map. The three output values, as they vary between 0 and 1, give a description of the sleep process which is much more precise than the arbitrary six-state classification of the rule-based system. We have also shown that severely disturbed sleep, which is unscorable with the rule-based system, can be analysed with our RBF network.

Clinical trials with our system are about to start in half a dozen European sleep laboratories. Although the approach described in this chapter has been within the context of a medical classification problem, it is generic and can be applied to other diagnostic problems.

BIBLIOGRAPHY

- [1] J.L. Alty and M.J. Coombs. *Expert Systems, Concepts and Examples*. NCC publications, Manchester, UK, 1984.
- [2] D. Broomhead and D. Lowe. Multivariable function interpolation and adaptive networks. *Complex Systems*, (2):321–355, 1988.
- [3] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
- [4] E. Hartman and J.D. Keeler. Predicting the Future : advantages of semi-local units. *Neural Computation*, (3):566–578, 1991.
- [5] B.H. Jansen, A. Hasman, and R. Lenten. Piecewise analysis of EEGs using AR-modeling and clustering. *Comput. Biomed. Res.*, 14:168–178, 1981.
- [6] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans of Am. Soc. Mech. Eng. (Series D)*, 82:35–45, 1960.
- [7] R.E. Kalman and R.S. Bucy. New results in linear filtering and prediction theory. *Trans of Am. Soc. Mech. Eng. (Series D)*, 83:95–108, 1961.
- [8] Kelley, J.T. and Reed, K. and Reilly, E.L. and Overall, J.E. Reliability of Rapid Clinical Staging of All Night Sleep EEG. *Clin. Electroenceph.*, 16(1):16–20, 1985.
- [9] B. Kemp, E.W. Groneveld, A.J.M. Janssen, and J.M. Franzen. A model based monitor of human sleep stages. *Biol. Cybern.*, 57:365–378, 1987.
- [10] T. Kohonen. Self-Organized formation of topographically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.
- [11] T. Kohonen. The self-organizing map. *IEEE Proceedings*, 78(9):1464–1480, 1990.
- [12] D. Lowe and A.R. Webb. Exploiting prior knowledge in network optimization: an illustration from medical prognosis. *Network: Computation in Neural Systems*, 1(3):299–323, 1990.

- [13] J. Moody and C. Darken. Fast Learning in Networks of Locally-tuned Processing Units. *Neural Computation*, (1):281–294, 1989.
- [14] J. Platt. A Resource-Allocating Network for Function Interpolation. *Neural Computation*, (3):213–225, 1991.
- [15] A. Rechtschaffen and A. Kales. A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects. Technical report, UCLA, Los Angeles, USA, 1968.
- [16] S. Renals and R. Rohwer. Learning Phoneme recognition using Neural Networks. *Proc. Int. Conf. Acous. Speech and Signal Processing (Glasgow, Scotland)*, pages 413–416, 1989.
- [17] S. Roberts and L. Tarassenko. A New Method of Automated Sleep Quantification. *Med. & Biol. Eng. & Comput.*, 30(5):509–517, 1992.
- [18] S. Roberts and L. Tarassenko. The Analysis of the Sleep EEG using a Multi-layer Network with Spatial Organisation. *IEE Proceedings-F*, 139(6):420–425, 1992.
- [19] B. Widrow and S.D. Stearns. *Adaptive Signal Processing*. Prentice-Hall, New Jersey, 1985.