An Analysis of Learning in Weightless Neural Systems

N.P. Bradshaw

Neural Systems Engineering Group
Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine

A thesis submitted for the degree of Doctor of Philosophy
and for the Diploma of Imperial College

September 1996

Abstract

This thesis brings together two strands of neural networks research - weightless systems and statistical learning theory - in an attempt to understand better the learning and generalisation abilities of a class of pattern classifying machines.

The machines under consideration are n-tuple classifiers. While their analysis falls outside the domain of more widespread neural networks methods the method has found considerable application since its first publication in 1959. The larger class of learning systems to which the n-tuple classifier belongs is known as the set of weightless or RAM-based systems, because of the fact that they store all their modifiable information in the nodes rather than as weights on the connections.

The analytical tools used are those of statistical learning theory. Learning methods and machines are considered in terms of a formal learning problem which allows the precise definition of terms such as learning and generalisation (in this context). Results relating the empirical error of the machine on the training set, the number of training examples and the complexity of the machine (as measured by the Vapnik-Chervonenkis dimension) to the generalisation error are derived.

In the thesis this theoretical framework is applied for the first time to weightless systems in general and to n-tuple classifiers in particular. Novel theoretical results are used to inspire the design of related learning machines and empirical tests are used to assess the power of these new machines. Also data-independent theoretical results are compared with data-dependent results to explain the apparent anomalies in the n-tuple classifier's behaviour.

The thesis takes an original approach to the study of weightless networks, and one which gives new insights into their strengths as learning machines. It also allows a new family of learning machines to be introduced and a method for improving generalisation to be applied.

Acknowledgements

I would like to thank my supervisor, Professor Igor Aleksander for offering me his experience, his many interesting insights and, of course, the chance to carry out this research. Also thanks to Nick Sales and Richard Evans for their technical help with the weightless systems, and thanks to the rest of the Neural Systems Group for their patience, good-humour and for not minding not being listed by name.

Outside the group I must thank my parents (in the most inclusive sense of the word) on whose support I have always been able to count. Similarly my friends in London, Oxford, the North of England, Belgium, the US and the few who have run away to Cambridge. Having people to help one through the more dispiriting times is half the battle of completing a PhD. They have also made the good times lots of fun.

Thanks also to the inventors of the research tools *Mathematica*, *gnuplot*, *latex* and *Foop*, without whom this thesis could never have been.

Contents

1 Introduction **9**

1.1 The Inspiration for the Thesis 9

1.2 Aims of the Thesis 11

1.2.1 The Current State of Affairs 11

1.2.2 What is lacking 12

1.2.3 What should be done 12

1.2.4 What will be done in the thesis 13

1.3 Contents of the Thesis 13

2 An Introduction to Learning Machines **16**

2.1 What is a Learning Machine? 16

2.2 History of Learning Machines 17

2.3 Neural Networks 19

2.3.1 Perceptrons and Incremental Methods 20

2.4 Weightless Systems 22

2.4.1 The N-Tuple Method 22

2.4.2 Thresholded Discriminators 26

2.4.3 WISARD 26

2.4.4 PLNs 26

2.4.5 GRAMs and GNUs 27

2.4.6 Other Systems 28

3 An Introduction to Learning Theory **29**

3.1 Introduction 29

3.2 Risk Minimisation 30

3.2.1 The Learning Problem 31

3.2.2 Expected and Empirical Risk 31

3.2.3 Common Risk Functions 32

3.2.4 Empirical Risk Minimisation 33

3.3 Bounding the Expected Risk 36

3.3.1 Non-Constructive Bounds on Generalisation 36

3.3.2 The VC Dimension 41

3.3.3 Constructive Distribution Independent Bounds 44

3.4 Structural Risk Minimisation 44

3.4.1 Controlling the VC dimension 45

3.4.2 Asymptotic Analysis of the SRM Principle 46

4 The VC Dimensions of Maximum Threshold Discriminators 48

4.1 Principles for Calculating the VC Dimensions 48

4.2 Types of Discriminator 49

4.3 Definitions and Notation 50

4.3.1 Conditions for Shatterability 51

4.4 The Simpler Cases 52

4.4.1 Two Disjoint Nodes 52

4.4.2 N Disjoint Nodes 55

4.4.3 Two Nodes Overlapping by l 57

4.4.4 More General Configurations 58

4.4.5 Constructing the Bounds on VC Dimension 58

4.5 Auto-Associative GNUs 62

4.5.1 Conditions for Shatterability 63

4.5.2 Calculating the VC Dimension 64

5 VC Results for Other Discriminators and n-Tuple Classifiers 65

5.1 Which Other Discriminators? 65

5.2 Sub-Maximal Threshold Discriminators 66

5.2.1 A Constructive Lower Bound 66

5.2.2 An Upper Bound 67

5.2.3 Impossible Node Value Sets 72

5.3 Thick Thresholds 72

5.3.1 A Lower Bound on Thick Threshold VC-dimension 73

5.3.2 An Upper Bound on Thick Threshold VC-dimension 74

5.4 The VC Dimension of the n-Tuple Classifier 74

5.4.1 Defining the Learning Machine 74

5.4.2 The VC Dimension Bounds 75

5.5 A Reformulation of the Two Discriminator Classifier 78

5.5.1 Advantages of the Reformulation 78

5.5.2 Counting the Hypotheses 79

5.6 Larger Output Sets 80

6 Experimental Results and Comparisons **82**

6.1 Theoretical Performance Predictions 82

6.1.1 Weightless Systems 82

6.2 Comparison with Experiment 83

6.2.1 Benchmarking with StatLog 84

6.2.2 Conclusions from the StatLog trials 85

6.3 Reasons for the Loose Bounds 86

6.4 Effective VC Dimension 88

6.4.1 Definitions and Theory 88

6.4.2 Maximising the Error Divergence 90

6.4.3 The Experimental Method 92

6.4.4 Analysis of Results 100

6.4.5 What to Make of the Results? 102

6.4.6 Investigating the SMA 109

6.5 Other Systems 109

7 The N-tuple Classifier as a Statistical Learning Machine **112**

7.1 Optimising the N-Tuple Classifier: A Review 112

7.1.1 Structure: The Size of the N-tuple 113

7.1.2 Structure: The Range of the Stored Values 114
7.1.3 Structure: The Connection Graph 115
7.1.4 The Training Algorithm 116
7.1.5 Preprocessing 118
7.1.6 A Summary of N-Tuple Classifier Techniques 118
7.2 The Problem for Learning Machines 118
7.2.1 The Bias/Variance Dilemma 119
7.2.2 Minimum Description Length 121
7.2.3 Regularisation and Bayesian Methods 122
7.2.4 Data-Dependent Methods 124
7.3 The N-tuple Classifier Explained 125
7.3.1 Empirical Risk Minimisers 126
7.3.2 Structural Risk Minimisers 128
7.3.3 Designing Bias 132
7.4 What Results from these Results? 132
7.4.1 Future Directions 135

8 Conclusions and the Future for the N-tuple Classifier 136
8.1 Fulfilling the Aims of the Thesis 136
8.2 A Recap of the Thesis 137
8.2.1 Chapter 1 137
8.2.2 Chapter 2 137
8.2.3 Chapter 3 137
8.2.4 Chapter 4 138
8.2.5 Chapter 5 139
8.2.6 Chapter 6 139
8.2.7 Chapter 7 140
8.2.8 Chapter 8 140
8.3 Were the Aims Achieved? 140
8.4 What Wasn't Done 141
8.5 Directions for Future Work 142

8.6 A Take Home Message 143

A Identical Hypotheses from Different Node Value Sets 150

B Features and Relatedness in GNUs 152

B.1 Introduction 153

B.2 The Relatedness Function 154

B.2.1 The Nearness Functions 155

B.2.2 Finding the *Real* Features 157

B.2.3 Uses of the Relatedness Function 157

B.3 Brief Analysis of GNU behaviour 158

B.4 Conclusions 161

B.5 References 161

C The Future for Weightless Systems 162

C.1 Introduction 163

C.2 Formalisation 163

C.3 The Generalisation Game 164

C.4 What they *can't* do 166

C.5 Feature Sensitivity 167

C.6 Enhancement Speculation 167

C.6.1 Ways and Means 168

C.6.2 Optimal Mappings and Connections 170

C.6.3 Statistical Learning Theory 171

C.7 Conclusions 171

C.8 References 172

D Improving the Generalisation of the N-Tuple Classifier using the Effective VC Dimension 174

List of Tables

6.1 Generalisation performance on StatLog against predictions from VC dimension bounds. 85

6.2 Effective (uniform) and actual VC dimensions for n-tuple classifier. Ratio 1 is the VC dim lower bound over EVC dim while ration two is VC dim upper bound to EVC dim. 107

6.3 Effective (Gaussian) and actual VC dimensions for n-tuple classifier. Ratio 1 is the VC dim lower bound over EVC dim while ration two is VC dim upper bound to EVC dim. 107

D.1 VC dimension lower bound, VC dimension upper bound and effective (Gaussian) VC dimension of n-tuple classifier. 178

List of Figures

2.1 The n-tuple Discriminator (schematically). 24

2.2 The weightless general neural unit. 28

3.1 All four dichotomies (given by two lines) of two points. 38

3.2 The fourteen available dichotomies of the four corners of the unit square. 38

3.3 (0,0) and (1,1) cannot take the opposite value to (0,1) and (1,0). . . . 39

3.4 (0,0) and (1,1) cannot take the opposite value to (1/2,1/2). 39

3.5 Possible (and impossible) forms of the growth function. 43

3.6 A structure on the perceptron based on the number of hidden units. . 46

3.7 The form of the total error for a structure of ERM-based learning machines with increasing VC dimension. 47

6.1 Empirically determined $E[\xi_l]$ for output distributions of varying difficulty. Uniform distribution. 94

6.2 Empirical $E[\xi_l]$ for some single discriminator classifiers. Uniform distribution. 96

6.3 Empirical $E[\xi_l]$ for a range of n-tuple classifiers with varying number of nodes per discriminator, N. Uniform distribution. 96

6.4 Empirical $E[\xi_l]$ for a range of n-tuple classifiers with varying n-tuple size, N. Uniform distribution. 97

6.5 Empirical $E[\xi_l]$ for a range of n-tuple classifiers with varying size of output set, i. Uniform distribution. 97

6.6 Empirical $E[\xi_l]$ for single discriminators against estimated EVCD over l plotted against a best-fit theoretical estimate. Uniform distribution. 98

6.7 Empirical $E[\xi_l]$ for varying N against estimated EVCD over l plotted against a best-fit estimate. Uniform distribution. 98
6.8 Empirical $E[\xi_l]$ for varying n against estimated EVCD over l plotted against a best-fit estimate. Uniform distribution. 99
6.9 Empirical $E[\xi_l]$ for varying i against estimated EVCD over l plotted against a best-fit estimate. Uniform distribution. 99
6.10 Estimates of $E[\xi]$ for Gaussian input and different difficulties of output. 102
6.11 Estimates of $E[\xi]$ for Gaussian input n-tuple classifiers with varying N. 103
6.12 Estimates of $E[\xi]$ for Gaussian input and n-tuple classifiers with varying n. 103
6.13 Estimates of $E[\xi]$ for Gaussian input and n-tuple classifiers with varying i. 104
6.14 The results for Gaussian inputs normalised by effective VC dim and with a best fit theoretical error term – N varying. 104
6.15 The results for Gaussian inputs normalised by effective VC dim and with a best fit theoretical error term – n varying. 105
6.16 The results for Gaussian inputs normalised by effective VC dim and with a best fit theoretical error term – i varying. 105

7.1 Error on the training set when trained with three different algorithms. Uniform distribution. 127
7.2 Error on the training set when trained with three different algorithms. Gaussian distribution. 127

A.1 The logarithm of the proportion of constant hypotheses for a $n = 4, N = 50$ discriminator thresholded at each value of Θ from 1 to 50. . 151

D.1 Max half-sample error against l for three different output distributions. 179
D.2 Empirical $E[\xi_l]$ against estimated effective VC dim over l plotted against a best-fit estimate. 179

Chapter 1

Introduction

1.1 The Inspiration for the Thesis

The forty year history of artificial neural networks from the perceptron in 1958 to the present day (1996) has been one of alternating excitement and retrenchment. Early successes of the perceptron learning rule with its apparently miraculous power to learn were followed by disappointment at the apparently limited range of both its learning and its modelling capability. Latterly the Hopfield and non-linear perceptron revival have extended the range and power of (what are now often known as) conventional neural networks to the point that neural networks now offer real alternatives to classical approaches in fields from pattern recognition to control to psychological modelling. Such networks and such applications continue to be the subject of intense and widespread study.

However, while perceptrons and similar architectures enjoy such ups and downs, there exist many related systems whose properties are sufficiently similar for them to be termed "neural networks" but whose histories have been less explosive. The n-tuple architectures of Bledsoe and Browning are one case in point and it is the analysis of these and their descendent systems which forms the core of this thesis. These systems sample binary input spaces in blocks of n (hence "n-tuple"), they can be easily implemented in logical or physical memory (hence the alternative term "RAM-based") and they are distinguished from perceptron-type networks by the fact that there are no weights attached to each connection (hence a second alternative "weightless'). Modifications with probabilistic output, internal feedback, weighted connections and complex architectures have led to the development of a large family of weightless networks whose capabilities have been investigated in less depth and breadth over the same forty years of the perceptron, but which have had many practical successes.

So how may such systems be compared with the perceptron family and others? The question can, and should be asked more generally. How can any systems be compared on any particular problem or set of problems? An answer to this question will require a precise formulation of the kinds of problems artificial neural systems are required to solve as well as the quantity and the type of training data available. These are questions of statistical inference, and quite naturally have been studied in depth by statisticians. The resulting theory of learning (as far as the different approaches to the problem constitute a single theory) comprises the second element of this thesis.

Learning theory began as a branch of applied statistics inspired by the success of Rosenblatt's perceptron algorithm in the early 1960s [46]. The perceptron was itself inspired by biological models of information processing dating back to McCulloch and Pitts [33], but what was of interest to some statisticians was the way in which the perceptron adapted its internal parameters to model the data set when given nothing but a set of training examples. Questions of convergence to the correct solution, performance on small training samples and control of the architecture of the machine for optimal performance were and remain at the heart of the learning theory enquiry. In this thesis the results of learning theory are applied to weightless networks to understand better the performance and behaviour of such systems and to determine where they are most effective and how they may be optimised.

1.2 Aims of the Thesis

1.2.1 The Current State of Affairs

This thesis deals with RAM-based or “weightless” classifiers. These are machines which are used (in real-world applications as well as in the lab) to classify patterns into pre-defined classes. Moreover, the behaviour of these classifiers, while obviously constrained by the architecture, is essentially guided by a so-called “training set” of example patterns and their correct outputs. For this reason RAM-based classifiers are examples of what are known as learning machines. They are known to have very quick training and operation times and in most cases exhibit acceptable accuracy. However, as learning machines they face considerable competition.

Many machines, computer programs and statistical techniques exist to perform this sort of pattern classification and the RAM-based techniques comprise only one set of methods out of a very large number. Even artificial neural networks, the category of classifiers in which the RAM-based classifiers are often placed, comprise a wide spectrum of different architectures and learning methods. Many of these methods have been shown to work more or less well on various test and benchmark sets, but to make the best use of this variety a principled general method for selecting one system over another is required. The very fact that there are so many different types of pattern classifier leads naturally questions of comparative performance. Can we say, a priori, which classifier will perform best on a problem, how the parameters of the classifier should be set, how the classifier should be trained or programmed, what level of generalisation error we can expect to obtain?

The questions posed above have been considered by many researchers studying learning machines and several general theories have been proposed for studying the efficacy of learning machines. Bayesian statistics is one such theory which has been applied to artificial neural network analysis and attempts have been made to put RAM-nets into a Bayesian framework. Statistical learning theory is another means of analysing learning machines in a general framework but to date no attempt has been made to study RAM-nets with these theoretical tools.

1.2.2 What is lacking

It is fair to say that at present a general theory of learning machine capabilities does not exist. Nor need it ever. However the results of statistical learning theory allow certain aspects of performance, notably the expected generalisation error, to be bounded in terms of simple scalar properties of the learning machine and error on the training set. These results, which assume that the learning machine must be selected with almost no prior knowledge of the learning problem, beyond the data in the training set, can be extended to try and make use of domain- or data-specific knowledge.

More importantly from the point of view of RAM-based systems there has been very little application of statistical learning theory to such systems. Furthermore no other means of analysis has been applied to RAM-based systems to obtain general bounds on generalisation error. Hence all comparisons between RAM-based systems and others have been on the basis of particular systems and benchmark tests. A learning theory analysis generates results which hold over a wide range (often all) data sets.

1.2.3 What should be done

It is assumed that a thorough and objective analysis of RAM-based systems would be useful for several reasons. Firstly if results can be obtained which allow comparison with other learning machines, then someone wishing to select a learning machine for a particular task can formulate criteria and test machines against them. In particular it is interesting to know if the high speed of training and testing which are associated with the RAM-based systems carries an overhead in terms of increased error rates. Secondly, because RAM-based systems are popular systems with real applications, any analysis of them will yield information about their behaviour in which current users, and those simply with an interest in the system, may be interested.

To make such an analysis several steps are necessary. The learning machine and the problem to which it is applied must be defined formally so that the the learning theory is applicable. Standard learning theory results may then be applied to give results about the generalisation error of the system. These results are then comparable between different learning systems and provide a useful point of comparison.

1.2.4 What will be done in the thesis

This thesis addresses precisely those points which were noted above as things that should be done. The popular RAM-based systems will be formally defined and a measure of their complexity, the "Vapnik-Chervonenkis dimension" is calculated, or is some cases simply bounded. This quantity is then used to obtain the generalisation bounds for the classifiers. Ways of improving these bounds, especially when the data set is small, are then considered. The precise way this is achieved is set out in the contents description in the following section.

1.3 Contents of the Thesis

Chapter 2 contains the background in learning systems in general and weightless systems in particular. The origins and progress of the subject area are discussed, and the systems themselves are introduced. This chapter does not aim to be fully comprehensive, but rather to place the n-tuple classifier in the context of its "competition".

Chapter 3 contains the background in learning theory necessary for the thesis. The pattern recognition learning problem is set out formally and conditions for its solutions in terms of the size of the training set and the complexity of the learning machine are given. Most proofs are not shown as the mechanics underlying most of the results are not needed and none of the original results presented in this thesis extend or generalise them. Indeed they are all specific cases of the general theory which use original results for n-tuple classifiers to describe classifier performance on the general learning problem.

Chapter 4 contains much of the original mathematics of the thesis. In order to evaluate the sample size bounds on generalisation (which are introduced in the general case in Chapter 2) for the weightless systems the Vapnik-Chervonenkis dimension (VC dimension) is calculated. The results and proofs of these calculations for the simplest n-tuple based classifier, the discriminator, comprise Chapter 4.

In chapter 5 the analysis is extended to more complex weightless system and upper and lower bounds on VC dimension are derived for these systems. Since the exact results are hard, bounds based on functionality are applied. Some analysis of duplicated functions is required to prove the validity of these bounds. Finally the n-tuple classifier with two discriminators is reformulated as a single discriminator with three possible stored values and the possibilities that arise from extending this model are considered.

Knowing the VC dimension of the weightless systems we can apply the results of Chapter 2 to get results relating generalisation ability to training set size and network structure. These are presented in Chapter 6 and compared with some real-world results. A distribution dependent analogue of VC dimension, effective VC dimension, is described and calculated for various n-tuple classifier configurations. This helps to explain some of the looseness in the generalisation bounds. A comparison with other learning machines is also presented in this chapter.

Chapter 7 examines the tools used to tune the n-tuple classifier and the results obtained by so doing. An overview of techniques used by previous researchers is presented, followed by a resume of the problems of statistical inference in the data-independent "tabula rasa" case. Finally the n-tuple work is put into context and as far as possible explained in terms of statistical learning concepts. The role of data-independent classifiers is considered and introduction of a priori knowledge into the system is also discussed.

Finally Chapter 8 contains an overview of the thesis and offers such conclusions as may be drawn, as well as suggestions for how this work may extended in future.

Chapter 2

An Introduction to Learning Machines

Learning theory deals with the subject of learning machines, so implicit in the title of this thesis is the claim that weightless systems are learning machines. Before we attempt any analysis we must define what we mean by a learning machine and show that weightless systems are indeed examples of such. To show the scope of the technology of learning machines we shall first consider their history and finally the particular systems under consideration in this thesis, weightless systems, are described in detail.

2.1 What is a Learning Machine?

Before answering this question it is worth first discussing what is meant by the word "learning". First of all it should be made clear that in this context learning is not intended to represent any psychological or even physiological phenomena. It is a purely formal exercise and any parallels or analogies with organic learning systems should be considered purely as inspiration or accident.

So what is left? Well the idea of learning that the learning machines in this framework aim to capture is the notion of learning from examples. It is assumed that some examples are produced probabilistically and then transformed according to some unknown rule into some value: a yes/no if we are considering a pattern recognition task or something more complicated in general. The machine is allowed to see a small sample of the inputs and their associated output values and from these is supposed to make a "best guess" at unseen inputs. If its error on the unseen inputs - the test data - is good then the machine has generalised well and an observer may feel that it has "learnt" the "underlying structure" of the task. The ability to generalise well is the crucial difference between learning and memorising.

A learning machine can be defined informally as follows. A learning machine, $\mathcal{L}$, comprises a set of functions and a mechanism for selecting one of the functions - hopefully the one with least expected generalisation error - when shown a labelled sample of training data.(The functions are considered to be parametrised by $\alpha \in \Lambda$ where Λ is some arbitrary set.) The most important point about a learning machine is that we have limited knowledge about the problem it is going to be asked to solve. To be successful a learning machine it must be able to produce a function with good generalisation in a wide range of situations. In this it differs from traditional statistical methods which require that a model for the data be specified up to the value of a small number of parameters. The importance of the distinction is discussed in conjunction with the so-called "bias/variance dilemma" in section 7.2.1. It is with this definition in mind that we consider the history of learning machines.

2.2 History of Learning Machines

Of course the original learning machines were not computer-based but living. The inspiration for inanimate learning machines was an attempt to reproduce in computers the learning ability of human beings in an attempt to exploit the particular advantages computers offer: time, numerical power and accuracy, freedom from boredom et al.. It is probably no coincidence that the first learning machine was inspired by human anatomy.

McCulloch and Pitts were the first to suggest that the spiking of neurons may offer a model for computation [33]. However it wasn't until the perceptron algorithm of Rosenblatt [46] that the idea was put into practical operation and a true learning machine was created. That is, a machine which in could attempt to model many problems given limited data about them. Indeed it was proved by Novikoff [38] that the perceptron would always converge to the exact solution, if it existed. As mentioned in chapter 3 this is considered to be the first result in learning theory. The perceptron consisted of one unit, a "neuron", with a large number of afferent connections in analogy with a real neuron. Each connection had an associated weight and by either sending a signal or not down each line an input pattern could be encoded. The neuron then summed the weights on the active lines and thresholded the result to give a hyperplanar discrimination surface. The training algorithm adjusted the weights incrementally until the classification error on the training data was minimised (where possible) and returned the weights it found. This system was to be the grandparent of all the systems later to be known as "neural networks".

Subsequent attempts to extend the perceptron model to more layers, that is, to introduce neurons between the input and output, were made in order to get round the limitations of the set of functions that the simple perceptrons could realise. While extending the architecture was a straightforward task, developing an algorithm to enable the system to learn was not. The presence of neurons whose required output was not known (because they did not connect directly to either the input nor the output) meant that Rosenblatt's algorithm could not be applied directly. Moreover, the usual solution applied to such tasks, the method of gradient descent, could not be used because the threshold function used by the neurons did not have a gradient at the critical point.

The problem was finally solved for multi-layer perceptron in 1986 [29], [47] (although the algorithm was discovered in 1963 in [15] for a control problem). The simple idea they had was to replace the linear threshold function with a continuous (and continuously differentiable) function which would approximate it. Such a function should tend to 1 as its argument tends to ∞ and to -1 or 0 as the argument tends to

$-\infty$. Such functions are known as sigmoids and $S(u) = \tanh(u)$ is one of the most commonly used. It then becomes possible to apply gradient descent methods based on knowing the input and required output and looking for a local minimum of the network error.

Other learning machines have also been developed, often in an ad hoc way to solve particular engineering problems. Some of these are usually classed as "statistical" methods and others as "neural" even though underlying principles may be the same. The difference tends to be the inspiration rather than the functionality. Since weightless systems are the subject of this thesis, and since they are classed as "neural networks", we shall take a further look at these systems.

2.3 Neural Networks

The reborn perceptron was restyled a "neural network" to emphasise the biological inspiration of the architecture and huge research effort was subsequently born. However, many machines which were inspired by but not based on the linear perceptron had been created during its period of dormancy and these too were included in the class of "neural networks". In fact almost all learning machines that were not fundamentally grounded in statistics were put into this class. By the mid-90s these included, in approximate order of development: Adaline/Madaline [62], Learning Matrices [51], ART [23], Topological Maps [28], Sparse Distributed Memory [27], Radial Basis Function [40] and, out of sequence for rhetorical reasons, Weightless Systems.

So what defines a neural network? The definition is both fuzzy and evolving. The only common external feature of neural networks is the ability to generalise from sets of labelled examples (ie. they are learning machines). Internally they are expected to contain a "large" number of identical processing units which share information amongst themselves in a "massively parallel" fashion. The training (learning) algorithm is usually an attempt to minimise the error on the training set by incremental adjustments but this is not always true, see weightless systems and sparse distributed memory.

The term "neural networks" has for several years been a convenient way of linking together all the new models of learning while emphasising their basis in neurophysiology. Indeed, the analogy is often extended to the point where it is claimed that since the architecture of neural networks more closely approximates the architecture of the brain so neural networks are intrinsically better suited to cognitive tasks. Perhaps a better way to consider the analogy is to note that learning machines emulate some of the functions of the brain better than artificial intelligence methods and so they may be more likely to solve the human-type "cognitive" problems that old AI techniques have left unsolved. If this is indeed the case then since neural networks are learning machines they are not excluded, but also whole new classes of machines and algorithms with firm statistical backing can be introduced to the problems facing neural networkers in all fields.

2.3.1 Perceptrons and Incremental Methods

The multi-layer perceptron (MLP) is the workhorse of practical neural network (and hence learning machine) research and application and so it is worth taking time to understand it in order to contrast it with the weightless systems studied in the thesis.

Each layer of the perceptron maps an input vector to an output vector according to a sigmoid function. Formally,

$$\begin{aligned} l : \mathcal{R}^n &\rightarrow \mathcal{R}^m \\ \mathbf{x} &\mapsto \mathbf{y} \end{aligned}$$

where

$$y_i = \sigma(\mathbf{x}.\mathbf{w}_\mathrm{i})$$

and σ is a sigmoid function. The vectors $\mathbf{w}_\mathrm{i}$ are known as the weight vectors. It is these which most training algorithms adjust. (Some also alter the sigmoid function for each y_i.) It has been proved theoretically by Cybenko [17] that any smooth function can be approximated arbitrarily closely by the weighted sum of sigmoid functions so there exists a perceptron to solve any such problem. There are two difficulties with this in practice. The first is that the approximation result is only asymptotically true. That is, any function can be approximated with arbitrary accuracy, but only if arbitrarily large perceptrons are used. In fact recent work [64] has shown that when implemented on computers, perceptrons are only able to function as well as polynomial approximators because of the finite nature of their implementation. The second, more mundane, difficulty rests in finding a local minimum which is as close to the global minimum as to be acceptable without doing an exhaustive search of the set of weight vectors. In practice this can often be achieved satisfactorally with the aid of rules-of-thumb which are often known from AI and linguistic usage as *heuristics*. However, these do not form a principled mathematical approach to solving a given problem and learning theory is one attempt to put such heuristics on a sound footing. Current theoretical results are outlined in chapter 7.

Another system which has recently attracted interest is the radial basis function network. In such networks the sigmoid functions of the MLP are replaced by analytically more tractable symmetric functions, usually Gaussian distribution functions. Again it has been proved that these systems can approximate any function [21]. The learning algorithms concentrate on finding the variance of a set a basis functions whose mean or *centre* is fixed, although moving the centres allows more flexibility.

Other learning systems such as Kohonen's topological maps aim to model distributions by deforming grids to represent incoming data or in more complex cases by birth/death process amongst nodes in the grid to optimise the model. All that need interest us for this thesis is that the definition of a learning machine is fulfilled by a large number of object in practical use today and that it is against these that weightless systems must compete, both in theory and in practice.

2.4 Weightless Systems

All the systems which described here as weightless are so-called in contrast to perceptron-like systems where training is accomplished by means of adjustments of the parameters called weights. The free parameters of a weightless system almost all reside in the nodes themselves. Amongst the systems here called weightless (and thus to which the results of this thesis can, to a greater or lesser extent, be applied) are the following: n-tuple recognition devices, weightless discriminators, WISARD, PLN pyramids, GNUs, GSN. Sparse distributed memory is sometimes described as a neural model and it has a certain amount in common with weightless systems, however it is too different for results about other weightless systems to be easily applied to it.

The grandfather of these systems, standing in the same position to them as the perceptron does to weighted systems, is the n-tuple method of pattern recognition of Bledsoe and Browning [12]. In the 60s and 70s the idea was refined into that of the discriminator and subsequently rendered in hardware as the WISARD [5]. The discriminator then gave birth to a family of systems designed or inspired by Aleksander, amongst which are the **probabilistic logic node** (PLN), the **generalising random access memory** (GRAM) and the ensemble of GRAMs, the **general neural unit** (GNU). Before going on to describe their abilities and their shortcomings we shall describe the systems in some detail.

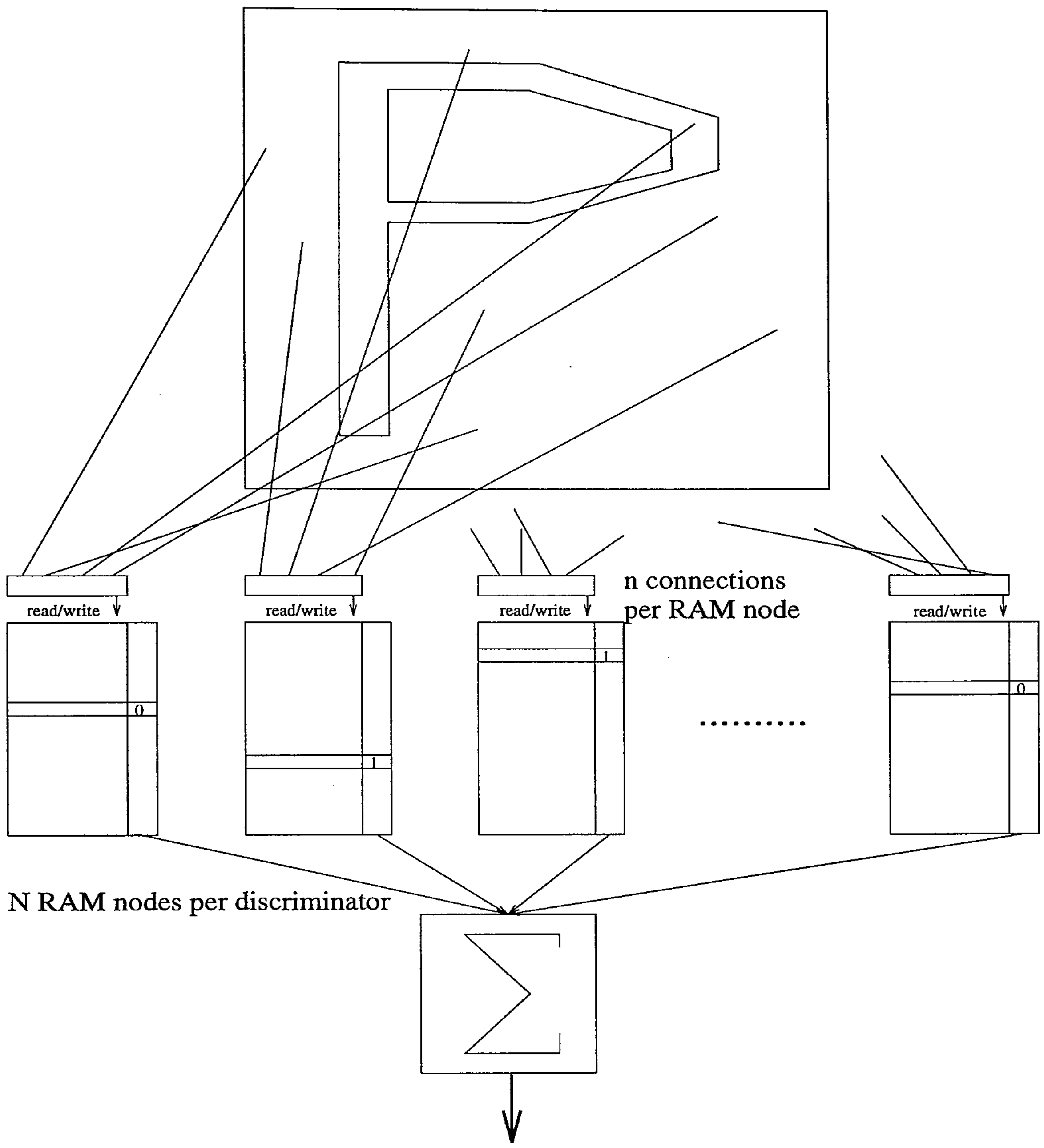
2.4.1 The N-Tuple Method

The first n-tuple classifier was the work of W.W.Bledsoe and I.Browning of the Sandia Corporation of New Mexico. The problem addressed in their paper is that of character recognition, but here will be described as a general pattern recognition task. The work was followed up by others such as Bledsoe and Bisson [11], Ullman [56] and Ullman & Kidd [57] often still in the field of handwritten character recognition. The framework of the pattern recognition task is as follows. The inputs consist of long binary input vectors (usually representing visual images) which must be classified into one of a small number of classes. This is not a clustering algorithm and the number of classes must be known before the task is started. A set of correctly classified input patterns is presented to the machine which is subsequently tested on further examples and the classification error measured.

The n-tuple machine works by a process of sampling the input space, X, in small groups of points. There are N nodes which are capable of performing any boolean function from $\{0,1\}^n \to \{0,1\}$ for some integer n. The input to each node is a randomly chosen projection from the whole input space onto $\{0,1\}^n$. The projection is fixed at the start of the learning machine's operation and can be thought of as a physical or virtual connection from n of the input pixels onto the node. This is the n-tuple that gives the method it name.

A group of N n-tuples, and hence N nodes, is assigned to each class. Although the term is a later one, a group of n-tuples responding to the same class will be termed a **discriminator**. To complete the definition of the discriminator we must add a summing device which takes the binary outputs of the N nodes and sums them to produce an integer output between 0 and N. This integer can be seen as approximating the likelihood that the input is in the class assigned to that discriminator, [35], [50]. The output of the whole n-tuple learning machine is then defined to be the class name of the discriminator with the highest output.

Training must necessarily involve defining the node functions which, since all 2^{2^n} functions are allowable, can be thought of as filling a look-up table at each node. Several different training methods are described. All methods start the learning process with all nodes for all classes performing the **zero** function (ie. responding with **0** for all inputs). The Bledsoe and Browning approach which was later echoed by Aleksander and Stonham [4] is to then present each labelled training pattern to the discriminator pertaining to its class and to fill the all the node table entries corresponding to the n-tuple it projects with a 1, whether the current entry is 1 or 0. Alternatively we can think of this as ensuring that the all the node functions



The summed output may be thresholded
or compared with other discriminators.

Figure 2.1: The n-tuple Discriminator (schematically).

output one when presented with the particular n-tuple in the training pattern.

Problems with the Training

This method of training ensures that the discriminator pertaining to the correct class of each training example will always output N when the pattern is presented again. Problems arise if so many node table entries are set to 1 that the pattern also yields N in other discriminators. This problem of the "filling-up" of node table tables as the training progresses is known as **saturation**. It is most serious when the data sets are large and the data in each class has a large variance. We can consider it as being a tendency of the node functions to approach **one**, the constant function that always outputs **1**. As this happens those training patterns which should output **0** begin to output **1**, so error on the training set is not minimised. Unless this is done in order to minimise some structural feature of the model (as in SRM in section 3.4) this is will in general give sub-optimal generalisation. Tarling and Rohwer [53] developed a new algorithm to overcome the problems of saturation.

Reducing Saturation

Training in the Bledsoe and Browning sense involves maximal saturation. That is, all possible **1** values are set. The method of Tarling and Rohwer is to check before presenting each training pattern as to whether it is already classified correctly. Only if it is not are the node value tables set to **1**. This has the effect of reducing saturation if the training set is large, but may reduce generalisation if it is small. The results they obtain for the new method of training are generally positive. A detailed examination of the issues involved is given in chapter 7.

Maximum Likelihood Training

It has been noted in several papers that the n-tuple classifier can be interpreted as a crude maximum-likelihood estimator. Bledsoe and Bisson [11] were the first to try to use this observation to create a new training algorithm. The idea was used subsequently by several other researchers such as Rohwer [35], Badr [9] and Tattershall et al. [50]. There are certain problems with justifying the approach statistically, especially with sparse data, but the method has enjoyed some empirical success.

2.4.2 Thresholded Discriminators

The discriminators defined above for use in the n-tuple classifier can be treated as learning machines in their own right . Their output is an integer between 0 and N so to turn them into machines implementing the $\{1, 0\}$ hypotheses we are considering, a threshold between these values is required. Discriminators are rarely used in this way for practical applications, but their analysis can help in the understanding of the n-tuple classifier. They are discussed in detail in chapters 4 and 5.

2.4.3 WISARD

The **WI**lkie **S**tonham and **A**leksander **R**ecognition **D**evice was presented in 1982 as a practical realisation of the n-tuple method (see [5]). Each node was modelled by a random access memory (RAM) and the values for each discriminator could then be summed and compared. The operation is exactly analogous to the n-tuple method but the hardware implementation meant that it could be employed to process data in real-time and even video images. The use of RAMs as nodes led to the widespread use of the term for nodes in all weightless systems even when implemented in software. The term RAM-based systems is often used as a synonym for weightless systems.

2.4.4 PLNs

The **probabilistic logic node**, PLN, is an extension of the normal deterministic two-valued node used in the systems described so far. In a PLN, the output set is extended to a finite set with more than two elements. These values are then interpreted probabilistically in order to give a **1** or **0** output. PLNs with more than three probabilistic output values are known as MPLNs. They are described and used by Myers in [36]. The standard PLN has three probabilistic output values **0**,**1** and **u** where a **u** value is interpreted by the node as a probabilistic function giving **0** and

1 with equal probability.

A popular architecture for PLNs has been the form of a feed-forward pyramid where the output of one layer feeds into the layer below, see [36], [3], [18]. A reward/penalty algorithm has been applied and been shown to allow learning of many complex functions.

2.4.5 GRAMs and GNUs

The **generalising random access memory**, the GRAM, is a development of the standard three-valued PLN which attempts to add "more" generalisation. Training starts with all node values set to **u** and these are then modified according to the architecture and training regime. After training is complete, each trained address "spreads" its stored value to nearby untrained addresses. This spreading occurs up to a radius defined by the machine design, and follows well-defined rules if two spread values clash, see [26]. However, the radius within which spreading occurs must be pre-determined and it is far from clear how performance at certain tasks is changed by spreading.

The **general neural unit** GNU is an assembly of GRAMs connected so that some input to each node comes from the usual external input, while some comes as feed-back from the outputs of the GRAMs. This entails the ensemble maintaining a memory of its last response and makes these systems very different to the previous arrangements of weightless nodes. (Although we see in section 4.5 that some results carry well between the two.) A diagram of the basic architecture is given below.

A GNU can function as an associative memory for high-dimensional binary vectors whether auto- or hetero-associative, but detailed performance statistics are not available to compare it with other, similar, systems. It is hoped that the GNU will be able to analyse complex, time-dependent inputs where its feedback connections will be able to provide context information. However it will only be treated in passing in this thesis, and only for the pattern recognition it can perform as an auto-associator

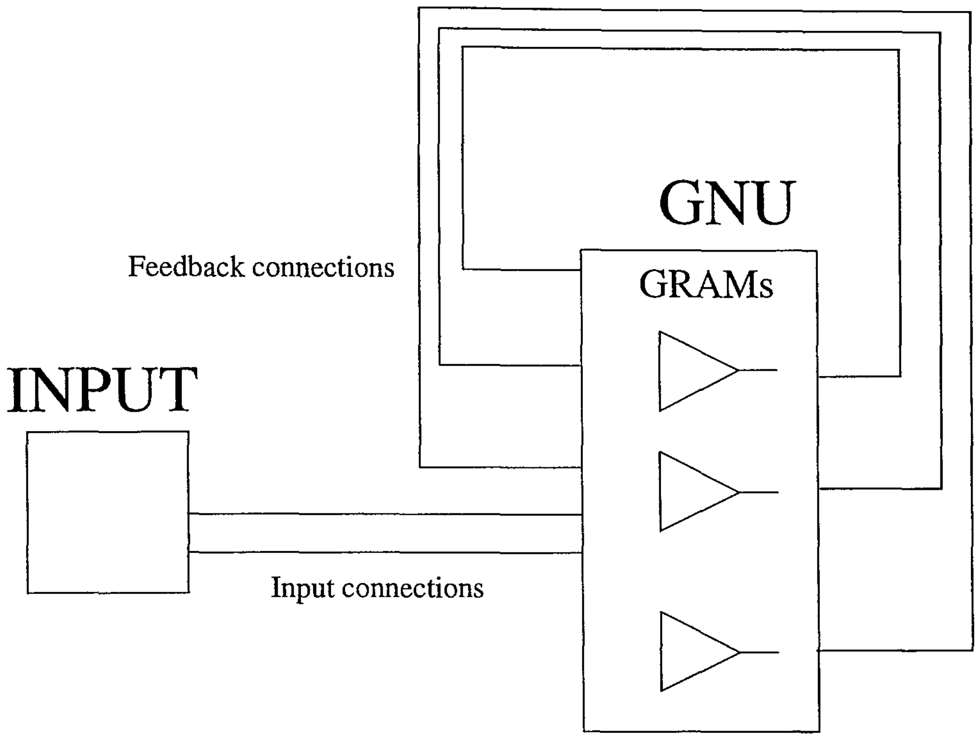


Figure 2.2: The weightless general neural unit.

when recall is simply treated as recognition.

2.4.6 Other Systems

Other systems within the weightless framework are in use such as ADAM (the Advanced Distributed Associative Memory) [8] which is designed for image analysis tasks and the BCN series [24]. However none of these will be analysed in this thesis so no more details will be given. Hopefully though, this exposition of weightless technology serves to demonstrate the wide range of different systems, algorithms and techniques.

Chapter 3

An Introduction to Learning Theory

3.1 Introduction

Vapnik [58] dates the beginning of learning theory to the work of Rosenblatt [46] and his perceptron. This early version of what we now call a neural network was the amongst the first implementations of algorithms which could learn from examples and generalise: what scientists call induction. Certainly his work produced a lot of interest in subject of adaptive learning machines and when Novikoff [38] produced the first theorem showing that the perceptron could separate any separable set this became the first result in learning theory. Moreover it was shown [2] that given an appropriate stopping rule the error can be guaranteed to be below any required value with any required probability. These results seemed to show that minimising the error of a learning machine on a training set was the best way to ensure minimal error on the whole input space. The principle this embodies is known as the Principle of Empirical Risk Minimisation or ERM. We shall return to this when we start to formalise our reasoning.

At this point two schools of thought (which were later to become those of Neural Networks and Learning Theory) began to diverge. One group decided to accept the ERM principle and search for better ways to minimise training error and different and more flexible systems. Any adjustment of architecture or system size was considered an engineering problem and many such problems were solved quite successfully. The other, more theoretical, group began to ask themselves the questions outlined in Chapter 1 about the conditions under which the ERM principle does indeed converge to the optimal solution and if so how fast. At the present time, when practical learning machines have shown themselves to be powerful practical tools but whose theory has severe lacunae, the two strands of research are meeting again.

3.2 Risk Minimisation

In this section we shall begin to lay out the formal framework of learning theory: both what it's trying to achieve and the concepts used to describe it. The fundamental problem of learning theory is the same as that of statistics, namely to estimate the properties of an unknown probability distribution using a limited number of data points. This is exactly the same problem that neural networks of all sorts have when given training data to solve a problem. The particular aspect of the question addressed by learning theory is how to make these estimations given a restricted number of functions which are parametrised by an arbitrary set when the only information about the input and output distributions is that contained in the training set. (This distinguishes learning theory from more traditional statistics where the distribution to be modelled is known up to the value of a few parameters which must be estimated.) This set of functions is equivalent to a learning machine, and familiar learning machines such as neural networks and radial basis function networks define such sets of functions: in the first case as being the sums of various sigmoid functions and in the latter as being sums of various symmetric kernel functions.

The algorithm chosen for selecting the best function available in a machine to match a given data sample, or "training" data in neural networks terms, can be in most cases be described by the ERM principle. The study of this principle and subsequently of its successor, the Statistical Risk Minimisation principle, is thus the natural focus of learning theory. However to explain properly what is meant by these principles we must first make clear what we mean by a statistical risk.

3.2.1 The Learning Problem

To formalise the ERM principle we must define the learning problem itself in the most general and formal terms. We assume an input set X and an output set Y. We have a probability measure F on X which corresponds to the probability of a given random generator, G, choosing any particular $x \in X$ in the training and test data. For the purposes of this work this probability distribution is assumed to be constant throughout the learning and test processes. We furthermore assume a conditional probability measure $F(y|x)$ which defines the (probabilistic) rule for the input/output mapping which the machine must model.(Note that this may be given by a deterministic function $y = y(x)$ and in many real-life cases will be. However we shall keep the analysis as general as possible, for instance so that noisy data can be introduced.) We assume a supervisor S which is able to deliver training and test data according to this rule. This supervisor is in real-life the sampling process that generates test and training data. The third part of the model is the learning machine itself which consists of a set of functions $f : X \times \Lambda \rightarrow Y$ where Λ is an arbitrary index of the available functions.

3.2.2 Expected and Empirical Risk

The above defines the underlying structure, but to define what we want the learning machine to do, we need some idea of what constitutes a "good function" in the learning machine. To do this we introduce the risk functional. First we define a **loss function**, $L(y, f(x, \alpha))$ which is some measure of the incorrectness of the function parametrised by $\alpha \in \Lambda$ at point y in the output space. Different choices of loss function correspond to different statistical problems. The **risk functional**, R, is the value of the loss function integrated over the whole output space. That is

$$R(\alpha) = \int L(y, f(x, \alpha))dF(x, y).$$

A particular value of R is known as the **expected risk** for parameter α. Good generalisation by a function which has been selected by reference to a training sample can be characterised by a low value of the expected risk. Minimising the expected risk of the learning machine (or the function selected by it) is the goal of any learning machine.

Now let z denote a pair (x, y) and let $Q(z, \alpha)$ be the loss associated with function α at z. We may then define the empirical risk over a set of data $\{z_i\}_{i=1}^{l}$ to be

$$R\text{emp}(\alpha) := \frac{1}{l}\sum_{i=1}^{l} Q(z_i, \alpha).$$

This represents the average error of the function parametrised by α on the training set. By the law of large numbers we can see that the empirical risk will tend to the expected risk as l, the size of the sample, is increased. However, the goal of the learning machine is to select the function that minimises the expected risk over *all* $\alpha \in \Lambda$.

The **Empirical Risk Minimisation** principle can now be stated as follows:

In order to minimise the expected risk over the set of functions provided by the learning machine, one should attempt to minimise the empirical risk on the training data.

This principle will be discussed in detail after some practical examples. From now on it is assumed that X and Y are some real spaces $\mathcal{R}^n$ and $\mathcal{R}^m$.

3.2.3 Common Risk Functions

The choice of loss function determines the sort of information that the learning machine attempts to extract from the data. Three of the most common examples are used for pattern recognition, regression analysis and density estimation.

Pattern Recognition

This is the simplest case and all the analysis of weightless systems in this thesis considers this problem. The only allowable values of y are 0 and 1 and the loss function is given by

$$L(y, f(x, \alpha)) = \begin{cases} 0 & \text{if } f(x, \alpha) = y \\ 1 & \text{if } f(x, \alpha) \neq y. \end{cases}$$

The loss function in this case simply counts the number of classification errors and the expected risk is the probability of wrong classifications on the rest of the data. Hence the problem of minimising *expected* risk is indeed the classical problem of finding the best classifier from the functions available.

Regression Analysis

General regression problems are not considered in detail in this thesis because n-tuple classifiers have only rarely been used in such situations. However as an example of a risk function it shows the scope of the preceding formalisation of the learning problem.

To perform a regression in the standard model one must minimise the mean squared error, so the appropriate loss function is

$$L(y, f(x, \alpha)) = (y - f(x, \alpha))^2.$$

Density Estimation

As a final example we consider a case where the input/output mapping is irrelevant and only the input distribution is of interest. According to the standard model [10, page 59] if we want to approximate the density function on X (and here we ignore Y) we must minimise the loss functional given by

$$L(p(x, \alpha)) = -\log p(x, \alpha).$$

3.2.4 Empirical Risk Minimisation

The ERM principle is central to most learning algorithms and so its study comprises a large part of learning theory. Multi-layer perceptrons, for example, minimise the error on their output nodes by back-propagation of error during training, while genetic algorithms do the same at the selection stage. Part of the aim of this thesis is to show how the n-tuple classifier can be put into the same framework and that this framework allows us new theoretical results and insights into the classifier's behaviour.

Several questions naturally arise from the definition of the ERM principle:

1. Does it yield a sequence of functions that asymptotically converge to the one with minimum expected risk?
2. Does it guarantee good generalisation (ie. low expected risk) for small sample sizes?
3. Can we improve the expected risk by controlling the machines capacity and preventing "over-fitting"?
4. Can learning machines based on other principles be as good or better?

Consistency of the ERM Principle

The question of asymptotic convergence is known as the question of the **consistency** of the learning process. It has been shown [61] that the ERM is consistent for a set of functions if and only if the maximum value of the difference between the empirical risk and the expected risk converges uniformly to zero in probability as the sample size tends to infinity. More precisely

$$\lim_{l\to\infty} P\{\sup(R(\alpha) - R\text{emp}(\alpha))\} > \epsilon,\ \forall\, \epsilon > 0. \tag{3.1}$$

In particular this shows that the ERM principle is always consistent for finite sets of functions, such as those produced by n-tuple classifiers and GNUs. However, because the thesis is concerned with such learning machines only, we will concentrate on the non-asymptotic, small sample size part of the theory.

Generalisation

One of the major benefits of this formalisation is that we have a precise notion of generalisation. We can say that a function generalises well if its expected risk is low. Thus we can say that a learning machine generalises well from a training set if the function selected by the learning machine after training has low expected risk. The question of good generalisation for small sample sizes is an important one in practice where a machine may be asked to learn from a fixed given set and to provide a function or classification with a low expected risk. Given no information about the problem, can we bound the expected generalisation error? The answer is that for all most all learning machines (those without infinite VC dimension, see section 6.5) such a bound can be found. These bounds are given in section 3.3.

Controlling the Capacity of a Learning Machine

The answer to question 3 above is an emphatic "Yes". The phenomenon of over-fitting is a special case of the theory of "ill-posed problems" and the tools of regularisation theory have been developed to solve this and similar difficulties. Techniques for achieving this goal for general learning machines are often ad hoc, but analysis in the context of learning machine gives a particular method of restricting over-fitting problems by bounding the generalisation error of a learning machine by a function of a scalar quantity the *VC dimension* and controlling it over a sequence of learning machines of increasing capacity. The theory of this procedure and related ideas is given in section 3.4.

Non-ERM machines

Learning machines which do not implement the ERM principle do exist. Stochastic approximation [43] is one such method of inference which minimises expected risk by a sequence of iterations based on each training pair. However the properties of the ERM principle are sufficiently attractive that most current learning algorithms execute some version of it. In particular it can be shown that the n-tuple method can be analysed in this framework.

It should be noted that some learning algorithms do seem to violate ERM, for instance by ignoring outliers, but by doing so they introduce a priori assumptions about the data, for instance that outliers are likely to be the result of noise or unexplained error. From a practical point of view such assumptions may be very useful, but in the context of the formal problem we cannot resort to such techniques. The only information about the unknown distribution is that contained in the training data. We shall look again at the idea of incorporating a priori knowledge in section 7.2.4.

3.3 Bounding the Expected Risk

In order to be sure that a learning machine generalises well for small sample sizes we need tight bounds on the expected risk given the sample size and machine type. To get tighter bounds still we can use information about the distribution of the data if this is available. This information about the set of functions comes in four flavours (and more can be devised). They are

- VC-Entropy
- Annealed VC-Entropy
- Growth Function
- Generalised Growth Function.

(The 'VC' stands for Vapnik-Chervonenkis who first proposed these quantities in [60].)

3.3.1 Non-Constructive Bounds on Generalisation

From now on this thesis will only be concerned with the theory pertaining to pattern recognition as this is the main capability of n-tuple classifier and most other weightless systems studied in this thesis. For such a set of functions (that is, ones which take values in {0,1}) we can make the following definitions. The notation used below is that of section 3.2.1.

Capacity Definitions

Let us consider a learning machine $\mathcal{L}$ characterised as before by the set of boolean functions that it can perform on its input space. Denote these functions by $Q(x, \alpha)$ where $\alpha \in \Lambda$ and Λ is an index set. Let $(x_1, x_2, ..., x_l)$ be an input sample and let $N^{\Lambda}(x_1, x_2, ..., x_i)$ be the number of dichotomies that $\mathcal{L}$ can realise on the set. That is, the number of binary vectors in the set

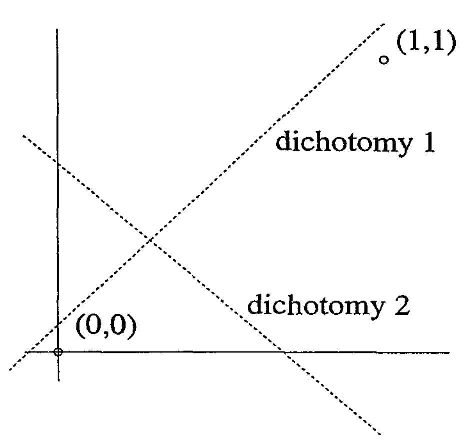
$$\{(Q(x_1, \alpha), Q(x_2, \alpha), ..., Q(x_l, \alpha)) \mid \alpha \in \Lambda\}.$$

Example 3.1 *Let us take as an example the linear classifier in two dimensions where the input set is the set of vertices of the unit square together with its centre point* $(\frac{1}{2}, \frac{1}{2})$ *and the separating hyperplane is just a line. Then if we consider the sample* $\mathrm{x} = \{(0,0),(1,1)\}$*,* $N^{\Lambda}(\mathrm{x}) = 4$ *since all* $2^{|\mathrm{x}|} = 4$ *possible classifications of* x *are possible, see figure 3.1. (Note that one line yields two dichotomies because either side of the line can be set to* **1** *or* **0***.) If* x *were to be extended to include* $(1,0)$ *and* $(0,1)$ *then we would have* $N^{\Lambda}(\mathrm{x}) = 14$ *and not* $2^{|\mathrm{x}|} = 16$ *because the dichotomies where* $(0,0)$ *and* $(1,1)$ *take the same value as each other but different to the other two input point are not available (figure 3.2).*

If we consider the samples as being chosen according the unknown probability distribution F we can consider $N^{\Lambda}(x_1, x_2, ..., x_l)$ as a random variable. We can then define **VC-entropy**, H^{Λ} as follows.

$$H^{\Lambda}(l) = E[\ln N^{\Lambda}(x_1, x_2, ..., x_l)].$$

Example 3.2 *Let us return to our previous example and consider VC-entropy. For* $l = 1$ *there is only one possibility, a singleton point. In this case both dichotomies of the set are available, so* $H^{\Lambda}(1) = 2$*. When* $l = 2$ *there are two possibilities, two distinct points or two choices of the same point. In the first case* $N^{\Lambda}(\mathrm{x}) = 4$ *and in the latter* $N^{\Lambda}(\mathrm{x}) = 2$*. The expected value of the number of dichotomies is the*



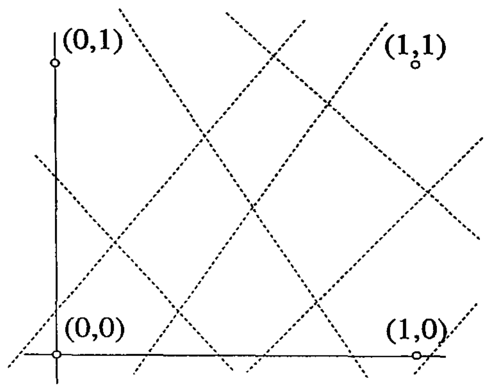


Figure 3.1: All four dichotomies (given by two lines) of two points.

Figure 3.2: The fourteen available dichotomies of the four corners of the unit square.

sum of the number of dichotomies for each set, weighted by the probability of that set occurring. Hence $H^{\Lambda}(2) = (\ln 4)P[\textit{two points different}] + (\ln 2)P[\textit{two points same}]$. *It is not hard to see that such calculations soon become very convoluted and for this reason are little used in practice.*

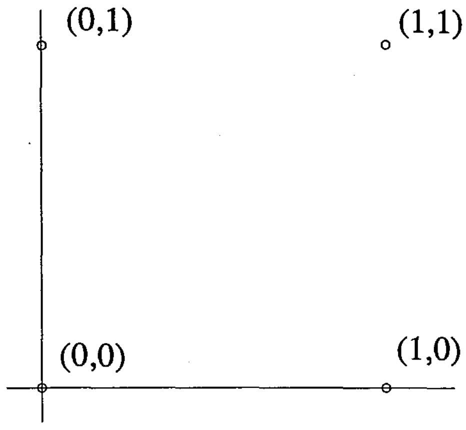
Furthermore the ***annealed*** **entropy** is given by

$$H^{\Lambda}_{\text{ann}}(l) = \ln E[N^{\Lambda}(x_1, x_2, ..., x_l)].$$

And the **growth function** is

$$G^{\Lambda}(l) = \ln \max_{x_1, x_2, ..., x_l} [N^{\Lambda}(x_1, x_2, ..., x_l)].$$

Example 3.3 *Once again considering the running example, we can see that the growth function takes the following values. (Note that if the classifier can split a set in any particular way that such a split generates two dichotomies in which the outputs all take opposite values.)*
Any singleton set can be classified either **1** *or* **0** *so* $G^{\Lambda}(1) = \ln 2$.
Any two points can be separated by a line, and either point may be classified as **1** *or* **0** *so* $G^{\Lambda}(2) = \ln 4$.
Any three non-collinear points can be separated in any configuration by a line, so, for instance, $\{(0,0),(0,1),(1,0)\}$ *can be shattered by a linear classifier. Thus* $G^{\Lambda}(3) = \ln 8$.



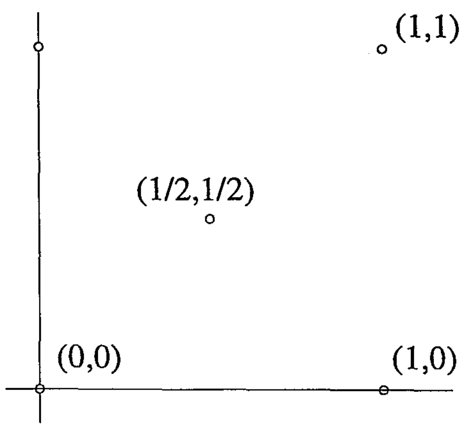


Figure 3.3: (0,0) and (1,1) cannot take the opposite value to (0,1) and (1,0).

Figure 3.4: (0,0) and (1,1) cannot take the opposite value to (1/2,1/2).

This is the first case where no subset of the input set can be dichotomised in all possible ways. We must consider which dichotomies can and can't be realised. In the case of a convex set of four points, any single point can be separated from the others, and two of the three separations of two points from the other two can be made. However the third separation of the type 2:2 cannot be made (see figure 3.3). This leaves two dichotomies unavailable. In the case of a set of four points with three points the collinear point between the other two (ie. (1/2,1/2)) cannot be separated from both its endpoints simultaneously. Again two dichotomies are not realisable (see figure 3.4).. Hence the maximum number of dichotomies of a four element subset of our input set is 14 so $G^{\Lambda}(4) = \ln 14$
There is only one five element subset of the input set and investigation shows that 18 dichotomies of it are realisable. Thus $G^{\Lambda}(5) = \ln 18$

These three quantities form an increasing sequence.

$$H^{\Lambda} \leq H^{\Lambda}_{\text{ann}} \leq G^{\Lambda}. \tag{3.2}$$

Although it is not relevant to the finite sets of functions studied in this thesis, the quantities are implicit in the asymptotic behaviour of a learning machine which follows the ERM principle. In fact

$$\lim_{l\to\infty} \frac{H^{\Lambda}(l)}{l} = 0$$

is a sufficient condition to guarantee asymptotic convergence to the best solution. (The necessary condition is very slightly weaker.)

The condition

$$\lim_{l\to\infty} \frac{H^{\Lambda}_{\text{ann}}(l)}{l} = 0$$

is sufficient (the necessity is at the moment an open question) for "fast" convergence in the sense defined in section (3.3.1).

Finally

$$\lim_{l\to\infty} \frac{G^{\Lambda}(l)}{l} = 0$$

is necessary and sufficient for convergence for any distribution on the input space. This is the most general criterion we have.

Note also that the sequence in expression (3.2) ensures that the conditions form an increasing set of conditions. If one is fulfilled then so are those to the left. These are what Vapnik calls the "milestones of learning theory", [58, page 52]. However, since we are here interested in finite sample size behaviour we shall consider the use of these quantities in bounding the generalisation error (expected risk) of learning machines.

Generalisation Bounds

The following results due to Vapnik [58] refer to learning machines trained according to the ERM principle.

$$P[\sup_{\alpha\in\Lambda} |\int Q(z,\alpha)dF(z) - \frac{1}{l}\sum_{i=1}^{l} Q(z_i,\alpha)| > \epsilon] \leq 4\exp\{(H^{\Lambda}_{\text{ann}}(2l) - \epsilon^2 l)\}. \tag{3.3}$$

If the probability distribution on the input space is unknown then we cannot use the annealed entropy. We must use the growth function instead. This is the importance of the growth function. It is used to bound the generalisation error when the distribution is unknown. Specifically

$$P[\sup_{\alpha\in\Lambda} | \int Q(z,\alpha)dF(z) - \frac{1}{l}\sum_{i=1}^{l} Q(z_i,\alpha)| > \epsilon] \leq 4\exp\{(G^{\Lambda}(2l) - \epsilon^2 l)\}. \tag{3.4}$$

We can rewrite this expression what is often a more useful form. First define

$$\mathcal{E} := 4\frac{G^{\Lambda}(2l) - \ln(\eta/4)}{l}.$$

It is then true (see [58])that for any $0 < \eta < 1$ and for all $Q(z,\alpha)$ the following results hold with probability $1-\eta$:

$$R(\alpha) \leq R\text{emp}(\alpha) + \frac{1}{2}\sqrt{\mathcal{E}} \tag{3.5}$$

and

$$R\text{emp}(\alpha) - \frac{1}{2}\sqrt{\mathcal{E}} \leq R(\alpha). \tag{3.6}$$

Moreover, for the function $Q(z,\alpha_l)$ which actually minimises the empirical risk on the sample (of length l) we have that the following holds with probability $1-2\eta$:

$$R(\alpha_l) - \inf_{\alpha\in\Lambda} R(\alpha) \leq \sqrt{\frac{-\ln\eta}{2l}} + \frac{1}{2}\sqrt{\mathcal{E}}. \tag{3.7}$$

3.3.2 The VC Dimension

There is a theorem due to Vapnik and Chervonenkis [60] (and independently to Sauer [48]) which shows the growth function to have a special form, which fact allows us to bound the growth function for all values of l using a single parameter, h. This parameter is known as the **VC dimension**. First we give the definition of VC dimension and then the statement of the theorem.

VC Dimension

Recall that the growth function, G^{Λ}, of a machine is the natural logarithm of the maximum number of dichotomies that the learning machine can realise on a sample of size l. (The maximum is taken over all samples of size l.) The value of the growth function is necessarily less than or equal to $l \ln 2$ since there are exactly 2^l possible dichotomies of a sample of length l. Thus we may define

$$h(\Lambda) := \text{VC dimension } (\mathcal{L}) = \max \{ \ l \text{ such that } G^{\Lambda} = l \ln 2 \}.$$

If no such l exists we say that the VC dimension of $\mathcal{L}$ is infinite. If a set of functions is able to dichotomise a set of size l in all 2^l possible ways it is said to **shatter** the set. The VC dimension of a set of functions can thus be expressed as the size of the largest set that can be shattered by the functions.

If the set of functions has finite cardinality N (as in most weightless systems) we have an immediate upper bound on h. Since 2^l distinct hypotheses are required to shatter a sample of length l, we must have $h \leq \log_2 N$. However h could be much smaller than $\log_2 N$.

Sauer's Lemma

For any $l > 0$

$$\exp[G^{\Lambda}(l)] < 1 + \binom{l}{1} + \binom{l}{2} + \ldots + \binom{l}{h} < \left(\frac{el}{h}\right)^{h}. \qquad (3.8)$$

For $l \leq h$ this simply states that $G^{\Lambda} \leq h \ln 2$ which is clear from the definition of h. However for $l > h$ the polynomial defined by the right-hand side of expression (3.8) is a polynomial in l of degree h. Thus if we know (or can bound) the VC dimension of a learning machine we have a logarithmic upper bound on the growth function. This can be used in the generalisation bounds of the learning machine to give a constructive upper bound to the expected risk for small sample sizes.

The Form of the Growth Function

Informally, Sauer's lemma shows that for a set of functions with a finite VC dimension, the growth function is linear (with slope $\ln 2$) for $l < h$ and is thereafter bounded by a function logarithmic in l^h. Some possible and impossible forms of the growth function are sketched in figure 3.5.

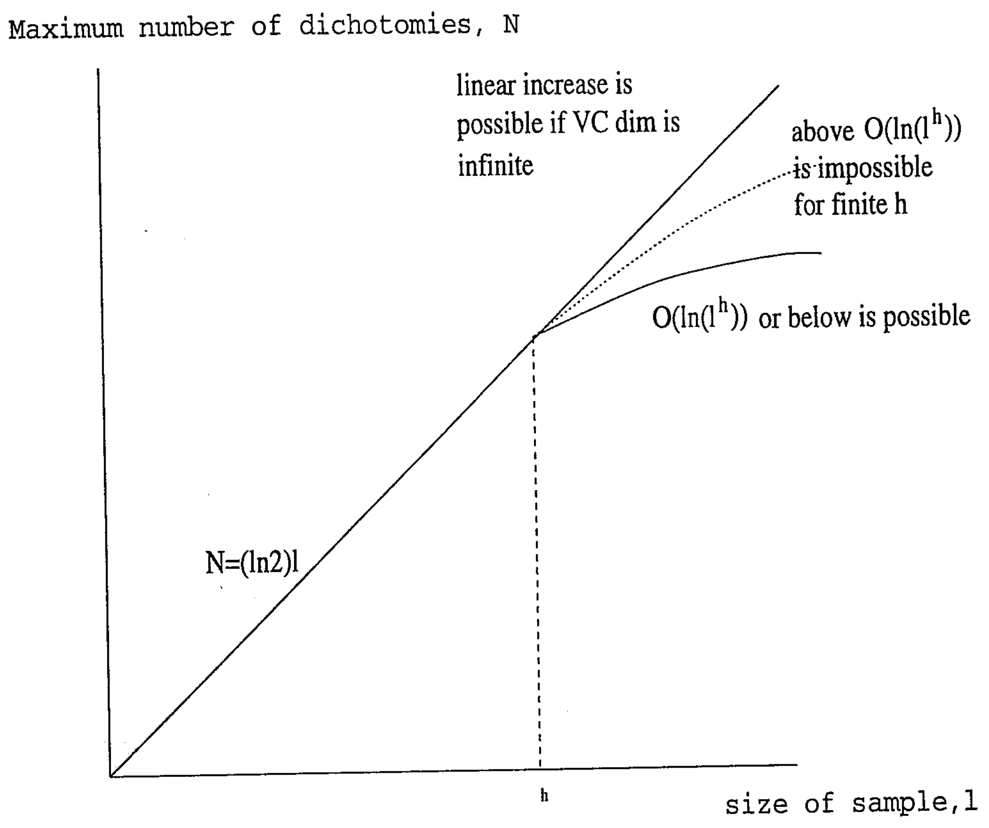


Figure 3.5: Possible (and impossible) forms of the growth function.

3.3.3 Constructive Distribution Independent Bounds

We can now bound the right hand side of expression (3.4) by using the version of inequality (3.8) found in [V&C 68].

$$G^{\Lambda} \leq h(\ln \frac{l}{h} + 1).$$

We can use the same formulae as section (3.3.1) but with the $\mathcal{E}$ defined by

$$\mathcal{E} := 4\frac{h(\ln \frac{l}{h} + 1) - \ln(\frac{\eta}{4})}{l}.$$

If instead of the VC dimension we choose to use the bound based on N, the number of functions in $\mathcal{L}$ we define

$$\mathcal{E} := 2\frac{\ln N - \ln \eta}{l}.$$

Inequalities (3.5) and (3.7) now hold without further modifications.

3.4 Structural Risk Minimisation

From now on we shall only consider machines with finite VC dimensions. An infinite VC dimension implies an inability to learn all unknown distributions no matter how much data is given. Thus they can be said to have no generalisation ability. Thus for a learning machine with a finite VC dimension h the bound given by expression (3.5) can be expressed formally as

$$R(\alpha) \leq R\text{emp}(\alpha) + \Phi(\frac{l}{h}, \eta)$$

with probability $1 - \eta$.

This expression us allows to see precisely what adherence to the ERM principle achieves. It bounds the left-hand side (the expected risk) by the smallest possible value of the empirical risk plus a confidence interval term which is related to the amount of data and the complexity of the machine. (We can neglect the term in η since it is a controlled variable). For large l, or more precisely for large $\frac{l}{h}$, the second term will become small, but for small sample sizes the Φ term can dwarf the other. To obtain the tightest bound on the expected risk we must control both the empirical risk and the confidence interval simultaneously, and that means controlling the VC dimension.

3.4.1 Controlling the VC dimension

A single learning machine consists of a single set of functions with a single VC dimension. To control the VC dimension it is necessary to consider a whole family of learning machines and for this reason the concept of a structure is introduced.

Let S be the set of functions $\{Q(z,\alpha) \mid \alpha \in \Lambda\}$. A **structure** on S consists of a sequence of nested subsets of S,

$$S_1 \subset S_2 \subset S_3 \subset ... \subset S_n ...,$$

where the VCdim, h_k, of each S_k is finite (although the VCdim of S may be infinite, it doesn't matter) and where $\bigcup_{k=1}^{\infty} S_k$ is dense in S.

Types of Structure

A structure can be induced in several ways. Most straightforwardly, some controlling parameter of the learning machine can be varied so that the resulting sequence of machines is of increasing complexity. A canonical example of this is a sequence of multi-layer perceptrons with an increasing number of hidden units (see figure 3.6). Each machine in the sequence contains the previous ones as special cases and the VC dimension increases monotonically.

Other ways of defining a structure included pre-processing the input data, for example with a smoothing algorithm, so that as the data becomes less smooth more dichotomies can be defined on it and hence the VC dimension of the machine on the available inputs increases. A third way is by varying the training algorithm: defining a maximum value for the norm of the weight vector gives one means of defining a structure in this way. Later in the thesis appropriate ways of defining structures over sets of weightless classifiers will be proposed.

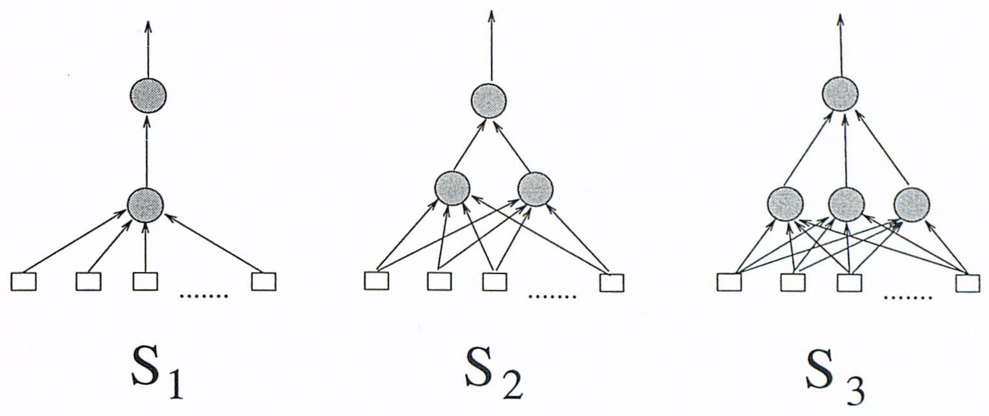


Figure 3.6: A structure on the perceptron based on the number of hidden units.

Minimising Risk with the SRM Principle

We may then sketch how the total risk varies as the ERM principle is applied to each set of functions in structure. Recall that the error bound is such that as the VC dimension of the elements of the structure increase we will expect empirical error to fall, but the confidence term in 3.5 to increase. The resultant graph of expected risk against the VC dimension of the learning machines in the structure is sketched in figure 3.7 where h^* is the VC dimension of the machine with the lowest error bound.

This relationship inspires the statement of the Structural Risk Minimisation principle as follows:

For a given set of training data $\{z_1, z_2, \ldots, z_l\}$ *the function chosen should be that for which the guaranteed risk (as given by 3.5) is minimised.*

To comply with the SRM principle a trade-off between fitting the data and over-complexity of the model is required. We shall return to this theme in chapter 7.

3.4.2 Asymptotic Analysis of the SRM Principle

It has been shown by Vapnik that under certain conditions, the approximations made using the SRM principle will converge asymptotically to the function with minimum expected risk in the whole set. The asymptotic rate of convergence can

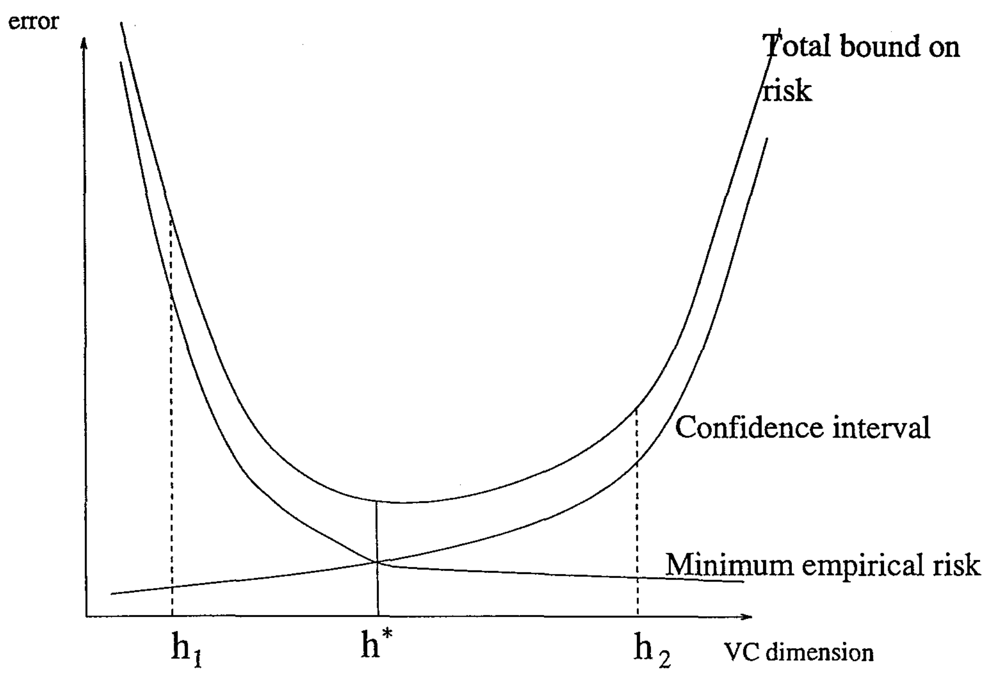


Figure 3.7: The form of the total error for a structure of ERM-based learning machines with increasing VC dimension.

be calculated, but the theory requires for this thesis does not require this. More important is the basic principle: that by adjusting the VC dimension for different sample sizes one may obtain better generalisation error. This will be applied, and extended in the analysis of the n-tuple method.

Chapter 4

The VC Dimensions of Maximum Threshold Discriminators

As described in chapter 3, the VC dimension of a machine is defined in terms of the functions that a learning machine can realise. In this sense it is a capacity measure which is largely independent of the training algorithm. The only dependence it may have is if the training algorithm never yields one of the theoretically possible hypotheses of the learning machine; this effect is considered in section 5.2.3. Hence this chapter will deal with discriminators which are allowed to contain any legal values which may be useful to determine the capacity of the discriminators.

4.1 Principles for Calculating the VC Dimensions

The VC dimension is defined to be the size of the largest set it can shatter. That is, the largest set it can dichotomise in all 2^l ways. The approach to calculating this quantity is two-fold. To find a lower bound for the VC dimension it is simply necessary to demonstrate a shatterable set. If it can be shown that any dichotomy of a set can be realised by a certain set of values in the memory locations of a RAM net then the VC dimension cannot be less than the size of that set. However, to find an upper bound for the VC dimension it is necessary to prove that a set larger than a given number must cannot be shatterable. That is, that at least one dichotomy of the set cannot be realised. For RAM nets which deal with binary inputs this is achieved by means of combinatorical considerations about the structure of training sets of a certain size. It should also be noted that since the systems under consideration contain a finite number of functions, the logarithm of the number of function constitutes an upper bound, (section (3.3.2)). In the cases where an exact upper bound on the VC dimension cannot be found, an upper bound on the logarithm of the number of available hypotheses may be used.

4.2 Types of Discriminator

Discriminators are differentiated from each other by their architecture: the number of RAM nodes in the discriminator, the size of their input addresses, the way in which their inputs are connected to the input and the value of their thresholds, if any. The VC dimension results below are built up from the most simple configurations and inductive arguments are used to extend these to the more complex discriminators. The method of training is also variable, but for the reasons given above has not been taken into account in calculating the VC dimension.

Firstly we need to formalise the system in the learning machine framework. To define a discriminator as a pattern recogniser we must define the circumstances under which its output is **1** and those under which it is **0**. If we limit the output to these two values, the only decision to make is that of assigning a value to the threshold Θ. All discriminators considered in this chapter are of this type.

[Note that if we allow a **u** output to designate a third output value (which we think of as denoting uncertainty as in a PLN) then we change the nature of the problem, since not every setting of the discriminator defines a dichotomy over all data samples. In this case we will only consider a function which takes values in $\{0, 1\}$, on the sample under consideration, as defining a valid dichotomy. We may revise this definition later, but for now it allows direct calculation of many classes of discriminators and n-tuple classifiers.]

In this chapter all the discriminators considered will have $\Theta = N$. This is equivalent to passing the outputs of the RAMs to a N-input AND gate. All RAMs must output 1 to a pattern for the system to output **1**. The relationship between these discriminators, known as **maximum threshold discriminators**, and those with different forms of threshold will be explored in chapter 5.

4.3 Definitions and Notation

The number of nodes in each discriminator is denoted by N while the number of bits sampled by each node, the **tuple size**, will be denoted by n. Discriminators are denoted by script letters (eg. $\mathcal{D}$,$\mathcal{D}_i$) while individual nodes are referred to by subscripted lower case letters (eg. d_i). In fact the $\mathcal{D}$ notation will be somewhat abused: sometimes it denotes the output function of a particular discriminator and sometimes it refers to all discriminators of a certain class. The meaning in any particular case, however, is unambiguous. The **support** of a node is defined to be the set of input nodes from which it receives input. It will sometimes be useful to split the input vector into two parts - the support of one node and the rest. When this is done it is assumed that the vector is ordered so that the support in which we are interested is at the start, delimited from the rest of the vector by a colon. In general the training set under consideration is denoted by T while the projection of the training set onto node d_i is denoted by T_i. Thus T_i is the set of addresses of d_i which are addressed by the training set. We denote the projection (or **component**) of pattern t at node d_i by $t|_{d_i}$.

Further definitions which are useful in the context of this work are those of the functions **fan** and **out** and the sets $\mathbf{X_i}$. Consider T_i for some d_i so we may decompose any input pattern into the part that lies in the support of node i and the rest which we write as $t = t_i : t^-$. For any $S \subset T_i$ we define $fan(S)$ to be the number of elements of T whose i-th projection lies in S. That is

$$fan(S) := |\{t_i : t^- \in T \mid t_i \in S\}|. \tag{4.1}$$

Let us suppose that the supports of the nodes do not overlap. We can then decompose any pattern set T into $T_i : T^-$, where T_i is the component of T at d_i and T^- is the projection of T onto the rest of the input space. Then for any $S \subset T_i$ we may define $out(S)$ to be the set of projections in T^- of any element of T whose projection on d_i lies in S. More formally

$$out(S) := \{t^- \mid t_d : t^- \in T and\, t_d \in S\}. \tag{4.2}$$

Note that crucially $|out(S)| \leq fan(S)$ with equality if and only if every pattern whose projection at d_i lies in S has a different projection on the rest of the space.

Finally, for each node d_i we may define X_i to be the set of memory locations at node d_i which are addressed by more than one training pattern. That is, those elements $x \in T_i$ such that $fan(x) > 1$, hence

$$X_i := \{t_i \in T_i \mid \exists a, b \ \in T \text{ s.t. } a|_{d_i} \text{ and } b|_{d_i} = t_i \ \text{ and } a \neq b\}. \tag{4.3}$$

These definitions become important when we try to prove that training sets above a certain size possess inseparable pairs, the definition of which is given below.

4.3.1 Conditions for Shatterability

We have a definition of a discriminator all of whose values lie in $\{0, 1\}$ and thus we have a consistent definition of shatterability. However, to effectively prove the shatterability or otherwise of a set we introduce the idea of inseparable pairs. Suppose R and S are disjoint subsets of T and $\mathcal{D}$ is the discriminator. We say that R is **inseparable** from S under $\mathcal{D}$ if training every component S to output 1 in $\mathcal{D}$ causes every pattern in R to be recognised, that is, to output **1**. (R, S) is then an **inseparable pair**. For example the pattern $a : b$ is inseparable from $\{a : x, y : b\}$.

Since we are considering maximal threshold discriminators, all components of any trained pattern must output 1 or the pattern will not be recognised. Thus if T

contains an inseparable pair it is not shatterable because no dichotomy in which the two sets take differing values can be realised. We now show the converse, that if T is not shatterable it must contain an inseparable pair.

Since T is not shatterable, by definition there must be some dichotomy which is unrealisable. Pick such a dichotomy and label it (R, U) where R is the set of patterns to be recognised and U is the complement in T. If we train $\mathcal{D}$ to recognise R, we must conclude that it recognises at least one element u of U or else the dichotomy would be realisable. (At this point we should notice that the two constant dichotomies, all $\mathbf{0}$ and all $\mathbf{1}$ can always be realised so $U \neq \emptyset$. Because of this, the case of the constant dichotomies will always be implicitly assumed to have been proven in the following proofs.) Hence $\{u\}$ is inseparable from R. This shows that the existence of an inseparable pair is a necessary and sufficient condition for unshatterability. This fact is used in the proofs below.

4.4 The Simpler Cases

We shall first consider three very simple cases whose analysis will illustrate the methods that can be applies to the more complex configurations. Firstly the case of two discriminators with non-overlapping supports, secondly the case of N discriminators with non-overlapping supports, and thirdly the case of two discriminators whose supports overlap on $l < N$ input points. All three discriminators have $\Theta = N$ so training a pattern requires training all of its components into all the nodes.

4.4.1 Two Disjoint Nodes

Consider the discriminator $\mathcal{D}$ which consists of two nodes, d_1 and d_2, whose supports are disjoint and each consist of n input bits. First we shall demonstrate a pattern set which is shatterable in order to get a lower bound for the VC dimension. Consider the set

$$T := \{0,1\}^n : \mathbf{0} \qquad (4.4)$$
$$\bigcup \mathbf{0} : \{0,1\}^n$$
$$- \mathbf{0} : \mathbf{0},$$

where $\bigcup$ represents the union of sets, $\mathbf{0}$ represents the zero pattern, $\{0,1\}^n$ represents all n-bit binary patterns and where the part of the pattern before the colon represents the support of d_1 and the part after is the support of d_2. (Any subset S of T may be similarly decomposed into $S_1 : S_2$ corresponding to the support of each node.) Note that each pattern is effectively discriminated by only one node: it is this which suggests it may be near-maximal.

We shall now show that this set is shatterable by $\mathcal{D}$. Pick any $S \subset T$ and train $\mathcal{D}$ on S. Any $t \in T$ can be written $t_1 : t_2$ where the two parts of the pattern correspond to the two nodes as before. Pick $t_1 : t_2 \in T - S$. From the definition of T either t_1 or t_2 (but not both) outputs $\mathbf{0}$. Assume without loss of generality that it is t_1. Then we must have $t_2 \neq \mathbf{0}$. If $t_2 \in S_2$ then by the definition of T we must have either $t_2 = \mathbf{0}$ or $\mathbf{0} : t_2 \in S$. Neither of these is possible so $t_2 \notin S_2$. Hence $t_1 : t_2$ is not recognised. Since it was chosen arbitrarily no $t \in T - S$ is recognised. Since S was also chosen arbitrarily the result is true for all $S \subset T$ and hence T is shatterable. Since $|T| = 2^{n+1} - 2$ we have exhibited a shatterable set of size $2^{n+1} - 2$, and hence

$$VCdim(\mathcal{D}) \geq 2^{n+1} - 2. \qquad (4.5)$$

To prove the upper bound we consider any training set, T, such that $|T| > 2^{n+1} - 2$. Let $T_1 : T_2$ be the decomposition of T as before and define X_1 as above. If $|X_1| = 0$ then each component at d_1 is contained in only one pattern of T. Hence T can contain no more than 2^n patterns so for our value of $|T|$ we must have $|X_1| \geq 1$. Now,

$$\begin{aligned} fan(X_1) &= fan(T_1) - fan(T_1 - X_1) \qquad (4.6) \\ &= |T| - (|T_1| - |X_1|) \end{aligned}$$

because for each $t \in T_1 - X_1, fan(t) = 1$.

We know $|T_1| \leq 2^n, |T| \geq 2^{n+1} - 1$ and $|X_1| \geq 1$, and hence

$$\begin{aligned} fan(X_1) &\geq 2^{n+1} - 1 - 2^n + 1 \\ &\geq 2^n. \end{aligned} \tag{4.7}$$

Consider two cases.

Case I: $fan(X_1) > 2^n$

Since there are at most 2^n elements of T_2, by the pigeonhole principle, $\exists\, y_1 \in T_2$ and $x_1, x_2 \in X_1$ such that $x_1 : y_1$ and $x_2 : y_1 \in T$.

Since $x_1 \in X_1, \exists\, y_2 \in T_2$ such that $x_1 : y_2 \in T$ and $y_1 \neq y_2$. Thus $S_1 := \{x_1 : y_2\ ,\ x_2 : y_1\}$ and $S_2 := \{x_1 : y_1\}$ are subsets of T. However, if $\mathcal{D}$ is trained so that S_1 is recognised, then S_2 will also be recognised. Hence S_2 is inseparable from S_1 so T is not shatterable.

Case II: $fan(X_1) = 2^n$

This can only happen when $|T| = 2^{n+1} - 1, |T_1| = 2^n$ and $|X_1| = 1$. Suppose $X_1 = \{x_1\}$.

Because $fan(X_1) = 2^n \leq |T_2| \leq 2^n$ we must have $\forall\, t_2 \in T_2, x_1 : t_2 \in T$. Also, if we pick any $t_1 \in T_1 - X_1$, there must of course be some $t_3 \in T_2$ such that $t_1 : t_3 \in T$. Choose $t_2 \in T_2, t_2 \neq t_3$.

Then the two sets $\{x_1 : t_2, t_1 : t_3\}$ and $\{x_1 : t_3\}$ are inseparable as before.

Thus in either case there are two inseparable subsets of T, so T is not shatterable by $\mathcal{D}$. Hence we have

$$VCdim(\mathcal{D}) < 2^{n+1} - 1. \tag{4.8}$$

Combining the above result with equation (4.5) we get our result.

$$VCdim(\mathcal{D}) = 2^{n+1} - 2. \tag{4.9}$$

4.4.2 N Disjoint Nodes

Let $\mathcal{D}_N$ be a discriminator with N nodes each of support size n.

For a lower bound on the VC dimension of $\mathcal{D}_N$, consider the union

$$\begin{aligned} T &:= \{0,1\}^n : \mathbf{0} : \ldots : \mathbf{0} \\ &\bigcup \ \mathbf{0} : \{0,1\}^n : \mathbf{0} : \ldots : \mathbf{0} \\ &\bigcup \ \ldots \\ &\bigcup \ \mathbf{0} : \ldots : \mathbf{0} : \{0,1\}^n \\ &- \ \mathbf{0} : \mathbf{0} : \ldots : \mathbf{0} : \mathbf{0}. \end{aligned} \tag{4.10}$$

To see that T is shatterable, pick any $R \subset T$ and train R to be recognised in $\mathcal{D}_N$. Consider any $s \notin R$. The non-zero component of s does not appear in any other pattern in T, so s will not be recognised unless it is directly trained. Since s was arbitrary, no pattern not in R will be recognised. Hence $T \backslash R$ is not recognised so the dichotomy is realised. Since R itself was arbitrarily chosen, $\mathcal{D}_N$ can realise any dichotomy of T so T is shatterable. Since $|T| = N.2^n - N$ we have

$$VCdim(\mathcal{D}_N) \geq N.2^n - N. \tag{4.11}$$

Suppose T is a training set such that $|T| > N.2^n - N$. As our inductive hypothesis we assume that for $1 < k < N$, $VCdim(\mathcal{D}_k) = k.2^n - k$. By result (4.9) this is certainly true for the case $N = 3$.

Decompose T as $T_d : T^-$ where T_d corresponds to the support of a single node, d, and T^- is the input to the $N-1$ node discriminator $\mathcal{D}^-$ obtained by removing d from $\mathcal{D}_N$. We want to use the inductive hypothesis to get an upper bound on $|T^-|$.

As before, in (4.2), for any $S \subseteq T_d$ define

$$out(S) := \{t^- \mid t_d : t^- \in T and\ t_d \in S\}. \tag{4.12}$$

Thus $fan(S) \geq |\vec{out(S)}|$ with equality if and only if no element of $out(S)$ is generated by more than one element of T. That is, if there are no $t, s \in S$ such that $t : t^-$ and $s : t^- \in T$. Define X_d as before. Note that if $|X_d| = 0$ then there would be a maximum of 2^n patterns in T since each pattern would have a unique projection at d. Since this is not the case we must have $|X_d| \geq 1$.
As before,

$$\begin{aligned} fan(X_d) &= fan(T_d) - fan(T_d - X_d) \\ &= |T| - (|T_d| - |X_d|). \end{aligned} \tag{4.13}$$

Also, $|T| > N.2^n - N, |T_d| \leq 2^n$ and $|X_d| \geq 1$, so

$$\begin{aligned} fan(X_d) &> N.2^n - N - 2^n + 1 \\ &= (N-1)2^n - (N-1). \end{aligned} \tag{4.14}$$

Since $fan(X_d) \geq |out(X_d)|$ we have two cases.

Case I: $fan(X_d) > |out(X_d)|$

In this case there must exist some $t, s \in X_d$ such that $t : t^-, s : t^- \in T$. Because $t \in X_d$ there is also some $t' \in T^-$ such that $t : t' \in T$. Hence we cannot separate $\{t : t^-\}$ from $\{s : t^-, t : t'\}$ so T is inseparable in this case.

Case II: $fan(X_d) = |out(X_d)|$

In this case $|out(X_d)| > (N-1)(2^n - 1)$ so by the inductive hypothesis, $out(X_d)$ is not shatterable by $\mathcal{D}^-$ Hence there are two disjoint sets $S_1, S_2 \subseteq out(X_d)$ which are inseparable by $\mathcal{D}^-$. That is, if S_1 is recognised so is S_2. Pick any $s_2 \in S_2$. Define S_1^+ to be the pre-image of S_1 in the whole of T. That is, S_1^+ contains all training patterns which project an element of S_1. Pick $x \in X_N$ such that $x : s_2 \in T$. By the definition of $X_N, \exists\, t^- \in T^-$ such that $x : t^- \in T$ and $t^- \neq s_2$.

Then $\{x : s_2\}$ is inseparable from $S_1^+ \bigcup \{x : t^-\}$.

Thus T is not shattered in either case, so

$$VCdim(\mathcal{D}_N) \leq N.2^n - N. \tag{4.15}$$

Putting this together with equation (4.11) we get

$$VCdim(\mathcal{D}_N) = N.2^n - N. \tag{4.16}$$

4.4.3 Two Nodes Overlapping by l

Let $\mathcal{D}_l$ be a discriminator consisting of two nodes whose supports overlap by l pixels where $l < n$.

For a lower bound consider

$$\begin{aligned} T &:= \{0,1\}^{n-l} : \{0,1\}^l : \mathbf{0} \\ &\bigcup \ \mathbf{0} : \{0,1\}^l : \{0,1\}^{n-l} \\ &- \ \mathbf{0} : \{0,1\}^l : \mathbf{0}. \end{aligned} \tag{4.17}$$

Then T has size $2^{n+1} - 2^{l+1}$ and can be shattered by $\mathcal{D}_l$ in the same way as the disjoint case. Hence

$$VCdim(\mathcal{D}_l) \geq 2^{n+1} - 2^{l+1} \tag{4.18}$$

Suppose T is training set such that $|T| \geq 2^{n+1} - 2^{l+1} + 1$. Then we must have

$$\frac{|T|}{2^l} > 2^{n-l+1} - 2 \tag{4.19}$$

We can partition T into 2^l disjoint subsets (some may be empty) according to the l-bit pattern in the overlap. From inequality (4.19) we see that one such partition must contain more than $2^{n-l+1} - 2$ patterns. Decompose this partition as $T_1 : u : T_2$, where u is the (fixed) pattern in the overlap, T_1 is the projection of the partition onto the part of the input which is only in the support of node d_1 and T_2 is the same but for d_2.

Hence by result (4.9), $T_1 : u : T_2$ cannot be shattered by a discriminator consisting of two non-overlapping nodes of size $n - l$ on T_1 and T_2. This means that some $R_1 : R_2$ and $S_1 : S_2 \subseteq T_1 : T_2$ are not separable by the two disjoint nodes. Therefore $R_1 : u : R_2$ and $S_1 : u : S_2$ are not separable by $\mathcal{D}_l$. Thus T cannot be shattered by $\mathcal{D}_l$. Hence

$$VCdim(\mathcal{D}_l) \leq 2^{n+1} - 2^{l+1} \tag{4.20}$$

Together with equation (4.18) this gives us the result

$$VCdim(\mathcal{D}_l) = 2^{n+1} - 2^{l+1}. \tag{4.21}$$

4.4.4 More General Configurations

In sections 4.4.1, 4.4.2 and 4.4.3 the most basic arrangements of nodes were considered. A glance at the VC dimension results obtained suggests that adding a node with n inputs adds 2^n to the VC dimension, but that for each overlap of size l between the supports of nodes the VC dimension falls by 2^l. Thus we may be led to speculate that for a general discriminator $\mathcal{D}$ which comprises N numbered nodes

$$VCdim(\mathcal{D}) = \sum_{x=1}^{N}(2^n - \sum_{i=x+1}^{N} 2^{l(i,x)}),$$

where $l(i, x)$ is the size of the overlap of the inputs to nodes d_i and d_x. For some highly connected configurations (for example $n = N - 1$), the right hand side of the above is negative so this equality cannot always hold. However the above expression is close to a lower bound for the VC dimension of a discriminator. It will be shown that for any discriminator a shatterable set whose size is bounded below by a similar (though slightly larger) expression can be produced.

4.4.5 Constructing the Bounds on VC Dimension

Suppose $\mathcal{D}$ is a discriminator. Its parameters are arbitrary except that we demand $\Theta = N$. We now distill the wisdom gained the from the previous calculations and derive some general bounds for the VC dimension of such a discriminator.

A Lower Bound

Consider a set of patterns, T, built up in the following manner. For each node, d_x, define T_x to be the set of all patterns which are $\mathbf{0}$ everywhere except possibly on the support of d_x (at this point $|T_x| = 2^n$) and let T be the union of the T_x. Now, for every pair of nodes d_x, d_y, if any $t \in T_x \cap T_y$ $(x \neq y)$, then remove $t : 0$ from the set. This removes at most $\sum_{y=x+1}^{N}(2^{l(y,x)} - 1)$ patterns from each T_x where $l(x,y)$ is size of the overlap of the support of nodes d_y and d_x. (The '-1' is because the zero pattern doesn't need to be removed yet.) To calculate the (disjoint) union of the remaining sets we must take care not to remove patterns more than once, so we only need to consider each pair of node (x,y) once. Hence this union has size greater than or equal to

$$\sum_{x=1}^{N}(2^n - \sum_{y=x+1}^{N}(2^{l(y,x)} - 1)) - N. \tag{4.22}$$

(This time the '-N' has to be put in at the end because the zero pattern must finally be removed.)
If $l(y,x)$ is equal for all x and y this simplifies to

$$\frac{N}{2}(2^{n+1} - (N-1)(2^l - 1)) - N \tag{4.23}$$

Since each $t \in T$ has at least one projection not in any X_i (all the overlap patterns having been removed) it contains no inseparable pairs. Hence T is shatterable. However, the true size of the set generated by this method will in general be greater because we are counting some patterns more than once in the subtraction phase. The exact number of patterns removed depends on the exact topology, so to tighten up the bound we need to make more assumptions about the overlaps of the node inputs. In particular, the above is certainly not guaranteed to be a maximal shatterable set.

An Upper Bound

Recall that the set X_i defined for a pattern set T and pertaining to each node d_i was defined to be the set of components in the support of d_i which were projected by more than one pattern in T. If the projection of some training pattern t onto each node is in each node's X_i then t will be inseparable from the rest of the training set. Moreover any pattern for which this is not the case will be separable from the rest of the set. This gives us a necessary and sufficient condition for a training set to contain an inseparable pair and hence to be shatterable. *It must contain no pattern whose projections all lie in the X_i.*

Suppose we know the sets T_i and X_i for each node i. How many separable patterns can we construct under these constraints? The optimal solution (where possible) is to take $N-1$ elements from the X_is and exactly one from some $T_{i_0} - X_{i_0}$. Now by definition we can only select each element of each $T_i - X_i$ once, so this gives us a maximum of

$$\sum(|T_i| - |X_i|) \tag{4.24}$$

separable patterns. This upper bound may not be attained if it is not possible to select a set such that each pattern contains exactly one element not in any X_i. However, if we can find a set for which we know the sizes of the T_i are optimal and the sizes of the X_i are minimal, then the bound in (4.24) will be useful.

Thin Uniform Coverage: A Special Case

If we have a large input space, X, and relatively small n-tuples, it is often legitimate to assume (or to contrive) that every point in the input space is in the support of some nodes d_i, d_j etc., but that no other input point lies in the same two supports. From the point of view of the nodes we can say that no two supports will overlap on more than one input pixel. We shall call this type of input sampling **thin uniform coverage.** In the most uniform case every input bit is sampled by $s := \frac{nN}{|X|}$ nodes.

Thus the value of $l(i,x)$ is always 1 or 0 so the lower bound construction and 4.23 yields a shatterable set of size greater than

$$\begin{aligned} &\frac{N}{2}(2^{n+1} - (N-1)(2^l - 1)) - N \\ =&\frac{N}{2}(2^{n+1} - (N+1)). \end{aligned} \tag{4.25}$$

In fact we can show that the size of the set generated by the algorithm in section 4.4.5 above is $N(2^n - (n+1))$ and that this is also the VC dimension of the discriminator. Let $T^+ := \cup_{d_i \in \mathcal{D}} \mathbf{0} : \{0,1\}^n : \mathbf{0}$ where the non-zero part of the pattern corresponds to the input to a particular node and the union is taken over all the nodes. Since any two nodes overlap at exactly one point, the training pattern which is $\mathbf{1}$ there and $\mathbf{0}$ elsewhere must be removed. Thus we must remove n patterns from each of the N components of the formula for T^+. This is Nn patterns in all. We must also allow for the fact that the zero pattern has been counted N times and must be removed. Thus for our final set T we must have

$$\begin{aligned} |T| &= N.2^n - Nn - N \\ &= N(2^n - (n+1)) \end{aligned} \tag{4.26}$$

as stated.

If we consider the case above where all $|T_i| = 2^n$ we see that each X_i consists exactly of the zero pattern and all patterns containing exactly one $\mathbf{1}$. Hence for all i we have $|X_i| = n + 1$. Thus the upper bound on the VC dimension is $\sum(2^n - (n+1)) = N(2^n - (n+1))$ which is the value given by our constructive lower bound algorithm.

Can we find a larger shatterable set by reducing the size of the X_i, perhaps reducing the size of the T_i to do so? We cannot, because each of the patterns in the X_i is generated by (on average) $s2^{n-2}$ training patterns. Thus to decrease $|X_i|$ by one, it would be necessary to remove a far greater number of patterns. However much we reduce T_i we can never remove fewer training patterns than the decrease in $|X_i|$.

Thus $|T_i| = 2^n$ is the optimal value for each i and because of the overlaps between the node supports, $n + 1$ is the minimal size of each X_i. Hence in the case of thin uniform coverage we have the following:

$$VCdim = N(2^n - (n+1)). \tag{4.27}$$

4.5 Auto-Associative GNUs

The GNU of Aleksander which is described in 2.4.5 can be configured and trained as an auto-associator. A GNU consists of the same sort of nodes as a discriminator, but some of the nodes take input from the output of other nodes. (Note that we shall assume that the nodes output either 0 or 1. There is no probabilistic output in this analysis.) The whole system is clocked and recall takes place over a number of time steps. The state of the GNU at a particular time is taken to be the pattern given by the outputs of all the nodes treated as an ordered sequence. There is no thresholding. If there are the same number of nodes as bits in the input space then a pattern can be associated with an internal state. This allows auto-association to take place.

Suppose the GNU takes all its inputs from the outputs of the other nodes (randomly sampled as before) except for when $t = 0$ when the nodes have yet to give an output and the internal state may be set as wished. The GNU is then said to be autonomous and a pattern is identified with the internal state to which it is identical. A state or pattern p is said to be **stable** if a GNU in state p remains at p at the next time step. (And hence forever if we have the time to wait.) If a particular pattern is a stable state of a trained GNU we can say it has been recognised. If the pattern is not stable it is not recognised. Since probabilistic outputs have not been allowed in the nodes this definition is consistent and we can therefore define the VC dimension of a GNU.

4.5.1 Conditions for Shatterability

In the case of discriminators with $\Theta = N$ any pattern that is trained is recognised. This is not true of GNUs because of the existence of contradictions in pattern sets. A contradiction in a pattern set at a particular node d_i occurs when two patterns p and q which differ at the bit represented by that node nevertheless have the same projection at that node. In this case whatever is stored in the node at that address it is not possible to simultaneously recognise or reject p and q. They will always take opposite values. Discussions of contradictions in auto-associative GNUs can be found in [14] and [1].

The failure of a set to be shatterable by a maximal discriminator was solely due to the existence of patterns in the set which were inseparable from some other subset of the training set. In the case of the GNU we require both that the training set be inseparable but also that it be non-contradictory. It is claimed that these two conditions are necessary and sufficient for a set to be shatterable by a GNU. Notice that to make a state s stable all locations addressed by the components of s must generate the correct output 1 or 0 depending on the node's assigned position in the internal state. This is equivalent to the $\Theta = N$ discriminator because in both cases every location of s must be trained (in the former case to be 1, in the latter to be the appropriate bit of the pattern). Hence a pair inseparable by such a discriminator is also inseparable by the GNU and vice versa.

For the necessary part of the proof, suppose patterns p and q are contradictory at some node then clearly no dichotomy in which they are both recognised can be realised. Suppose, instead, that pattern p is inseparable from $Q \subset T$. Then no dichotomy in which all members of Q are recognised while p is not can be realised. Hence the conditions are necessary for shatterability.

Now consider a training set T which satisfies these two conditions. Pick any $S \subset T$ and train $\mathcal{G}$ to recognise S. Since T contains no contradictions this can be done for any such S. Now pick any $t \in T$. Is it a stable state? If so then it is inseparable from S both in the sense of a GNU (by definition) and in the sense of the $N = \Theta$ discriminator (as explained above). However we are assuming this is not the case and hence t is not recognised. Hence $\mathcal{G}$ can realise this dichotomy. Since t and S were chosen arbitrarily we have shown that $\mathcal{G}$ can realise any dichotomy of T. Thus we have shown that the conditions are sufficient.

4.5.2 Calculating the VC Dimension

To apply the results above we shall suppose that the GNU in question is sparsely connected, in which case we can assume or contrive that its connection graph satisfies the constraints of thin uniform coverage, see section 4.4.5. Separability was a necessary and sufficient condition on T for shatterability by a discriminator. Hence the VC dimension result in (4.27) is an upper bound for the VC dimension of GNUs. The question is how close is it possible to go to these bounds? In fact, for any n and N we can reach the upper bounds simply by insisting on self-connection. That is, that every GRAM in the GNU receive its own output as input at the next time step. As pointed out by Braga in [4], a node which is connected to itself can never generate a contradiction during auto-association because the required output forms part of each node's input. Hence the sets constructed in sections 4.4.5 and 4.4.5 are maximal shatterable set for such GNUs.

Are such self-connected nets useful? Certainly they are perfect auto-associators in that any trained pattern is recognised. The problems come in the form of false attractors and basins of attraction and the fact that GNUs are usually used for more complex, hetero-associative tasks. These, however, do not currently fall within the scope of the VC dimension measurement.

Chapter 5

VC Results for Other Discriminators and n-Tuple Classifiers

The VC dimension calculations in chapter 4 pertain only to discriminators with maximum thresholds, that is, where $\Theta = N$. What was changed was simply the connection graph, the way in which the input space is sampled by the nodes. In this chapter we shall extend the results to discriminators whose output is a different function of the node outputs. Then we shall turn our attention to the n-tuple classifier itself and bound its VC dimension. As by-products of the VC dimension calculations various theorems are proven about the number of hypotheses these systems can generate.

5.1 Which Other Discriminators?

The most obvious case which has not yet been considered is that of the discriminator whose threshold is less than N. Such discriminators are known as **sub-maximal**. The other variation considered is the **thick threshold** discriminator. In this variation both an upper and a lower threshold are specified. If the sum of the node outputs is below the lower threshold the pattern is not recognised, if it is above the upper the pattern *is* recognised. If it is between the two the discriminator takes an uncertain value, denoted by **u**. The purpose of this discriminator is to draw lessons about how to deal with the uncertain outputs which occur in more complex systems such as some variations on the n-tuple classifier.

5.2 Sub-Maximal Threshold Discriminators

If we set the threshold to a value $\Theta < N$, recognition performance is affected in two ways compared with the maximal threshold value. Firstly, for a given set of node values more patterns will be recognised since fewer components of each pattern must be trained for the pattern to be recognised. Secondly when we train a pattern we only *need* to train Θ of its components. Indeed, if we are to minimise false recognition of patterns, we should never train more than Θ components of each training pattern to be recognised. We shall show that these effects almost cancel each other out and the VC-dimension is almost the same for all values of Θ.

5.2.1 A Constructive Lower Bound

Suppose $1 < \Theta < N$. Let $\mathcal{D}_\Theta$ be a discriminator with threshold Θ and let $\mathbf{1}$ and $\mathbf{0}$ represent the patterns containing n 1s and 0s respectively. Consider the set

$$\begin{aligned} T \;:=\; & \{0,1\}^n : \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \mathbf{1} \\ \bigcup\; & \mathbf{1} : \{0,1\}^n : \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \mathbf{1} \\ \bigcup\; & \ldots \\ \bigcup\; & \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \{0,1\}^n : \ldots : \mathbf{0} \\ -\; & \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \mathbf{1} \\ -\; & \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \mathbf{0}. \end{aligned} \tag{5.1}$$

In each pattern $\Theta - 1$ of the components are $\mathbf{1}$ and $N - \Theta$ are $\mathbf{0}$. The other component has a unique value. None of the patterns which only contain one $\mathbf{1}$ and $\mathbf{0}$ components are in T. The discriminator can be made to recognise a pattern t by training all the non-$\mathbf{0}$ components in the relevant RAMs. Because there are exactly

Θ non-**0** components in each pattern, this is sufficient to guarantee that the pattern so trained is recognised.

Using this training method we can show that the set T is shatterable by $\mathcal{D}_\Theta$. Pick any subset of R and train it as described above. It is claimed that this ensures that no pattern in $T - R$ is recognised. Pick $s \in T - R$. All the $(N - \Theta)$ **0** components will yield a 0 output, so for s to be recognised all of the other components must output 1. However, since $s \notin R$ the "distinctive" component of s - the one which is neither **0** nor **1** - has not been trained. Thus at least $N - \Theta + 1$ nodes output 0 to s so at most $\Theta - 1$ output 1. Hence s is not recognised. Since s was arbitrary, no element of $T - R$ is recognised, and since R was arbitrary any dichotomy can be realised in this manner. Thus we have shown that T is shatterable.
Since $|T| = N(2^n - 1)$ we have the result

$$VCdim(\mathcal{D}_\Theta) \geq N(2^n - 1). \tag{5.2}$$

5.2.2 An Upper Bound

In section 3.3.2 it was shown that the logarithm base 2 of the number of hypotheses a learning machine can generate is an upper bound for the VC dimension of the machine. In the case of any discriminator with a single (that is, not thick) threshold, the number of realisable functions is bounded above by the number of different ways of filling all the addressed locations in all the nodes. There are $N2^n$ such locations and since the stored values can be either 0 or 1 there are 2^{N2^n} ways of filling them. Each such set of stored values is referred to as a **node value set**. This gives an upper bound on the VC dimension of all discriminators of

$$\log_2(2^{N2^n}) = N2^n.$$

So the bounds we have for the VC dimension are

$$N(2^n - 1) \leq VCdim(\mathcal{D}_\Theta) \leq N2^n. \tag{5.3}$$

To reduce the value of the upper bound we may consider how many node value sets yield the same hypothesis on the input space. For instance, if $1 < \Theta < N$ then the node value set in which all locations are filled with 0s will give the same output on any input pattern as that which contains all 0s except for a single 1. We shall now show that two node value sets generate different hypotheses on the input space unless they both yield the same constant hypothesis (ie. all $\mathbf{1}$ or all $\mathbf{0}$). We use this result to calculate the number of distinct hypotheses available and thus see if the upper bound on the VC dimension can be tightened up.

Distinct Hypotheses from Distinct Node Value Sets

To prove the results we will consider what conditions on two node value sets NV_1 and NV_2 cause the discriminators containing them, $\mathcal{D}_1$ and $\mathcal{D}_2$, to generate the same hypothesis on the whole input space. To show that the hypotheses generated are different we need only produce one pattern which takes a different value under $\mathcal{D}_1$ than $\mathcal{D}_2$.

The proposition is as follows:

Two different node value sets give rise to different non-overlapping discriminators if and only if they do not both generate the either all $\mathbf{1}$ *nor the all* $\mathbf{0}$ *hypothesis.*

The proof is by induction. First consider the case $\Theta = N$. Let NV_1 and NV_2 be different node value sets and suppose that the discriminators they generate when thresholded at N, $\mathcal{D}_1$ and $\mathcal{D}_2$, are the same. Since the two node value sets are different, some component y_i in some node d_i must take different values in each, and since the discriminators are not constantly $\mathbf{0}$ there is some pattern p such that $\mathcal{D}_1(p) = \mathcal{D}_2(p) = \mathbf{1}$. Let p_y be the pattern formed by replacing the component of p at node d_i by y_i. Then we must have $\mathcal{D}_1(p_y) \neq \mathcal{D}_2(p_y)$ which is a contradiction. Hence one of the assumptions must be false so the result is true for the case $\Theta = N$. This gives us the base case for our induction, namely when $N = \Theta = 1$.

Assume that for all $k < N$ and for all $1 \leq \Theta \leq k$ the proposition is true. Consider two differing node value sets NV_1 and NV_2 defined on N nodes and let the discriminators they generate when thresholded at $\Theta < N$ be $\mathcal{D}_1$ and $\mathcal{D}_2$. Pick any node and label it d_N. Consider the $N-1$ node discriminators generated by removing d_N and thresholding the raw output at Θ. Call them $\mathcal{D}_1^-$ and $\mathcal{D}_2^-$. Let $\mathcal{O}(p)$ be the raw integer output from a discriminator given input p. Let us distinguish three cases.

<u>Case 1</u> $\mathcal{D}_1^- = \mathcal{D}_2^- \equiv \mathbf{1}$

In this case the raw outputs from each smaller discriminator are already greater than Θ for all patterns. Hence the output when d_N is added will still be greater than Θ so $\mathcal{D}_1 \equiv \mathcal{D}_2 \equiv \mathbf{1}$.

<u>Case 2</u> $\mathcal{D}_1^- = \mathcal{D}_2^- \equiv \mathbf{0}$

Let us assume that $\mathcal{D}_1 = \mathcal{D}_2$, so for all patterns p in the input space of $\mathcal{D}_i^-$ and for all x in the input space of d_N, $\mathcal{D}_1(p:x) = \mathcal{D}_2(p:x)$. Suppose also that $\mathcal{D}_1 \not\equiv \mathbf{0} \not\equiv \mathcal{D}_2$. Suppose there is some pattern x in the input to d_N at which NV_1 and NV_2 take differing values. Without losing any generality we may assume that NV_1 outputs 1 and NV_2 outputs 0. Then since $\mathcal{D}_2^-(p) = 0$ it is true that $\mathcal{D}_2(p:x) = 0$ and since we are assuming that the discriminators both perform the same function, $\mathcal{D}_1(p:x) = 0$. Thus, since $d_N(x) = 1, \mathcal{O}_1(p) < \Theta - 1$ for all p. Thus $\mathcal{O}_1(p:x) < \Theta$ for all p and x. Hence $\mathcal{D}_1 \equiv \mathbf{0}$ so we must assume that no such x exists. So the node values in NV_1 and NV_2 must be the same as each other at node d_N for all values of x.

Since $\mathcal{D}_1 \not\equiv \mathbf{0} \not\equiv \mathcal{D}_2$ we may choose $p:x$ such that $\mathcal{D}_1(p:x) = \mathcal{D}_2(p:x) = 1$. This can only happen when both NV_1 and NV_2 store 1 at address x and $\mathcal{O}_1^-(p) = \mathcal{O}_2^-(p) = \Theta - 1$. Now, NV_1 and NV_2 do not differ on d_N so they must differ on some other node d_i. Let y be the pattern on which they differ and consider p_y, the pattern p with y inserted at the support of d_i. Since the output of the $\mathcal{D}_j^-$ is $\mathbf{0}$ we must have either $\mathcal{O}_1^-(p_y) = \Theta - 2$ and $\mathcal{O}_2^-(p_y) = \Theta - 1$ or vice versa. Either way,

$\mathcal{D}_1(p_y : x) \neq \mathcal{D}_2(p_y : x)$ which is a contradiction. Hence in this case $NV_1 = NV_2$.

Case 3 $\mathcal{D}_1^- \neq \mathcal{D}_2^-$

Suppose $\mathcal{D}_1 = \mathcal{D}_2$, then there is some p such that $\mathcal{D}_1^-(p) \neq \mathcal{D}_2^-(p)$, and for all x in the input to d_N, $\mathcal{D}_1(p : x) = \mathcal{D}_2(p : x)$. Thus NV_1 and NV_2 must differ at d_N for each value of x. Moreover they must differ in the same way so as to ensure $\mathcal{D}_1(p : x) = \mathcal{D}_2(p : x)$. Thus the node values at d_N in both NV_1 and NV_2 must differ and not vary with x.

Now consider each choice of d_N in turn, if any yield Cases 1 or 2 then the result is proven. If not, then all the nodes contain constant values which are opposite each to each other so they certainly do not yield the same hypothesis. Thus for $\Theta < N$ the proposition is true. Since the result is true for the case $N = 1$ it is true for all N by mathematical induction.

Moreover we have proved above that we know the result is true for all $\Theta = N$, this proves the result for all N and all $1 \leq \Theta \leq N$.

Counting Distinct Hypotheses

The result above tells us that distinct node value sets generate distinct hypotheses as long as they do not generate a constant function. To count the distinct hypotheses we must therefore count the number of ways of generating the constant hypotheses. To do this we note the following

A NV set generates constant **0** if fewer than Θ nodes contain a 1.
A NV set generates constant **1** if fewer than $N + 1 - \Theta$ nodes contain a 0.

To write down the number of NV sets generating the constant hypothesis we must consider the number of NV sets with exactly k nodes without a 1 for each $N - \Theta < k \leq N$. This number is given by $\binom{N}{k}2^{k(2^n-1)}$. To obtain the total number of hypotheses we must sum this expression for $N - \Theta \leq k \leq N$. Hence the number of node value sets which yield the **0** hypothesis is

$$\binom{N}{N-\Theta+1}2^{(N-\Theta+1)(2^n-1)} + \ldots \binom{N}{N-\Theta+k}2^{(N-\Theta+k)(2^n-1)} + \ldots \binom{N}{N}2^{N(2^n-1)}.$$

The equivalent expression for node value sets which realise the **1** hypothesis is

$$\binom{N}{\Theta}2^{\Theta(2^n-1)} + \ldots \binom{N}{\Theta+k}2^{(\Theta+k)(2^n-1)} + \ldots \binom{N}{N}2^{N(2^n-1)}.$$

These expressions are not immediately amenable to analysis, but all we need to know is that such duplications of functionality by differing node value sets is not so great as to make the number of node value sets a poor estimate for the number of hypotheses. Numerical simulations were carried out in *Mathematica* for which the quantitative results are listed in Appendix A. The important qualitative results however are the following:

1. The number of node value sets generating constant hypotheses is a maximum for $\Theta = N$ and $\Theta = 0$. It is minimal for $\Theta = \frac{N}{2}$.

2. For $n << N$ the proportion of constant hypotheses is negligible compared to the number of non-constant ones.

The Consequences for VC Dimension

We see from the results above that when $n << N$, we can almost equate the number of node value sets with the number of hypotheses, and so we cannot appreciably tighten up the upper bound on the number of hypotheses. Hence we cannot use this method to obtain a tighter bound on VC dimension either. However this is not too important for the discriminator as the bound is fairly tight already. We shall return to the problem of multiply generated hypotheses when considering the n-tuple classifier in section 5.4.2 when the upper bound obtained from counting node-value sets is much higher than the maximum lower bound achieved and we again need a bound based on counting hypotheses.

5.2.3 Impossible Node Value Sets

It was noted at the very beginning of 4 that the training algorithm only affects the VC dimension of a classifier if it renders some of the hypotheses of the learning machine unrealiseable. The standard training algorithm of Bledsoe and Browning does indeed make certain node value sets of the n-tuple classifier unobtainable. Training with only one pattern will ensure that all the RAMs in one discriminator store at least one **1**. Indeed if any RAM in discriminator contains a **1** at some address then all the other RAMs in the same discriminator but also contain at least one . This does remove a number of node value sets, and since not all those sets yield constant hypotheses, a number of different hypotheses. However, the proportion of hypotheses lost is tiny and the effect on the VC dimension estimate is equally small.

5.3 Thick Thresholds

Recall that **thick threshold discriminator**, $\mathcal{D}_\Theta$ is defined as a normal discriminator, but with an upper and a lower threshold Θ_u and Θ_l. A pattern is recognised by $\mathcal{D}_\Theta$ if and only if more than Θ_u RAMs fire. A pattern is unrecognised if fewer than Θ_l RAMs fire. If the number of RAMs is between Θ_l and Θ_u then the output is probabilistic. For a set to be dichotomised by $\mathcal{D}_\Theta$ we make the definition that there must be no probabilistic outputs on that set. (Note that this is an ad hoc definition to keep the thick threshold discriminator within the same framework as the other discriminators so far considered.)

Because this discriminator is a generalisation of the standard discriminator, the results are only proven for the cases $\Theta_u = N$ and/or $\Theta_l = 1$. The generalisations of the bounds for non-maximal discriminators proceed in a similar way, but are not presented here since the thick threshold discriminator is mainly a qualitative system designed to investigate the effects of probabilistic outputs.

5.3.1 A Lower Bound on Thick Threshold VC-dimension

Again we provide a constructive lower bound. The training set has a similar form to that for "thin" discriminators except that each training pattern contains $\Theta_u - \Theta_l + 1$ non-**1** and non-**0** components instead of just one. It is a generalisation of equation(5.2).

Let **a** be a fixed non-**1** and non-**0** pattern component, say the alternating pattern. Then define T as follows.

$$\begin{aligned} T &:= \{0,1\}^n : \mathbf{a} : \ldots : \mathbf{a} : \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \mathbf{1} \qquad (5.4) \\ &\cup\ \mathbf{1} : \ldots : \mathbf{1} : \{0,1\}^n : \mathbf{a} : \ldots : \mathbf{a} : \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \mathbf{1} \\ &\cup\ \ldots \\ &\cup\ \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \{0,1\}^n : \mathbf{a} : \ldots : \mathbf{a} : \ldots : \mathbf{0} \\ &-\ \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \mathbf{1} \\ &-\ \mathbf{1} : \ldots : \mathbf{1} : \mathbf{0} : \ldots : \mathbf{0}. \end{aligned}$$

Each pattern must contain $\Theta_l - 1$ **0** components, $\Theta_u - \Theta_l - 1$ **1** components and $\Theta_u - \Theta_l + 1$ non-**1** and non-**0** components. Any dichotomy of the set T can be realised by training all the **1** components and none of the **0** components to output 1 while also training the non-**1** and non-**0** components of the patterns which are recognised in the dichotomy. The non-**1** and non-**0** components of the patterns not to be recognised are not trained. Since any dichotomy of T is realisable in this way, T is shatterable by the thick threshold discriminator.

None of the terms in the union that forms T can have overlapping non-**1** and non-**0** components, so there are at most $\frac{N}{\Theta_u - \Theta_l + 1}$ terms in the union. Hence the size of the set T is

$$\left\lfloor \frac{N}{\Theta_u - \Theta_l + 1}(2^n - 1) \right\rfloor,$$

which we may conclude is a lower bound for the VC dimension.

5.3.2 An Upper Bound on Thick Threshold VC-dimension

If the upper threshold is equal to N, then we can use a similar argument to that in section 4.4.5 to state that if the sets T_i (ie. the projection of the training set onto each node's support) are given, and if $\Theta_u = N$ (or conversely $\Theta_l = 1$) then no element of T can have more than $N - \Theta_l + 1$ (conversely Θ_u) components belonging to the sets X_i. Hence for given T_i and $\Theta_u = N$ we have

$$VCdim \leq \frac{\Sigma(|T_i| - |X_i|)}{N - \Theta_l + 1}. \tag{5.5}$$

This also holds for the original case in which $\Theta_u = \Theta_l$ and so is a generalisation of the original result.
However, the exact result for general Θ_u is a generalisation of the result for the sub-maximal discriminator and so is still not known.

5.4 The VC Dimension of the n-Tuple Classifier

5.4.1 Defining the Learning Machine

The n-tuple classifier was defined in section 2.4.1, but here we ensure that the definition is precise enough to be used to calculate VC dimension. The output of the classifier must lie in $\{0, 1\}$ since we are only considering classifiers with binary output, and for a two-discriminator classifier the output is simply taken to be the class label of the discriminator whose raw output (ie. summed output of individual RAMs) is highest. This leaves an ambiguity when both discriminators have the same output to the same pattern. This ambiguity can be removed in one of several ways.

One method is to follow the lead of the PLN and define a logical **u** output which is passed through a probability generator and outputs **1** or **0** with equal probability. This has the advantage of not biasing the classifier, but the disadvantage of making the concept of a dichotomy ill-defined. A trained machine may give different outputs to the same pattern at different times and hence it is hard to say what dichotomy the machine is performing. It is instead possible to introduce the concept of a **deterministic dichotomy** which is considered to exist on an input set only if none of the input patterns has a probabilistic output. It can be seen that such an arrangement gives a system similar to the thick threshold discriminator whereby the raw discriminator outputs need to be different by more than one. (In fact, in terms of the reformulation of section 5.5 this discriminator with deterministic dichotomies only can be considered as a 3-valued thick threshold discriminator.) Whatever the exact formulation, the existence of a probabilistic value clouds the issues surrounding the classifier and another approach will be taken.

Rohwer and Morciniec in [45] set the output of a "tied" classifier to be that of the class which had most training examples on the grounds that that is the class with the highest a priori probability. This method is well-motivated, but makes the analysis tricky just because of a rather rare case. Instead the convention is adopted that if a pattern elicits equal outputs from the two discriminators it outputs **1**. This biases the classifier, but hopefully to an insignificant degree in most cases.

5.4.2 The VC Dimension Bounds

The Lower Bound

To get a lower bound we may note that a single discriminator with $\Theta = N$ has VC dimension equal to $N(2^n - 1)$ and we may modify the training set used to prove the lower bound for that to get a lower bound for the n-tuple classifier. Consider the following set

$$\begin{aligned} T &:= \{0,1\}^n : \mathbf{0} : \ldots : \mathbf{0} \\ &\bigcup \ \mathbf{0} : \{0,1\}^n : \mathbf{0} : \ldots : \mathbf{0} \\ &\bigcup \ \ldots \\ &\bigcup \ \mathbf{0} : \ldots : \mathbf{0} : \{0,1\}^n \\ &- \ \mathbf{0} : \mathbf{0} : \ldots : \mathbf{0} : \mathbf{0}. \end{aligned} \tag{5.6}$$

We may realise any dichotomy $\chi := (R, U)$ where R is the set to give output $\mathbf{1}$ and U is the set to give output $\mathbf{0}$ as follows. Train all components of all elements of R into $\mathcal{D}_1$ and all components of all elements of U into $\mathcal{D}_0$. Thus we have $\mathcal{D}_1(r) = N \ \forall r \in R$ and $\mathcal{D}_0(u) = N \ \forall u \in U$. Since a single discriminator is able to make this dichotomy we know that $\mathcal{D}_1(u) < N \ \forall u \in U$ and $\mathcal{D}_0(r) < N \ \forall r \in R$. Hence the n-tuple classifier, comparing the outputs of $\mathcal{D}_1$ and $\mathcal{D}_0$, outputs $\mathbf{1}$ for $r \in R$ and $\mathbf{0}$ for $u \in U$ as required. Since χ was chosen arbitrarily this shoes that $\mathcal{W}$ shatters T. Hence

$$VCdim(\mathcal{W}) \geq N(2^n - 1). \tag{5.7}$$

The Upper Bound

First, in order to obtain an upper bound on $VCdim(\mathcal{W})$ we consider the total number of distinct hypotheses available on the input space. Given any set of hypotheses $\mathcal{H}$, $VCdim(\mathcal{H}) \leq log_2|\mathcal{H}|$. Therefore counting the hypotheses available to n-tuple classifier will allow us to bound the VC dimension of n-tuple classifier above.

There are $N2^n$ addressable locations in each N-RAM, n-tuple discriminator. As before, particular instantiation of these values will be termed a **node value set.** There are two such discriminators in our n-tuple classifier and hence $2N2^n = N2^{n+1}$ locations. This gives $2^{N2^{n+1}}$ possible node value sets.

However, if a particular address has the same value in both discriminators it is not having any effect on the hypothesis generated by the n-tuple classifier. Thus two node value sets which only differ at locations where the two discriminators have the same value will generate the same hypothesis on $\{0,1\}^{Nn}$. Consider the node value sets of $\mathcal{D}_0$ and $\mathcal{D}_1$ as binary patterns and let h be the Hamming distance between them. This means that the node value sets differ at h locations and so there are $\binom{2^{N2^n}}{h}$ ways of choosing the locations. We can then calculate how many distinct hypotheses are generated by all node value sets in which h takes a fixed, given value.

In the case $h = 0$ the node values are the same in $\mathcal{D}_0$ and $\mathcal{D}_1$. Thus only one distinct hypothesis is generated by all of the 2^N eligible node value sets. For any choice of node value sets at distance h there are 2^h distinct hypotheses generated on $\{0,1\}$. Hence the number of dichotomies of a subset of the input space is less than

$$\sum_{h=0}^{N2^n} \binom{N2^n}{h} 2^h = (1+2)^{N2^n} = 3^{N2^n} \tag{5.8}$$

by applying the binomial theorem.
We may take the base 2 logarithm of the right-hand side of (5.8) to get the following bound on the VC dimension.

$$VCdim(\mathcal{W}) \leq \log_2 3.N2^n. \tag{5.9}$$

Thus we have

$$N(2^n - 1) \geq VCdim(\mathcal{W}) \leq (log_2 3)N2^n. \tag{5.10}$$

The upper bound seems to be loose, and in trial-and-error work no training set that contained patterns which were inseparable from the rest of the training set (in the sense of maximal threshold discriminators) was successfully shown to be separable. It is suggested as a conjecture for future analysis that no such set is shatterable and that the VC dimension of the n-tuple classifier is exactly $N(2^n - 1)$.

5.5 A Reformulation of the Two Discriminator Classifier

The bound on the number of hypotheses, and more precisely the fact that it is considerably lower than the number of possible node value sets suggests a different formulation for the n-tuple classifier with two discriminators. Suppose we have two trained discriminators $\mathcal{D}_1$ and $\mathcal{D}_0$ which comprise a classifier $\mathcal{W}$. Let $\mathcal{E}$ be a trained 3-valued discriminator with the same parameters as $\mathcal{D}_1$ and $\mathcal{D}_0$ and whose stored values lie in $\{-1, 0, 1\}$ and are formed by subtracting the corresponding stored values of $\mathcal{D}_0$ from $\mathcal{D}_1$. If we consider $\mathcal{E}$ to be a discriminator thresholded at 0, then $\mathcal{E}$ is functionally identical to $\mathcal{W}$.

This functional identity takes advantage of the fact that locations storing the same value have no effect on the discrimination. Moreover there are exactly $3N2^J$ possible node value sets for a discriminator of type $\mathcal{E}$, so the bound on the number of hypotheses realisable by $\mathcal{W}$ is immediate.

5.5.1 Advantages of the Reformulation

Since we have already derived the bound on the number of hypotheses, it is apposite to ask whether this reformulation has any benefits. There are several. It becomes easier now to determine which node value sets correspond to different hypotheses, thereby tightening up the bound on the number of hypotheses. Also in this formulation the n-tuple classifier becomes a more direct extension of the discriminator so that results and insights about discriminators can be more easily transferred to the classifier. It also suggests a role for discriminators with even larger output sets, which offers an important possibility for the application of the structural risk minimisation principle to a family of such machines. However, first we need more information about the VC dimension of, and the number of hypotheses realisable by, such multi-valued machines.

5.5.2 Counting the Hypotheses

We shall use the symbol $\mathcal{E}$ to represent the discriminator with output values in $\{-1, 0, 1\}$ and threshold at 0, and the symbol $\mathcal{F}_3$ to represent the equivalent discriminator which takes values in $\{0, 1, 2\}$ and is thresholded at N. (We can then extend the notation to encompass all $\mathcal{F}_i$.) As in the normal discriminator case, given two node value sets we must try and find a pattern which they discriminate differently. As before we shall argue inductively, but this time informally, that most node value sets do indeed generate distinct hypotheses.

We are considering three-valued discriminators with N nodes and the threshold set to N: that is to say, examples of $\mathcal{F}_3$. Suppose we have some $\mathcal{F}_3$ discriminator called $\mathcal{F}^+$. Pick some node d and consider the $N-1$ node discriminator $\mathcal{F}^-$ formed by removing d. What conditions on $\mathcal{F}^-$ are necessary and sufficient for at least some of the possible value sets at d to yield identical hypotheses? This will happen if and only it is possible to interchange some pair of distinct stored values in d without altering the hypothesis generated. We have three pairs of node values to consider: (0,1), (1,2) and (2,0).

Let us consider (2,0) first: $\mathcal{F}(p:0) = \mathbf{0}$ while $\mathcal{F}(p:2) = \mathbf{1}$ if and only if there is any pattern p in the input to $\mathcal{F}^-$ such that $\mathcal{O}^-(p) = N-2$ or N. Hence for 0 and 2 to be interchangeable we require that $\forall p, \mathcal{O}^-(p) < N-2$ or $> N$. Moreover, if there are p and q such that $\mathcal{O}^-(p) < N-2$ and $\mathcal{O}^-(q) > N$ then there is some pattern r formed by combining components of p and q such that $\mathcal{O}^-(q) = N$. Hence the condition we require is even stronger, namely

$$\forall p, \mathcal{O}^-(p) < N-2 \textit{ or } \forall p, \mathcal{O}^-(p) > N.$$

We can reason similarly that for 1 and 2 to be interchangeable we must have

$$\forall p, \mathcal{O}^-(p) < N-2 \textit{ or } \forall p, \mathcal{O}^-(p) > N-1$$

and for 1 and 0 we require

$$\forall p, \mathcal{O}^-(p) < N-1 \textit{ or } \forall p, \mathcal{O}^-(p) > N.$$

This splits the set of node value sets on $\mathcal{F}^-$ into four groups as follows:

1. Those such that $\forall p, \mathcal{O}^-(p) < N - 2$.
2. Those such that $\forall p, \mathcal{O}^-(p) < N - 1$ and $\exists p_0, \mathcal{O}^-(p) = N - 2$.
3. Those such that $\forall p, \mathcal{O}^-(p) > N$.
4. Those such that $\forall p, \mathcal{O}^-(p) > N - 1$ and $\exists p_0, \mathcal{O}^-(p) = N$.

If we once again assume that N is large, then cases 1 and 3 dwarf the effect of the other two. Moreover, these are just the cases where whatever the node value set is at d, the hypothesis generated by $\mathcal{F}^+$ on the whole input space is constant. Hence we obtain the result that almost all node value sets generate distinct hypotheses or a constant hypothesis. Moreover, for $N >> n$ the number of constant hypotheses is tiny compared to the others (see Appendix A) so we once again deduce that the number of node value sets of $\mathcal{F}_3$ discriminators is a good approximation to the number of distinct hypotheses. Since $\mathcal{F}_3$ is functionally equivalent to the two-discriminator n-tuple classifier $\mathcal{W}$ we can deduce that the result is also true for $\mathcal{W}$.

5.6 Larger Output Sets

The construction of the set $\mathcal{F}_3$ as an isomorphic image of the set of n-tuple classifiers suggests a way of generating a whole family of learning machines. By increasing the set of allowed output values until N is reached we increase the functionality of the machines (or at least the number of available node value sets). The important question is as to whether or not we increase the VC-dimension. If VCdim$(\mathcal{F}_i)$ increases with i then we have a candidate for a structure to which the SRM principle may be applied. If not, then since the functionality of the $\mathcal{F}_i$ does increase we may be able to reduce the empirical risk on our data without increasing the complexity of the model (as measured by the VC dimension). To progress any further in this analysis the VC dimensions of all the $\mathcal{F}_i$, including $\mathcal{F}_1 \equiv \mathcal{D}_{\frac{N}{2}}$, must be calculated.

Work up to now has not produced a set of size greater than $N(2^n - 1)$ which is shatterable by any member of any of the $\mathcal{F}_i$. It seems that the sets used for the lower bounds in this chapter cannot be extended to larger sets shatterable by the $\mathcal{F}_i$ and if larger separable sets exist they must contain elements with n-tuple distance between them of more than two. (The term "n-tuple distance" refers to the number of n-tuples on which the patterns differ.) Work is proceeding to find accurate measures of these VC dimensions. At present all than can be said with certainty is that VC dimension does not decline with i. Unfortunately the large number of ways in which a training set may be trained makes theoretical upper bounds hard to generate. However for any particular distribution of input patterns it is possible to define the so-called *effective VC dimension* which can be used to obtain similar bounds on generalisation. The definition and empirical calculation of this quantity is given in the next chapter.

Chapter 6

Experimental Results and Comparisons

One use which can be made of the VC dimension is to bound the number of training examples that a learning machine which obeys the ERM principle requires to achieve a given level of generalisation. The formulae governing the bounds were given in chapter 3 and the results they give for n-tuple systems are given in this chapter.

6.1 Theoretical Performance Predictions

6.1.1 Weightless Systems

The only VC dimension that was calculated precisely was that for the maximal discriminator, $\mathcal{D}_N$, which is $N(2^n - 1)$. The upper bound for the sub-maximal discriminator is $N2^n$ while for the n-tuple classifier it is $\log_2 3.N2^n$, although it is conjectured that both of these systems also have a VC dimension of about $N(2^n - 1)$. The results quoted in chapter 3 then allow us to derive a probabilistic upper bound on the generalisation error with any probability η we choose. To recap,

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \frac{1}{2}\sqrt{\mathcal{E}} \tag{6.1}$$

where

$$\mathcal{E} = 4\frac{h(\ln\frac{2l}{h}+1)-\ln(\frac{\eta}{4})}{l}. \qquad (6.2)$$

This upper bound assumes no a priori knowledge of the problem: neither of the input distribution nor of the classification itself. In this sense it can be described as a worst case bound. However, given that we have no such a priori knowledge it is the best bound we can expect. (This is not strictly true. If we knew the growth function explicitly rather than bounding it by a function of the VC dimension we could possibly tighten up the bound, but work by Vapnik and others have shown [58] that in the worst case this bound is tight.)

We may therefore calculate $\mathcal{E}$ for the various machines. For the maximal discriminator, $\mathcal{D}_N$ we have

$$\mathcal{E}_{\mathcal{D}_N} = 4\frac{N(2^n-1)(\ln\frac{2l}{N(2^n-1)}+1)-\ln(\frac{\eta}{4})}{l} \qquad (6.3)$$

and the other bounds can be written down by means of similar substitutions.

6.2 Comparison with Experiment

Can these theoretical bounds be tested empirically? Since they hold for all distributions, a proper empirical test would involve generating data sets using every possible distribution and classification. This fact renders testing the bound directly impossible in all but the most trivial of cases. Indeed, since we can be sure analytically that they do hold we do not need to test them at all. More important is the tightness of the bound on the generalisation error. It may be true that real-world problems require in general far fewer training examples than the $O(N(2^n-1))$ suggested by the bound above. However, to know this requires a priori knowledge of the data and hence has not been considered in the preceding analysis. Nevertheless, it is possible to make use of experimental data to draw some qualitative conclusions.

6.2.1 Benchmarking with StatLog

The n-tuple classifier has been used by Rohwer and Morciniec [45] in benchmarking tests which compare the performance of the RAM-based method with other neural and statistical machines. The source of the data was the StatLog project (ESPRIT project 5170, see [34]) which was designed to achieve precisely this sort of benchmarking on a range of real-world data sets.

Although we cannot exactly predict generalisation performance on any dataset, the VC dimension bounds hold over all distributions, the results of Rohwer and Morciniec give an indication of the tightness of the bounds in some, hopefully typical, cases. As we shall see, the raw bounds turn out to be very loose indeed.

The Nature of the Trials

The StatLog data sets mainly comprised real-valued vector data with discrete-valued outputs. For the purposes of this comparison, only the data sets with boolean outputs are considered. There were eleven such sets. The real-valued inputs were encoded as binary vectors using the CMAC encoding [6] and the RAMs were trained using the original method; all components of all training patterns were addressed and trained. The experiments were performed with an n-tuple size of 8, although it is reported that a size of 6 gave similar results. The finding was that the n-tuple classifier performed comparably with most other methods apart from on four of the data sets on which its performance was worse than random.

Table 6.1 lists the error rates on each training set together with the number of training examples available and an upper and lower bound on the of training examples predicted by the VC dimension results to be necessary to ensure such a level of generalisation with probability 0.05. The lower VC dimension bound is $N(2^n-1) = 255,000$; the upper is $log_2 3N(2^n-1) = 404,165$. It is important to notice that the value of the largest prior is not directly used by the classifier. However when Rohwer and Morciniec evaluate the performance of the classifier they do so relative to the (almost) trivial classification assigning every pattern to the class with

Problem	Error rate	Largest Prior	Number of examples given
Belgian II	0.70	0.924	2000
Cut50	0.056	0.941	11220
Cut20	0.055	0.941	11220
Belgian I	0.053	0.566	1250
Tsetse	0.050	0.508	3500

Problem	Fewest theoretically required	Most theoretically required
Belgian II	$5.50 * 10^6$	$8.7 * 10^6$
Cut50	$128 * 10^6$	$200 * 10^6$
Cut20	$128 * 10^6$	$200 * 10^6$
Belgian I	$134 * 10^6$	$210 * 10^6$
Tsetse	$143 * 10^6$	$226 * 10^6$

Table 6.1: Generalisation performance on StatLog against predictions from VC dimension bounds.

the largest prior. Hence Cut20 and Cut50, although they have low overall error rate, are judged failures by this criterion. However from a pure learning theory standpoint they are successful. This illustrates the problems caused to the the analysis when prior knowledge is introduced.

6.2.2 Conclusions from the StatLog trials

It is clear from the table above that far more training data should be available if the low levels of error achieved were to be guaranteed with high probability by the VC dimension analysis. As a rule of thumb the number of training examples should be at least of the order of the VC dimension of the classifier. For this trial of the n-tuple classifier on the StatLog database this is not the case. Thus some other explanation is required for the unexpectedly good performance of the n-tuple method. One such explanation is considered in the next section.

6.3 Reasons for the Loose Bounds

Most importantly it must be stressed that five experiments hardly offer any information about the general problems discussed theoretically so far. Hence the discussion can only be informal. However two main facts apply. Firstly the training algorithm used is surprisingly insensitive to excess capacity if data are fairly tightly clustered. This is because in the cases considered two patterns which share an n-tuple probably also share an $n+1$-tuple – indeed Rohwer and Morciniec report very similar results with either 6-tuples or 8-tuples. In the general case however this is not true. The training algorithm takes advantage of this by keeping this information rather than discarding n-tuples which may be in conflict with other patterns as can happen in an ERM algorithm. The implicit assumptions of the algorithm are that input patterns occur in Hamming distance clusters and that n and N are large enough to avoid clashes on training patterns (ie. saturation). This is not an approach to solving the pure learning problem, and the role of such data-dependent techniques is discussed in chapter 7.

The second important reason also has to do with the distribution of patterns in real-world problems. As has been stressed repeatedly, the VC dimension bounds are independent of the distribution over the data. In this sense they are completely general and assume no a priori knowledge of the problem. The disadvantage is that they are in a sense "worst case" bounds, because they have to consider even pathological problems which one would probably not even try to solve with an n-tuple classifier. The set constructed to prove the lower bound of the VC dimension of the classifiers, in section 5.4.2, is an example of such a hard case.

For this reason attempts have been made to reformulate the bounds on generalisation error using quantities other than the VC dimension which reflect at least some assumptions about the problem in hand. In the cases where the large shatterable sets are highly unlikely to occur this will allow the error bound to be significantly tightened up. One such quantity which was discussed in section 3.3.1 is VC entropy.

However this is an analogue not of VC dimension, but of the growth function and hence it is not a convenient scalar quantity like VC dimension. It is also hard to calculate and so highly dependent on the data that it is not likely to be any easier to calculate than the target function itself.

If something, but not everything is known about the data distribution then the function required in the bound is the **generalised growth function** which is defined as follows). Suppose that it is known that the data follow a probability from a set $\mathcal{P}_0$ which is smaller than the set of all possible distributions, then we define the generalised growth function in a similar way to the standard growth function, except that the maximum is taken over all distributions F in $\mathcal{P}_0$ rather than $\mathcal{P}$. Formally,

$$G^{\Lambda}_{\mathcal{P}_0} = \ln \sup_{F \in \mathcal{P}_0} E_F[N^{\Lambda}(z_1, z_2, ..., z_l)].$$

However, this function is not easily calculated for most problems. Indeed, as is the case for VC entropy, calculating it could involve as much work as actually solving the problem.

The SRM principle in 3.4 gives another way of reducing the VC dimension of a class of functions by decomposing the whole class into a chain of subclasses with increasing VC dimension. However this method is not applicable to analysing the Rohwer and Morciniec experiments because no such decomposition was made. More will be said about SRM in chapter 7 where the n-tuple method will be considered in light of current techniques for matching an optimal learning machine to a problem.

A further, and more tractable way of incorporating distribution information into the generalisation bound is by means of the **effective VC dimension**. This quantity is defined by Vapnik, Levin and LeCun in [59] and is considered in the next section.

6.4 Effective VC Dimension

Effective VC dimension is defined by considering the dichotomies defined on subsets of the input space whose probability is high but possibly less than 1. As long as the probability of the subset, X' say, is "sufficiently large", the VC dimension of the functions restricted to X' may be used in place of the VC dimension in the generalisation error bound. The effective VC dimension is then defined to be the lowest VC dimension of the set of functions restricted to the subsets of probability "nearly 1". The technical details here have been skirted round, but are presented in the paper [59]. For the purposes of this thesis it will be sufficient to understand merely the properties of the effective VC dimension as they apply to RAM-based systems.

If it is true that the large shatterable subsets which fix the lower bound of the VC dimension are indeed uncommon under most distributions, it may be that they are ignored in the calculation of the effective VC dimension for many distributions. If so we may expect the effective VC dimension to be significantly lower than the actual VC dimension and hence our generalisation bound will be correspondingly tighter. This leaves one major problem: how is the effective VC dimension to be calculated?

The definition of effective VC dimension in terms of subsets of probability close to 1 does not make its calculation easy. However, Vapnik et al. do offer an empirical method for making the calculation in the same paper, [59]. Experiments and calculations based on this method were then carried out on various n-tuple classifiers.

6.4.1 Definitions and Theory

The details of the derivation of the following results are all to be found in [59] and will not be repeated. What follows is an exposition of the key results and the key assumptions.

The empirical estimation of effective VC dimension (henceforth EVC dimension) depends upon estimating a bound on the maximal deviation of the generalisation error on two half samples. Given a training sample xy, this quantity can be defined as follows:

$$E[\xi_l] := E[sup_{\alpha \in \Lambda}(R_{emp}(\mathrm{x}, \alpha) - R_{emp}(\mathrm{y}, \alpha))]. \tag{6.4}$$

where $E[\ldots]$ denotes the expectation with respect to the fixed but unknown input distribution P. Here x and y should be considered as sets drawn at random, according to P, of labelled examples each of length l. That is, there are l labelled examples in each of x and y. As before, $\alpha \in \Lambda$ parametrises the hypotheses realisable by the learning machine, and the maximum deviation between the error on each half sample (when tested with the same hypothesis) is taken over the whole hypothesis set. What is important is that this quantity can be empirically estimated for a given sample xy.

Theoretical considerations suggest that the quantity in 6.4 is tightly bounded by a function of the half-sample length, l, and the effective VC-dimension of the classifier, h. The following function estimates this bound for $\tau = (l/h)$. Hence we have

$$\Phi(\tau) := \begin{cases} 1 & \text{if } \tau < 0.5 \\ a\frac{\ln(2\tau)+1}{(\tau-k)}(\sqrt{1 + \frac{b(\tau-k)}{\ln(2\tau)+1}} + 1) & \text{otherwise} \end{cases}$$

where a and b are free parameters and k is derived from the condition of continuity at $\tau = 0.5$. For smaller values of τ (say < 5) an acceptable approximation is

$$\Phi_1(\tau) := \begin{cases} 1 & \text{if } \tau < 0.5 \\ d\frac{\ln(2\tau)+1}{(\tau+d-0.5)} & \text{otherwise} \end{cases}$$

where there is only a single free parameter d. Under a certain number of assumptions these bounds can allow us to approximate the effective VC dimension of any classifier on any distribution.

Necessary Assumptions

For an empirical estimate of EVC dimension to be made by the method described the following assumptions must hold true.

- $E[\xi_l]$ does not depend on the distribution of the classes, only the patterns themselves.
- The expected deviation depends on the learning machine only through the effective VC dimension, h.
- $\Phi(l/h)$ or $\Phi_1(l/h)$ is a good approximation to $E[\xi_l]$ for appropriate a and b.
- a and b are constant over a large class of related learning machines.

These assumptions were verified by Vapnik et al. for linear threshold machines for which the EVC dimension for the uniform distribution is known. Unfortunately it is exactly because the EVC dimension is *not* known for the n-tuple classifier that this method is being employed, so verification of these assumptions was less complete.

6.4.2 Maximising the Error Divergence

Before trying to calculate the EVC dimension, or indeed before trying to verify the assumptions on which the method rests, it is necessary to find a way to experimentally find the maximum value of $E[\xi_l]$. The method employed is to invert the labels on the first half of the sample, $\mathbf{x}$, to obtain a new sample, $\bar{\mathbf{x}}$, and then minimise the total error on the new sample $\bar{\mathbf{x}}\mathbf{y}$. Again, the details are in [59].

However, as has been pointed out in section 2.4.1 and will be discussed in chapter 7, the standard n-tuple classifier algorithms do not attempt to minimise total error on the training sample because they are prone to saturate: even given training sets that, with careful setting of the node values, they may be able to classify with no errors. Hence a new algorithm was devised in attempt to find a best fit for the training data.

The Stochastic Minimisation Algorithm

The **stochastic minimisation algorithm** or SMA was devised with only one goal in mind: to fit a training set to an n-tuple classifier with minimal error. Considerations of time and space were ignored as long as the experiments in this thesis could be performed within the requisite three years. Also no theoretical proof is provided that this algorithm works in all cases; it is sufficient for the experiment that it works fairly well and consistently most of the time.

The SMA applies to single discriminator classifiers with any threshold and range of stored values. (In particular, since it has been shown that the standard two-discriminator n-tuple classifier, $\mathcal{W}$, can be represented in such a form when the stored values can be taken from $\{0, 1, 2\}$ and the threshold is set to N – see section 5.5 – the SMA can be applied to such a system.) The initial state of the classifier could be anything, but in this work every location was assumed to contain the median value of the allowed range. The algorithm proceeds as follows.

1. Set current training pattern to the first in the training set.
2. Test current training pattern. If it is classified correctly set current pattern to the next in the training set and repeat this step.
3. Count how many locations need to be changed to make the current output correct.
4. Randomly select the required number of locations and alter their values so as to make the current output correct.
5. Set current pattern to be the next in the training set and go back to step 2.

Notice that this algorithm doesn't have an explicit termination condition. In practice it terminates if zero error is reached on the training set or if further iterations through the training set are not producing lower error rates. Notice also that the training error rate does not decrease monotonically. It is necessary to store the lowest error rate achieved up to any point in order to make a comparison with the current error rate.

This algorithm is used throughout the following experiments to minimise the error on "flipped" the training sets of section 6.4.2 to estimate $E[\xi_l]$ and also on other data such as in [32] to assess its usefulness as a tool for training n-tuple systems in real applications. Repeated use showed that it gave consistent results for the systems and number of iterations used (typically several hundred per problem). Hence it could be used as at least a good approximation to a global minimisation algorithm.

6.4.3 The Experimental Method

To verify the assumptions, experiments were performed using computer simulations written in C. These modelled a wide class of n-tuple classifiers, generated random training and test sets according to certain specified distributions and allowed the classifier to be trained according to either the original, the Tarling-Rohwer or the stochastic minimisation algorithm above. The first assumption was tested for few cases by generating output values of differing difficulty and calculating $E[\xi_l]$ on them. The last two assumptions can only be approximately proved by fitting the theoretical error divergence, Φ, to the experimentally generated error curve. If this is done and shown to be a good model the best-fit effective VC dimension can be ascertained for each value of the classifier's parameters. It will then be possible to see (although only by comparing curves) if the free parameters of the theoretical error function are constant for a range of different classifiers.

The first experiments involve a uniform distribution of input patterns. (Note that in all cases the distribution must be the same during both training and testing.) The success of these experiments lead to work on a more complex input distribution, where the patterns are distributed with Gaussian Hamming distance around a single pattern. The results of these experiments are presented later in the chapter in section 6.4.5 where they are compared with the uniform results.

Varying the Output Distribution

To test the independence of the maximal values of the empirical deviation ξ_l from the distribution of the labels, training sets of increasing difficulty were constructed. This was achieved by selecting a base pattern and labelling the patterns depending on their distance from it. A probability value p is chosen and patterns with a relative

Hamming distance from the base pattern of ≤ 0.5 are assigned the output value **1** with probability p. Those further from the base pattern were assigned the output value **0** with the same probability p. Since the n-tuple classifier approximates a Hamming distance classifier (see for example Allinson and Kolcz in [7]) the problem is easier for the n-tuple classifier the larger the value of p. Conversely a value of 0.5 for p would ensure that labels were distributed randomly which will be a more difficult problem, especially for large training sets.

A standard two discriminator n-tuple classifier (modelled by a single three-valued discriminator) with $n = 4, N = 50$ was used for this part of the simulation. Early work with SMA showed that with these values consistent error rates were obtained on test problems. Thus it is reasonable to assume that the algorithm was finding local minima close to the global minimum. Training sets (of size 2l) were created for values of p ranging from 0.5 to 1 and values of l from 50 to 400. In order to find the values of ξ_l for each of these (p, l) pairs, the assigned output values of the first l training examples were inverted and the error on the resulting training set (approximately) minimised by the stochastic minimisation algorithm. As per Vapnik et al., the proportional error after minimisation was doubled and subtracted from 1 which gives the empirical estimate of ξ_l. The values obtained are plotted against l, the length of the half-sample, in figure 6.1. (Each data point is the average of at least 10 runs.)

We see from figure 6.1 that the maximal deviation curve for an n-tuple classifier with $n = 4, N = 50$ is very little affected by the distribution of the output labels, as hoped. For simplicity all further experiments to measure ξ_l use a uniform (50%) distribution of both labels with the assumption that the results so generated would be the same with any other output distribution. (However when the experiments are performed with other input distributions this independence must be re-checked.)

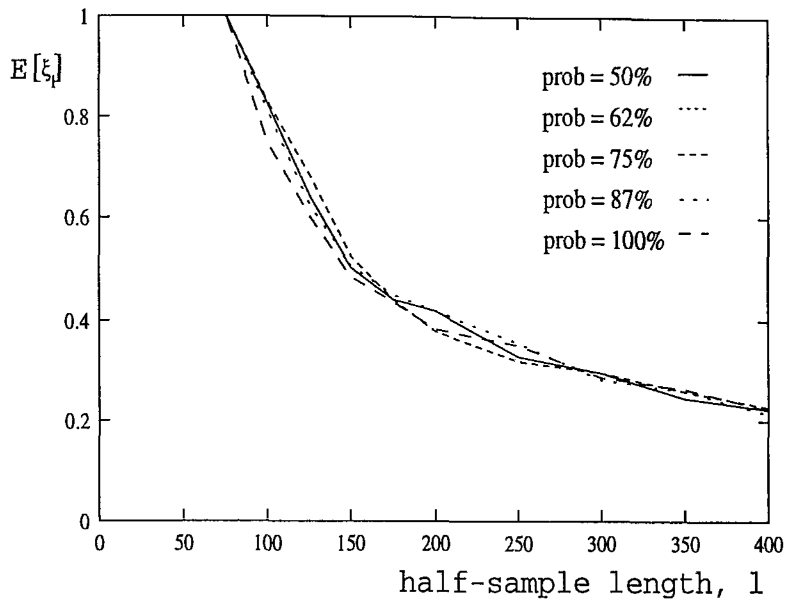


Figure 6.1: Empirically determined $E[\xi_l]$ for output distributions of varying difficulty. Uniform distribution.

Varying the Classifier

The second assumption, as to whether or not the estimate for ξ_l depends only on h could not be directly tested for n-tuple classifiers as there is no good theoretical estimate of effective VC dimension. Hence this assumption can only be proved right if some precise relationship between the parameters of the classifiers (and in particular the size of the hypothesis set they generate which at least gives an upper bound on actual VC dimension) and the effective VC dimension. Initially, however, it will simply be assumed to hold as well for the n-tuple classifier as Vapnik et al. showed it did for the linear classifier.

In practice the estimation of the effective VC dimension is done in two parts. Firstly for each set of differently parametrised classifiers a corresponding set of effective VC dimensions, h, is estimated such that the resulting curves of $E[\xi_l]$ against l/h coincide as far as possible. This gives the correct ratios of effective VC dimensions between the classifiers. To fix a base point only one of the experimentally generated curves needs to be fitted, by varying h, to an appropriately parametrised graph of Φ or Φ_1. Using the previously generated ratios the effective VC dimensions of all the classifiers may then be calculated from the value of h that fits the theoretical estimates. If this procedure is followed, and if the appropriate curves can be fitted, we have shown that the theoretical model really can estimate error deviation.

The last two assumptions could at least be circumstantially checked. If a wide range of n-tuple classifiers can be approximated by an appropriately parametrised version of Φ_1 then the third assumption is true. Moreover if these parameters are constant over the set of classifiers, then it may be reasonable to assume that the last assumption also holds. We shall consider the fate of these assumptions when the appropriate results are presented.

Values of ξ_l were calculated for various values of n and N as well as i, the range of the integers stored in each RAM location. Values were also calculated for for the maximal and $\Theta = N/2$ single discriminator classifiers. These are plotted in figure 6.2 for the single discriminators, figure 6.3 for varying values of N, figure 6.4 for varying values of n and figure 6.5 for varying values of i. To allow comparison the plot for the basic two discriminator classifier with $n = 4, N = 50$ is included in all graphs. (Note that for varying i the basic two discriminator classifier is equivalent to $i = 3$.) As a guide to interpreting these graphs it is worth noting that the further to the right a curve is, the higher its effective VC dimension.

To make an estimate of the EVC dimension, we must plot $E[\xi_l]$ against l/h and show that some curve Φ fits the resulting graph. If the same Φ fits all the $E[\xi_l]$ graphs then our assumption that the parameters a and b are independent of the learning machine will have been justified. We may also be able to judge whether or not the $E[\xi_l]$ depends on the machines other than through h. Since in almost all cases the ranges of l are such that $l/h < 5$, the approximation by Φ_1 with a single free parameter d is valid. Figures 6.6, 6.7, 6.8, 6.9 show the results for a range of machines and the best fit curve of type Φ_1, with $d = 0.225$. The same value of d fits all setting of the parameters.

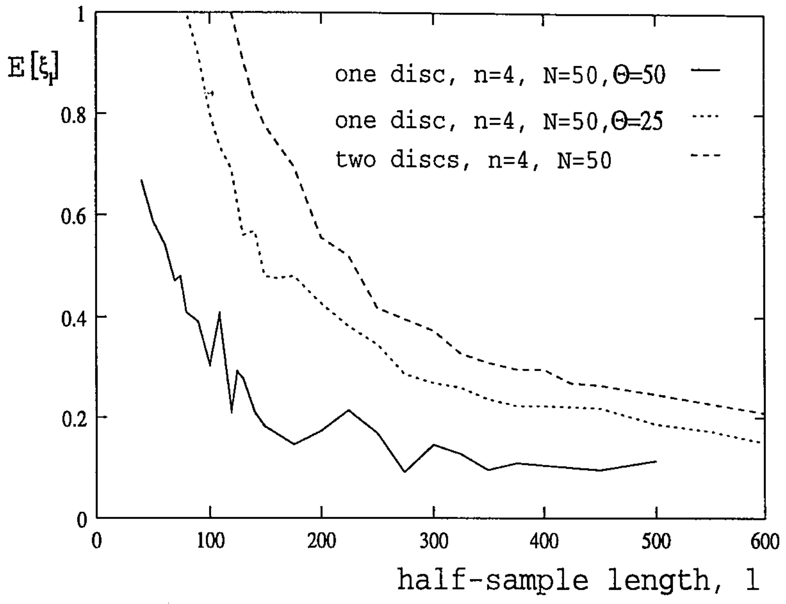


Figure 6.2: Empirical $E[\xi_l]$ for some single discriminator classifiers. Uniform distribution.

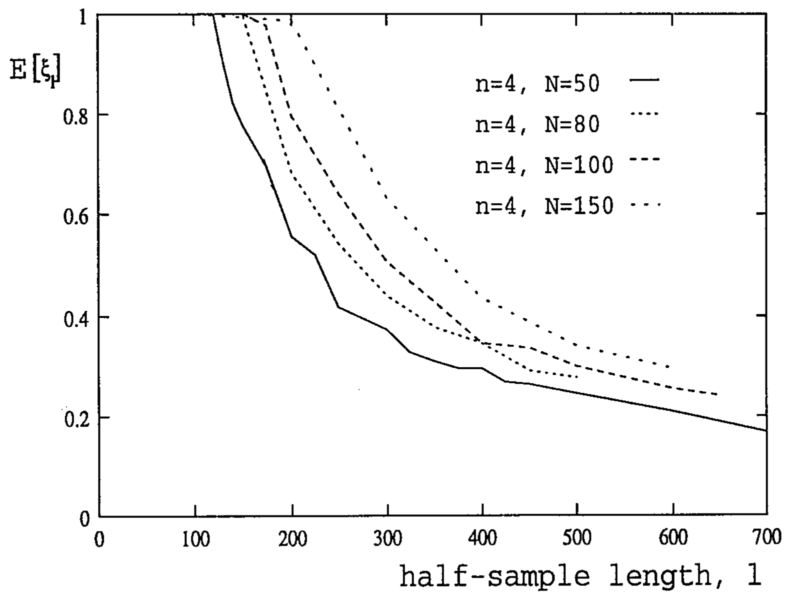


Figure 6.3: Empirical $E[\xi_l]$ for a range of n-tuple classifiers with varying number of nodes per discriminator, N. Uniform distribution.

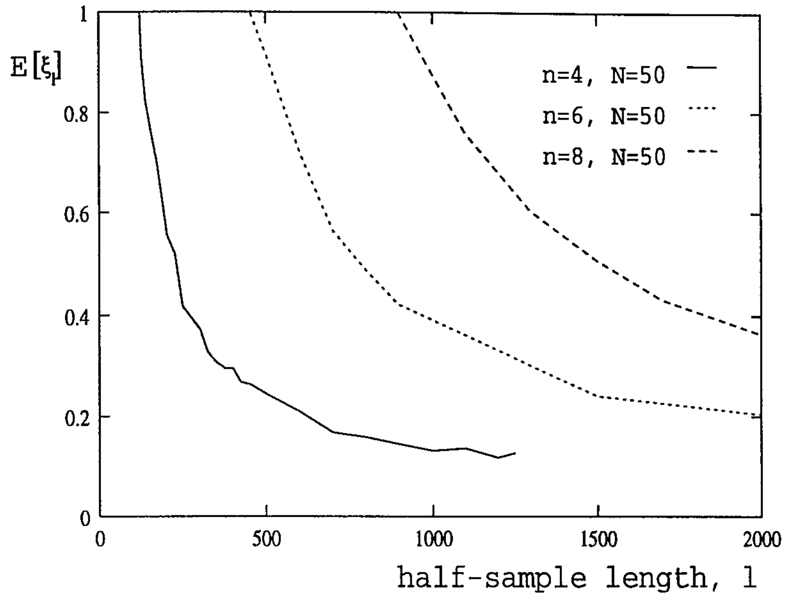


Figure 6.4: Empirical $E[\xi_l]$ for a range of n-tuple classifiers with varying n-tuple size, N. Uniform distribution.

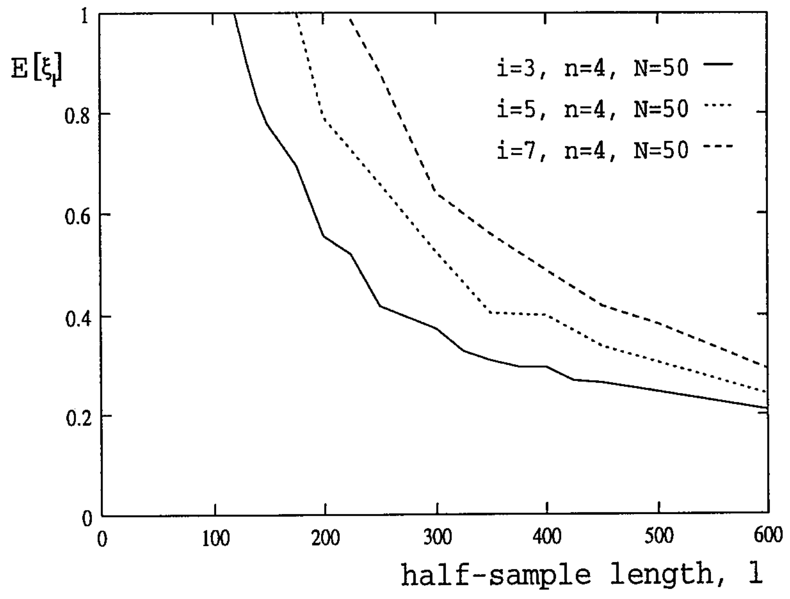


Figure 6.5: Empirical $E[\xi_l]$ for a range of n-tuple classifiers with varying size of output set, i. Uniform distribution.

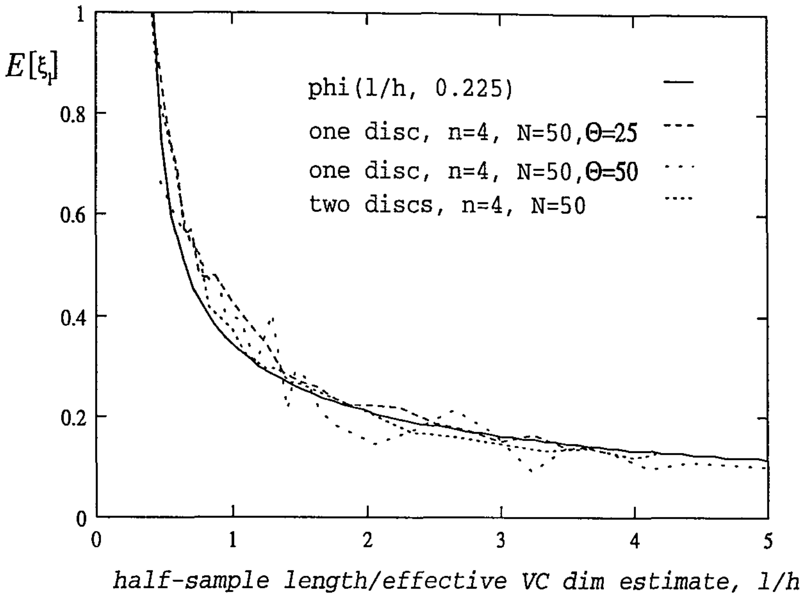


Figure 6.6: Empirical $E[\xi_l]$ for single discriminators against estimated EVCD over l plotted against a best-fit theoretical estimate. Uniform distribution.

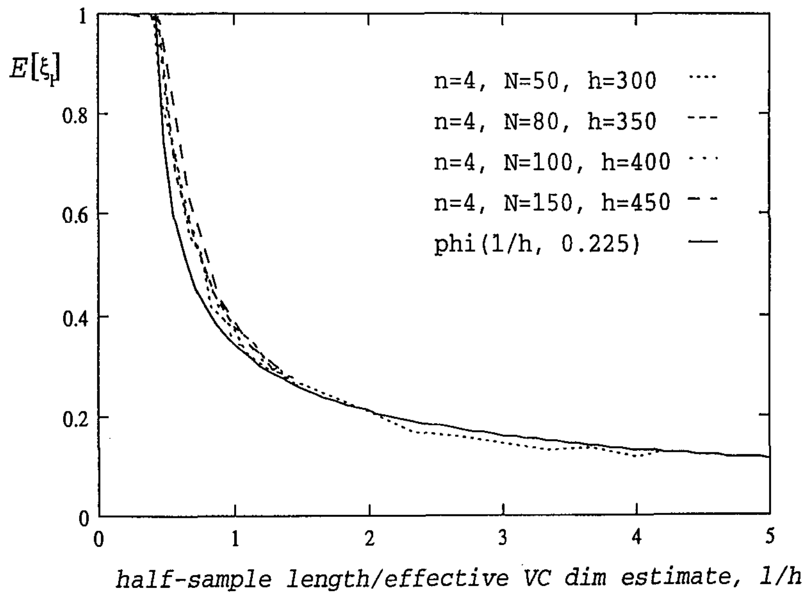


Figure 6.7: Empirical $E[\xi_l]$ for varying N against estimated EVCD over l plotted against a best-fit estimate. Uniform distribution.

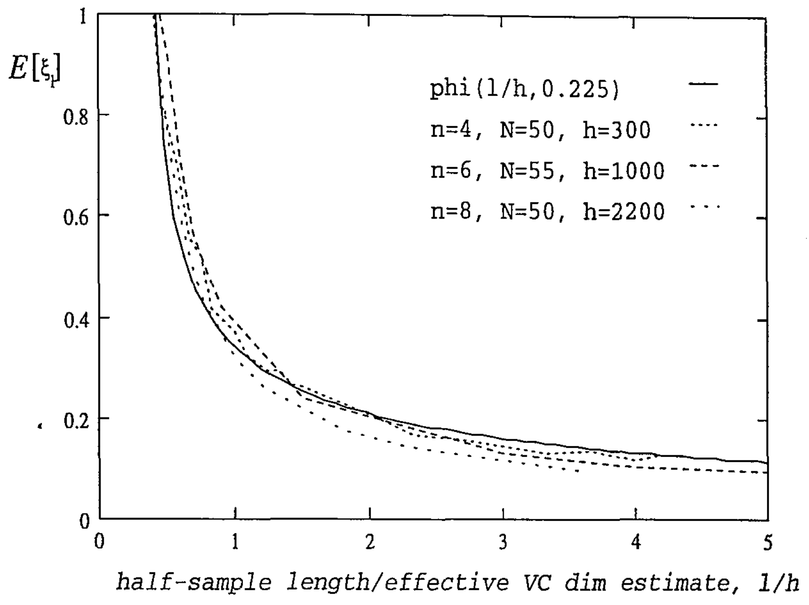


Figure 6.8: Empirical $E[\xi_l]$ for varying n against estimated EVCD over 1 plotted against a best-fit estimate. Uniform distribution.

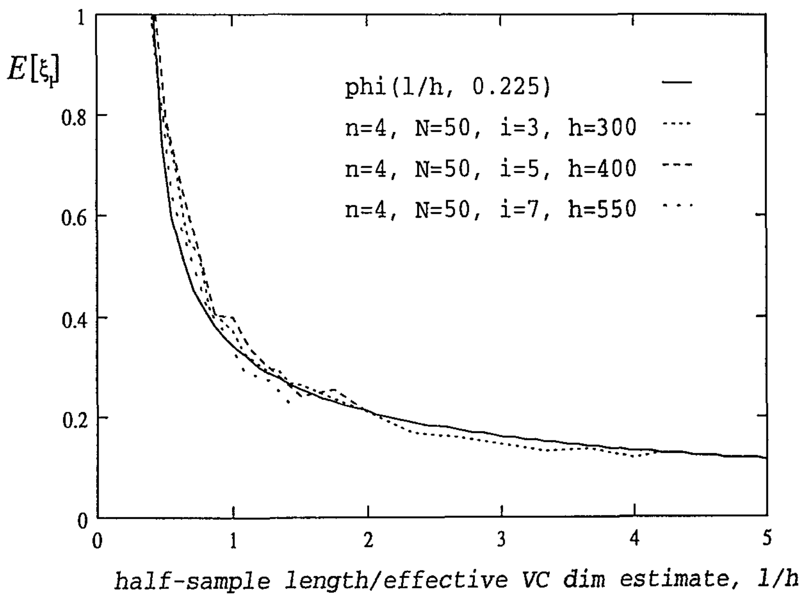


Figure 6.9: Empirical $E[\xi_l]$ for varying i against estimated EVCD over 1 plotted against a best-fit estimate. Uniform distribution.

6.4.4 Analysis of Results

Because this method relies on so many assumptions, backed up by curve fitting, it can never be counted on to be totally accurate. However there are several reasons to believe that it gives a good indication of the effective VC dimension of the n-tuple classifiers. It is worth considering the results in detail and seeing what conclusions can be drawn.

Firstly the consistency of the results over a wide range of values suggests that the assumptions of the method are justified. Figure 6.1 shows that the maximal deviation of errors on two half samples does not depend significantly on the output distribution; that is, on the way in which the output labels are assigned to the random input patterns. (Note that the input distribution remains unchanged as the uniform distribution in all graphs above.) If this were not the case we could not say that $E[\xi_l]$ depends only on the parameters of the classifier and the distribution of input patterns so the relationship between $E[\xi_l]$ and EVC dim could not be as simple as that given by $\Phi(l/h)$. This independence was also found to be true for linear classifiers by Vapnik et al..

The second set of graphs, figures 6.2, 6.3, 6.4 and 6.5, present the raw values of estimated $E[\xi_l]$ against the size of the half-sample, l, for various parametrisations of the n-tuple classifier. The results are qualitatively plausible in that the curves are shifted to the right - which implies higher EVC dim - according to the functionality of the classifier. Since the functionality provides and upper bound on the VC-dimension, and hence on effective VC-dimension, this is in line with what we would expect. (Although because the theoretical results are in general upper bounds only, this is not proven theoretically.) Functionality increases linearly in N, exponentially in n and logarithmically in i. Perhaps results like these can be used to see if effective VC dimension follows suit.

The final set of graphs, figures 6.6, 6.7, 6.8 and 6.9, plot $E[\xi_l]$ against l/h where h is an estimate of effective VCdim. Since we have no analytically derived values that can be used to fix the parameters of Φ and/or Φ_1 we must fit both the parameters (a, b or d) as well as the values h of effective VCdim. This matching problem possesses several degrees of freedom, but the resulting curves are closely fitted by Φ_1 with $d = 0.225$. Hence it is reasonable to assume that this method allows us to estimate effective VC dimension at least to an approximation equivalent to the worst fitted $E[\xi_l]$ curve. What is crucial is the fact that the effective VCdim results are considerably lower than the bounds and results for the true VC dimension.

The Gaussian Results

Since the experiments with uniformly generated input data were successful in that they showed the theoretical assumptions to be justified and yielded acceptable deviation curves, they were repeated with a different data distribution. Instead of selecting each bit of each pattern at random, a base pattern was chosen (the all zero pattern, without loss of generality) and Hamming distances from this base were selected according to a Gaussian (with the end of the tails truncated because the pattern space is discrete and finite) distribution with mean 0 and a variance of half the size of the pattern. (Thus the distribution was not constant between experiments). For each pattern generated, the requisite number of bits are set to one. This gives an input set of patterns whose Hamming distances from the all-zero pattern are distributed binomially. It is a more likely scenario for a "real-life" problem.

As before, output distributions of increasing difficulty were applied to the input sets to ascertain the effect of task difficulty on the expected maximal error deviations. This was done by splitting the input data into two approximately equal parts, those nearer and those further from the base pattern. A probability parameter, p, was selected and the outputs for those patterns nearer the base pattern were set to one according to a Bernoulli distribution with parameter p. Those further away were set according to $1 - p$. A probability parameter of 1 or 0 makes the task easier since patterns requiring similar outputs will be clustered. A parameter value of $p = 0.5$ is equivalent to a random distribution of classes. The results for several values of the probability parameter for the two discriminator classifier with $n = 4$ and $N = 50$

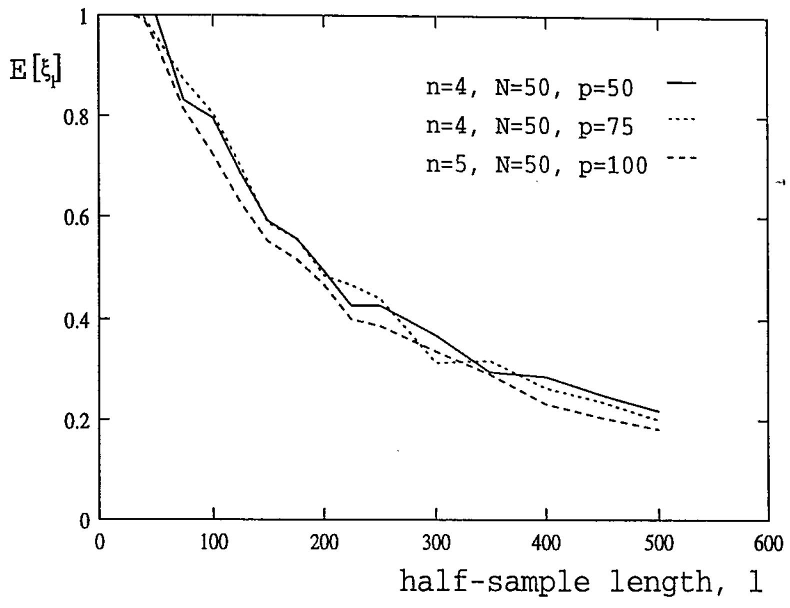


Figure 6.10: Estimates of $E[\xi]$ for Gaussian input and different difficulties of output.

are shown in figure 6.10. As before we see that $E[\xi_l]$ is unaffected by the difficulty of the task.

All the experiments performed with the uniform input patterns were repeated with the Gaussian data. The results for various n-tuple classifiers are in figures 6.11, 6.12 and 6.13 while the same results scaled by an estimate of effective VC dimension are given in figures 6.14, 6.14 and 6.16. The good fit with the theoretical estimate allows us to make estimates of effective VC dimension for Gaussian inputs and compare them with both the uniform case and the with actual VC dimensions. This is done below in tables 6.2 and 6.3.

6.4.5 What to Make of the Results?

Are the Results Accurate?

There are three immediate sources of inaccuracy in this method. Firstly there is the fact that the values of $E[\xi_l]$ are only estimates from fairly small sample sizes. This introduces noise into the data points. Secondly the data are generated by a deterministic random number generator ([41, "gasdev" on pages 288-290]) which, however good, may have some unforeseen correlations. The third problem is with

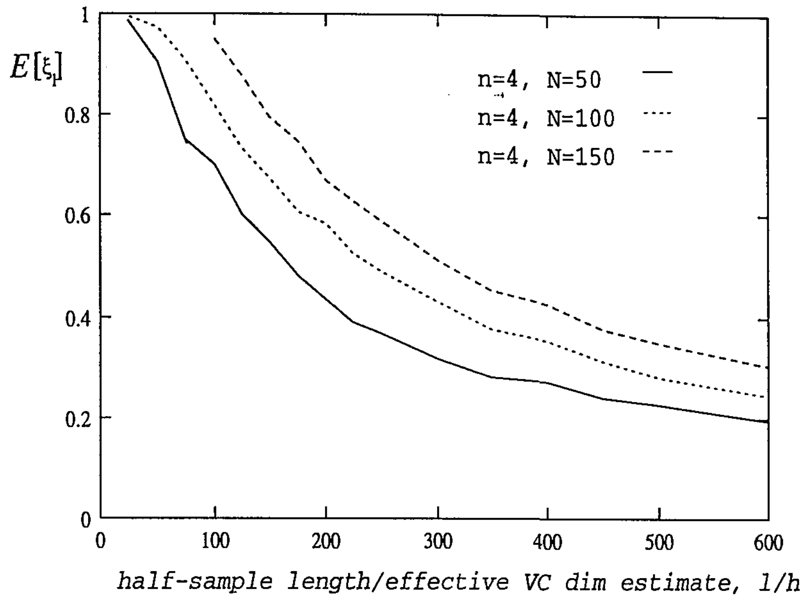


Figure 6.11: Estimates of $E[\xi]$ for Gaussian input n-tuple classifiers with varying N.

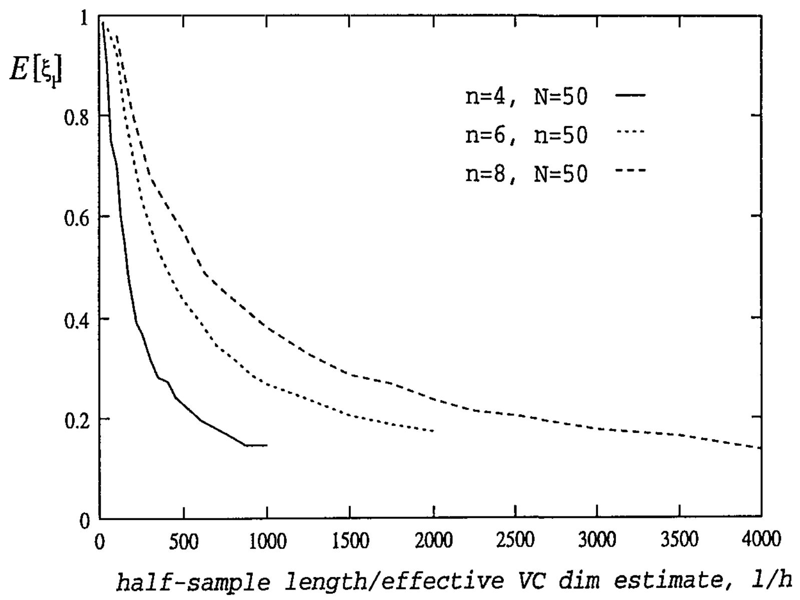


Figure 6.12: Estimates of $E[\xi]$ for Gaussian input and n-tuple classifiers with varying n.

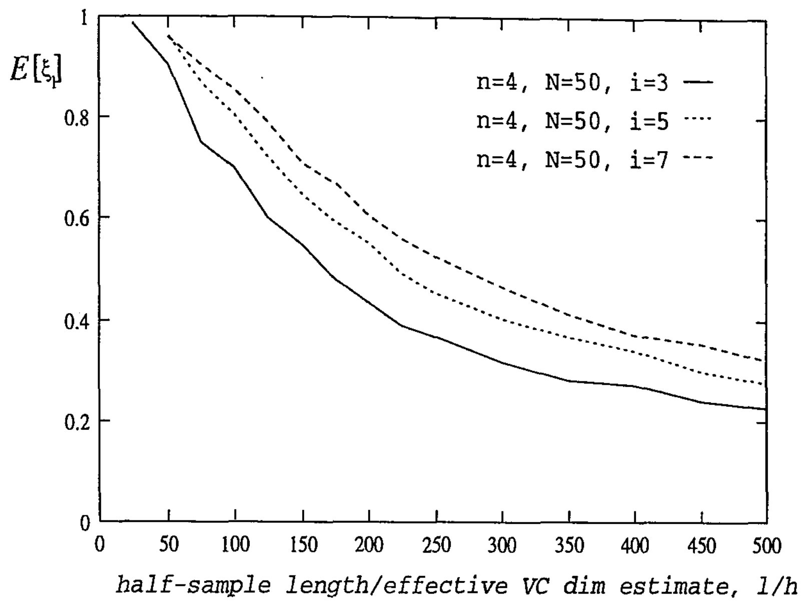


Figure 6.13: Estimates of $E[\xi]$ for Gaussian input and n-tuple classifiers with varying i.

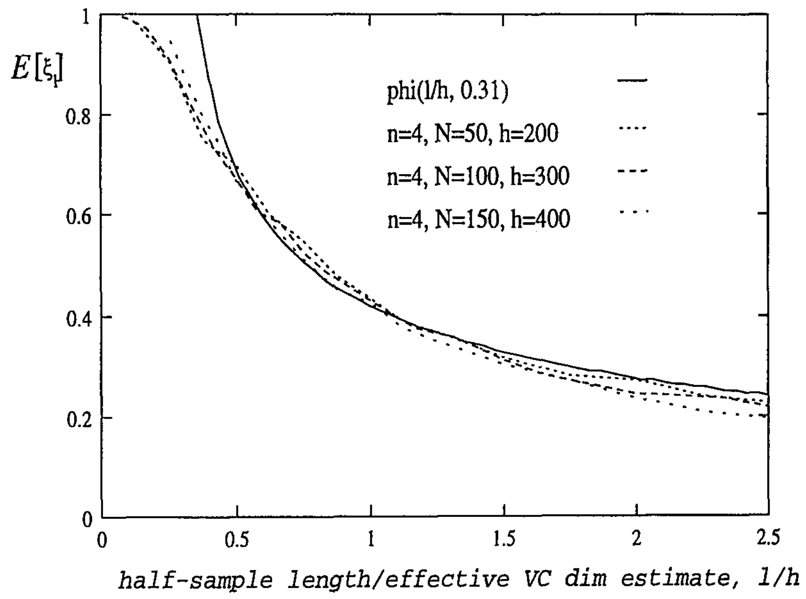


Figure 6.14: The results for Gaussian inputs normalised by effective VC dim and with a best fit theoretical error term – N varying.

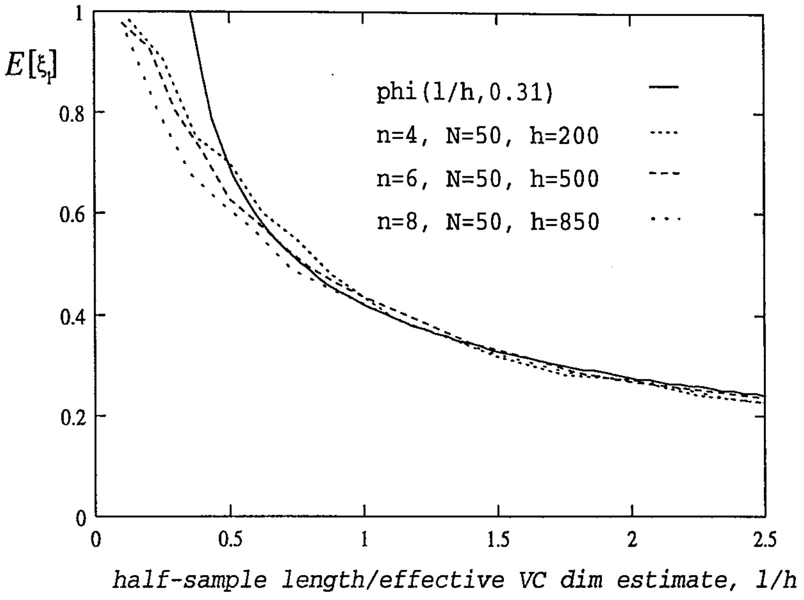


Figure 6.15: The results for Gaussian inputs normalised by effective VC dim and with a best fit theoretical error term – n varying.

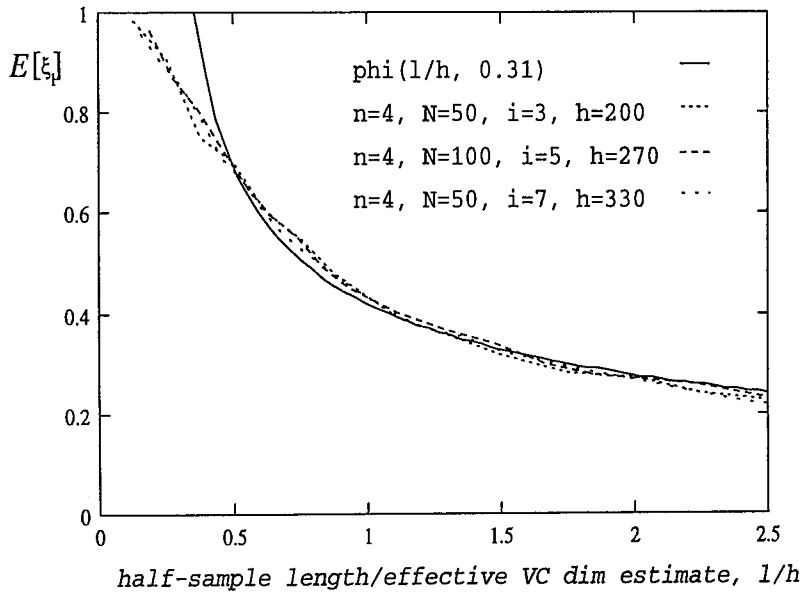


Figure 6.16: The results for Gaussian inputs normalised by effective VC dim and with a best fit theoretical error term – i varying.

the SMA which takes a long time to run over large sets and which does not *guarantee* a minimal solution. The data suggest an over-estimation of training error on large sets of large patterns. This leads to incorrectly low values of $E[\xi_l]$ which is seen from the graphs when the experimental result curves often go slightly below the theoretical prediction given by Φ_1. (In fact Φ_1 is simplification which should itself slightly under-estimate $E[\xi_l]$ for larger values of l/h.) This third effect is most noticeable in figure 6.12 where both pattern sets and pattern sizes are largest.

Nevertheless, given the caveats above, the fit between theory and experiment is fairly close for almost all cases. Moreover, the assumptions made in section 6.4.1 have been verified for both the uniform and the Gaussian cases with the exception of the independence of $E[\xi_l]$ from any property of the classifier other that h. However, circumstantially this assumption is not invalidated.

The Estimated Values

The known VC dimension values and bounds are shown in table 6.2 along with the corresponding effective VC dimension values for the uniform distribution. This table shows at a glance how pessimistic the VC dimension results are when the patterns are drawn from a uniform distribution. The equivalent results for the Gaussian distribution are given in table 6.3.

Table 6.2 shows clearly that the effective VC dimension of the n-tuple classifier over a uniform distribution of input patterns is significantly less than the actual VC dimension. How can this be explained? The answer lies in the definition of effective VC dimension which only requires dichotomies to be counted on sets of probability close to 1. The "pathological" sets created in chapters 4 and 5 are sparse in the set of all possible input sets, and can be ignored for the purposes of calculating EVC dim. The effect of this on the error bound is to push the possible error on such sets from the "epsilon" term, that is the maximum error expected, to the "delta" term, the confidence with which the error can be expected. This can only be done when something is known about the input distribution (in general it could be *only* the

n	N	i	VC dim >	VC dim <	EVC dim	Ratio 1	Ratio 2
4	50	3	750	1190	300	2.5	4.0
4	100	3	1500	2380	400	3.8	6.0
4	150	3	2250	3570	450	5.1	7.9
6	50	3	3150	4990	1000	3.2	5.0
8	50	3	12750	20,200	2200	5.8	9.2
4	50	5	750	1740	400	2.5	4.4
4	50	7	750	2110	550	2.5	3.8

Table 6.2: Effective (uniform) and actual VC dimensions for n-tuple classifier. Ratio 1 is the VC dim lower bound over EVC dim while ration two is VC dim upper bound to EVC dim.

n	N	i	VC dim >	VC dim <	EVC dim	Ratio 1	Ratio 2
4	50	3	750	1190	150	5.0	7.9
4	100	3	1500	2380	340	4.4	7.0
4	150	3	2250	3570	460	4.9	8.2
6	50	3	3150	4990	700	4.5	7.1
8	50	3	12750	20,200	2000	6.4	10
4	50	5	750	1740	320	2.3	5.4
4	50	7	750	2110	450	1.7	4.7

Table 6.3: Effective (Gaussian) and actual VC dimensions for n-tuple classifier. Ratio 1 is the VC dim lower bound over EVC dim while ration two is VC dim upper bound to EVC dim.

pathological sets which have probability > 0) which is why effective VC dimension requires some knowledge about the data.

The other important qualitative result is that the effective VC dimension is higher for the uniform distribution than for the Gaussian. This is not surprising, given that some patterns have a very low probability of being selected. Hypotheses which only differ on such patterns are not differentiated on input sets which do not contain those patterns. Hence one would be led to expect that the input distribution with fewest "hard-to-reach" patterns would have the largest effective VC dimension. This informal reasoning appears to be borne out by the results.

Does this mean that the n-tuple classifiers good performance on StatLog and elsewhere can be fully explained by these figures? Unfortunately not. The fact that the bounds are true for any distribution of output values means that even the EVC dim bounds are not fully optimised for any particular data set. Moreover it is not the case that the StatLog input data are distributed uniformly or even as sums of Gaussians so we cannot be confident that any such bounds apply. There is also the fact that the StatLog experiments used the original n-tuple classifier training which does not minimise empirical risk.

However, it is not really the purpose of these bounds to analyse particular results since any particular learning problem depends upon its data to a degree that is not taken into account by either the VC dimension nor the effective VC dimension. The practical usefulness of the theoretical results is predicting, before the problem is attacked with the learning machine, what generalisation error can be expected for what training set size. The general problem of minimising generalisation error is discussed in the following chapter in light of all the experimental data and theoretical results pertaining to it.

6.4.6 Investigating the SMA

Since the SMA is better able to minimise error on a training set one would expect it in general to outperform other training algorithms as long as the VC dimension (or effective VC dimension) of the classifier is not too large. In this subsection we briefly review a comparative experiment with the SMA. An example of such a classification experiment in which this approach worked is with the electro-cardiograph data of Manintveld [32]. Here the image consisted of 30,000 bits and the training set contained only 500 images. A subset of the data was taken so that each image could be classified as either normal or as having one particular abnormality.

The results with the original algorithm are given in [32] and discussed at length. Generalisation error was on the whole below 5%. When the machine was trained on the same data with SMA, the error was close to 50%. In both cases the classifier was the same, so had the same VC dimension and effective VC dimension on the data. In both cases there were no errors on the training data.

It is clear that in this case the fact that the original algorithm implicitly makes assumptions about the data is crucial to the success of the classifier. When treated as a pure learning problem there are simply insufficient training images to ensure good generalisation. This problem of the choice of algorithm is discussed in greater detail in chapter 7.

6.5 Other Systems

Before leaving the discussion of VC dimension bounds it is instructive to consider the VC dimensions of other machines including some of those involved in the StatLog test. Linear and non-linear perceptrons are considered since they are such popular neural network classifiers, as is the support vector machine, an improved linear classifier, and, for completeness, a classifier based on the sine curve which can be shown to behave maximally badly. These results will be presented and compared with the n-tuple classifier, but full discussion of the implications will be held over to the next chapter. Most of these results can be found in the summary paper by Maass [30].

The Linear Perceptron

The VC dimension of a single linear threshold gates with n real input and boolean output was shown by Wencour and Dudley to be

$$VCdim = n + 1. \tag{6.5}$$

Multi-Layer Perceptrons

The situation is more complicated for the non-linear perceptron with real weights and sigmoid activation function. However it was shown by Karpinski and MacIntyre that if there are w weights in the network then

$$VCdim < O(w^4). \tag{6.6}$$

The Support Vector Machine

The support vector machine was introduced by Cores & Vapnik in [16] to make use of a certain class on hyperplanes with a useful structure defined on them. Let A be the, sum of the defining vector of the hyperplane when written in Vapnik's "canonical form". Let R be the radius of the smallest sphere bounding the input vectors and let n be the dimension of the space being classified. Then

$$VCdim = min(A^2R^2, n) + 1. \tag{6.7}$$

This example is to show that VC dimension can be significantly lower than the number of free parameters in the systems. The result is a significant improvement on (6.5) for small A and R.

The Sine Classifier

It is worth taking a brief look at a classifier with an infinite VC dimension. For such a machine the generalisation bound is also infinite, and it has been shown by

Vapnik that such a classifier can fit any training data, in some locality , with any of the possible interpolations.

The classifier consists of the set of functions of the form

$$x \mapsto sign(sin \alpha x), \;\; \alpha \in [0, \infty].$$

This classifier maps the real line to $[0, 1]$ and has a single free parameter α. However for any l the set

$$\{10^{-i}\}_{i=1}^{l}$$

can be shattered. For each possible set of outputs

$$y_1, y_2, \ldots, y_l, \;\; y_i \in \{0, 1\}$$

it is only necessary to set

$$\alpha = \pi.(\sum_{i=1}^{l}(1 - y_i)10^i + 1).$$

(For more details see Vapnik, [58, page 78].) The result of this is that

$$VCdim = \infty \tag{6.8}$$

so the sine classifier can never be expected to generalise well, no matter how big the training set nor how low its error on it.

Chapter 7

The N-tuple Classifier as a Statistical Learning Machine

The results presented in the previous chapters comprise an analysis of n-tuple classifier learning from the single viewpoint of statistical learning theory. The aim of this chapter is to place these results in the context of previous work, both in learning theory and in the study of n-tuple machines. This allows us to see how the particular problems of the n-tuple method are echoed in more general learning theory terms, and perhaps to see how such problems may be overcome. The chapter can be divided into three main parts: a summary of the problems encountered when using the n-tuple method, a review of the analogous problems in theories of learning and finally a synthesis of the two which illustrates how existing methods for improving n-tuple classifier performance are described by learning theory and offers suggestions from learning theory as to how they may be improved.

7.1 Optimising the N-Tuple Classifier: A Review

In most previous work, the difficulties associated with the classifier have been considered from the system point of view. That is to say the architecture – in particular the size of the n-tuples – and the training algorithm, have been analysed separately in an attempt to maximise the accuracy of the classifier. From the point of view of learning theory this is not the crucial distinction. The two important factors are the behaviour of the learning machine on the training set and its generalisation to the whole input space. Whether these two are optimised by appropriate choice of n-tuple size or training algorithm or both is in not really relevant; it is the performance of the machine on the learning problem alone which is important. In this section the structural (in the architectural sense), algorithmic and other attempts to optimise classifier performance are considered. In section 7.3 their effect in terms of learning theory will be analysed.

7.1.1 Structure: The Size of the N-tuple

As the first example of an optimisation problem let us consider the old staple – size of the n-tuple. This most fundamental parameter is well known to be crucial to the perennial problem of learning machines: balancing error on the training set against generalisation. The effect of trade-off can be stated informally as follows

> It has been found empirically that for a given size of training set there is an optimum value for the size of the n-tuples which will give maximum performance; smaller samples causing overgeneralisation [false positives] and larger samples undergeneralisation [false negatives].
> *Aleksander and Stonham, 1979* [4]

Naturally various attempts have been made to measure the effect of the n-tuple size. Generally these attempts take the form of experimental evidence and plots of error rate against n. The work of Ullman was among the first to consider this difficulty. In [56] he considers the generalisation error on a hand-writing recognition problem for various n. The results he obtains show the same form as that expected from a learning theory analysis: the error rate at first decreases with increasing n-tuple size, reaches a minimum and increases. The minimum error rate is attained for higher values of n if the training set is larger, as predicted. Later work [52], [63] made use of these "Ullman" curves to try and optimise the value of n for particular applications, but systematic theoretic analysis has been rare. The difficulties associated with setting this parameter are also discussed by Rohwer and Lamb in [44] and Rohwer and Morciniec in [45]. The former claim that bigger choices of n

give better performance until "very large" sizes are reached, although "a value of 8 is generally enough". The latter suggest that this may be due to the fact that 8th order correlations contain enough information to solve the learning problem for most data sets. They also suggest that it is useful to keep the proportion of written memory locations "neither too high nor too low". They also point out that highly skewed input data may result in one discriminator having too many stored 1s while the other has too few. In this case simply adjusting n cannot correct the problem. Indeed, as we shall see in section 7.3, the value of n is constrained by a number of considerations, not all of which can be simultaneously satisfied in all situations. However, many researchers have found that the choice of a value of n requires a trade-off at some point if good generalisation is to be achieved.

7.1.2 Structure: The Range of the Stored Values

Perhaps the first attempt to extend the power of the n-tuple method by extending the range of values that could be held in each address was made by Bledsoe and Bisson in [11]. In this scheme unlimited values could be stored and the whole was trained by an algorithm based on maximum likelihood. The results reported by Ullman in [56]for this version of the classifier were generally worse than for the original method.

Sixsmith, Tattershall and Rollet in [50] consider the n-tuple classifier as an approximation to the Parzen windows technique in non-parametric statistics. However the clipping effect caused by the fact that each discriminator has a limit to its maximum score (ie. N) means that patterns similar in Hamming space yield Parzen kernels which overlap but whose joint output is not double either individual kernel. Their solution was to use Bayes' Theorem to set the RAM locations which must therefore take real values. The results obtained with this methods in speech recognition experiments were a slight improvement over those obtained with the original system, not in terms of accuracy but rather in terms of the difference between the output of the correct discriminator and the output of the second.

A related way of extending the output range was applied by Goutos for alphanumeric character recognition in [22]. He sets a *memory element threshold*, an integer t, such that a memory location is only set to 1 during training if that location is accessed at least t times during training. This is equivalent to having an infinite output range and an output function incorporating a threshold. The idea behind it is to decrease saturation due to noise or rogue data. The value of t is a tunable parameter, which does not alter the VC dimension of the classifier, but rather the effectiveness of the training algorithm. Some a priori knowledge is required to set a value of t so that noise is filtered out while important data are not discarded.

The classifiers $\mathcal{F}_i$ proposed in chapter 5 also fall into the category of expanded storage range machines. The only difference between these machines and those above is that they are designed to give an output after thresholding. They aim to define a structure (in the statistical learning theory sense) that includes the standard n-tuple classifier as a low-dimensional element.

7.1.3 Structure: The Connection Graph

As Aleksander and Stonham point out in [4], the number of possible training graphs is astronomical, about 10^{1000} for a 256 bit input pattern. Consequently any means of selecting between them will require either a lot of time, a very subtle algorithm, or a willingness to accept a sub-optimal solution. This last was the case for Aleksander and Stonham whose main concern was that they should not have accidentally picked a graph that was unusually bad for the data they were processing. Hence their method consisted initially of training the classifier and testing the data for ten different connection graphs. In order to make better use of the connections which are obtained, points which do not vary in value between any of the training patterns can be removed from the input field. It is also suggested that this process may be extended to entire n-tuples which will therefore give no classification benefit. With large systems this is unlikely to be of much help in general. However it suggested that after training, if the two classes are only just separable, that an advantage may be gained by removing common n-tuples from the discriminators.

More ambitious was the work of Christiansen et al. [25] who developed a cross-validation scheme for selecting the best connections. The method was one of random selection followed by removal of poor connections according to an information-based measured termed*cogentropy* which is designed to assess the suitability of a particular classifier. As a result the algorithm requires many passes to find a solution. However, the empirical results given in the paper suggest that the effort was worthwhile in improving the error rate.

7.1.4 The Training Algorithm

It was claimed in chapter 2 that the purpose of the training algorithm in the general learning problem had to be to minimise empirical risk. However if the training algorithm is formulated appropriately it is possible to implement structural risk minimisation (see 3.4) simultaneously. That is to say, a training algorithm may be used to restrict the set of available functions to one with a low VC dimension and only increase the set of functions when it is no longer possible to maintain a sufficiently low error on the training set.

The original training method of Bledsoe and Browning is still the most widely used and the one against which the others are generally compared. Its benefits of simplicity and speed are always cited and since its results are usually acceptable many using the method feel no urgent need to change it. Moreover as long as saturation is avoided, the algorithm has advantages when the data consists of a number of clusters in Hamming distance. However, modifications and wholesale changes have been made by many users of RAM-based systems. Perhaps the first was the maximum likelihood training of Bledsoe and Bisson in [11] and mentioned in 7.1.2. As described above, in this method the output range was allowed to be infinite and the stored values were set by a maximum likelihood method. However the results obtained in this way in a comparison by Ullman [56] were not encouraging, although the analysis was more amenable to traditional statistical techniques. Other training algorithms using the idea of maximum likelihood are [57], [50] and [9]. An attempt was made by Morciniec and Rohwer in [35] to circumvent the inherent problems of using the n-tuple classifier as a maximum likelihood classifier, most notably the problem of dealing with n-tuples which have never been seen (the 'zero-tally problem'). They used another statistical formulation which they termed the Good-Turing Estimate (GTE) to improve the estimation for low tallies and introduced a smoothed tally to deal with gaps in the training data. Their empirical results however did not find that an algorithm based on the GTE gave them a significant benefit.

More recently the method of Tarling and Rohwer [53], first mentioned in section 2.4.1, which was designed to reduce the problem of saturation whereby so many locations in the RAMs are filled that almost all test patterns yield a raw value of N in each discriminator. Their idea was simply to test each new training pattern before training it. If the classifier already classified it correctly then it would be ignored, if not then the pattern would be trained in the usual way. It was found in the experiments performed in [53] that three passes through the training data were sufficient to train all patterns accurately. The same experiments showed that this method brought a decrease in the level of saturation and and increase in generalisation accuracy over an identical experiment performed with the original algorithm.

A further training algorithm was developed in this thesis: the SMA, see section 6.4.2. The inspiration behind this algorithm is that, given a learning machine, the best approach to learning a completely unknown distribution is to minimise the error of the machine on the training set. Other algorithms, notably the original one, fail to do this, especially when saturation occurs. This algorithm was used for estimation of VC dimension but has also been applied to real-world data from ECGs, provided by Manintveld, [32]and discussed in section 6.4.6. In this case the test error was about 5% with the original algorithm and 50% with the SMA. However in both cases the error on the training set was zero. We shall return to this example after the discussion of the problems of learning in the next section.

7.1.5 Preprocessing

If the network can't solve the problem, change the problem. Most data when presented are already processed or pre-processed in whatever way yields the best results. Preprocessing is sometimes discussed in relation to its effect on the performance of the n-tuple classifier and this illuminates some important properties of the classifier. A very thorough discussion in this vein is perhaps that of Rohwer and Morciniec in [45] in which real-valued vector data must be converted into the bit patterns in which the n-tuple classifier deals. The approximation of the n-tuple classifier to a nearest-neighbour (or k-nearest neighbour) look-up table is examined, and a preprocessing which as far as possible maintains Hamming distance relationships which mirror the Euclidean distance relationships of the real-valued data. Rohwer and Morciniec then discuss the successes and failures of the classifier on various data sets in terms of the parameters used in the preprocessing. It is clear from this discussion that the preprocessing is crucial in getting good generalisation. However, the *raison d'etre* of neural networks is to learn to classify any data with little or no knowledge of the underlying model. The validity of the above statement is discussed in the next section in relation to preprocessing and all the other methods used for tweaking the performance of learning machines.

7.1.6 A Summary of N-Tuple Classifier Techniques

This section has given the briefest of overviews of those techniques which have been applied to the basic n-tuple classifier to either improve its performance or occasionally to improve its analytical tractability. The point of this list is to understand the success or failure of these variations according to an analysis using the framework of statistical learning theory.

7.2 The Problem for Learning Machines

When learning machines were introduced in chapter 2 and learning theory in chapter 3, the problem of learning from examples was presented formally and shown to be solvable in most cases, at least most of those cases involving real data and popular learning machines. However in real world situations it is not always possible to generate the amount of training data required to guarantee successful generalisation according to the ERM and SRM analyses of Vapnik and Chervonenkis. In this section other methods for analysing learning are set out, and it is shown how the basic problems of learning do not depend on the framework in which it they are analysed. Consequently it may be necessary to alter the problem slightly and produce "data-dependent" methods which are sub-optimal in general but give much improved results for particular learning problems. These are also considered briefly.

7.2.1 The Bias/Variance Dilemma

The title of this subsection is taken from a paper by Geman et al. [20] in which the problems of learning machines are presented in the general theory of non-parametric statistics. The problems considered in the paper are essentially the same as those considered by Vapnik in [58] and I shall make use of both approaches to explain the problem.

The problem concerns the trade-off described by Vapnik and elucidated in section 3.3.3 between fitting the machine to the training data and knowing, within certain probabilistic parameters, that the generalisation will be good. Vapnik shows that the expected risk is bounded by the sum of the empirical risk on the training data and a confidence interval determined by the amount of training data and the complexity of the learning machine. Geman describes the trade off in more traditional statistical terms as being between the bias and the variance of the estimator. His formulation is as follows. Given an unknown mapping from $X \rightarrow Y$, a learning machine $\mathcal{L}$ and a data sample $\mathbf{z}$ of length l he obtains

$$\begin{aligned} E[\mathcal{L}(x,\mathbf{z}) - E[y|x])^2] &= (E[\mathcal{L}(x,\mathbf{z})] - E[y|x])^2 && \text{"\textit{bias}"} \\ &+ E[\mathcal{L}(x,\mathbf{z}) - E[(\mathcal{L}(x,\mathbf{z}))^2] && \text{"\textit{variance}".} \end{aligned}$$

The first summand, the "bias" term, represents the systematic error made by the learning machine. If, on average, the learning machine differs from the the expected output for any input (that is, the *regression function*) it is said to be a *biased estimator* for that sample **z**. The extent of this bias is measured by the first term. Suppose however that the bias is zero, so on average the classifier from the learning machine fits the observed data sample **z**. However the distance from the correct regression function for any particular data set could be wide. Geman gives the example of a linear interpolator which is used to solve a complex, noisy problem. Although the bias will be zero, the variance will be high if the noise level is high because many data points will be far from the regression function, and fitting the classifier to them will be of little benefit for the correct classification. Hence the actual (mean-squared) error will be large.

This decomposition exactly mirrors 3.3.3 where the bias is more generally the empirical risk and the variance term is a confidence interval based on the complexity of the machine. The main difference is that Geman's expression is a precise decomposition of expected mean squared error, while Vapnik's is more general, but is an *upper bound* on the expected risk. What is most important however, is that the trade-off between minimising empirical risk/minimising the bias of the estimator and minimising the confidence interval/minimising the variance of the estimator. Indeed, other frameworks for fitting a model to unknown data display the same dichotomy.

Geman goes on to consider the bounds on generalisation error, both theoretically and in a number of simulated examples. The conclusion he reaches is that if nothing is known about the data being modelled, and non-parametric techniques such as neural networks are being used to do the modelling, then unfeasibly large training sets are required to minimise simultaneously the bias and the variance. Two possibilities for getting round this problem present themselves: either large amounts of data must be available so that bias and variance *can* be minimised together, or the learning machine is chosen so that it is already biased towards the correct data. The former is analogous to following the SRM principle and providing the amount of data required by the VC dimension bound that applies to the optimal element of the structure, while the latter is equivalent to knowing enough about the data to choose an appropriate structure that is expected to model the data well. A technique for formalising this choice within the SRM framework is described in section 7.2.4.

7.2.2 Minimum Description Length

The minimum description length principle was put forward by Rissanen in [42] as a means of bounding the error of a learning machine by considering the *algorithmic complexity* of the relationship between the input and output parts of the training sample. The idea of algorithmic complexity was considered first by Solomonoff and refined by Kolmogorov who used it to define the randomness of a string. His idea was that the randomness of a string depends on the amount by which the string may be compressed. He defined the algorithmic complexity of a string to be the size of the shortest computer program that can generate the string and showed that this size was invariant up to an additive constant between computers. Hence high algorithmic complexity implies low compressibility which is associated with high randomness.

The process of bounding the error involves considering a table of input/output relations and considering how much the training data can be compressed by referring to this table. This *coefficient of compression* can then be shown, by a coarsening of the bound for the SRM principle, to give a bound on the expectation of making an error with table T with confidence $1 - \eta$ of

$$R(T) < 2(K(T)\ln 2 - \ln\frac{\eta}{l})$$

where l is the training set size and $K(T)$ is the coefficient of compression of the training data using T. Note that although this bound is worse than the standard bound in 6.1.1 it depends only on η, l and the coefficient of compression. No information about the number of tables available, the number of errors made by the table T, nor how the tables and code-books were organised is used in this bound. This shows the power of the MDL principle, but also that the SRM principle (by using more information) gives better error bounds.

7.2.3 Regularisation and Bayesian Methods

Regularisation is the name given to the set of techniques, originally developed in statistics, for solving "ill-posed problems": that is, problems for which a sequence of approximations to the correct solution do not necessarily converge to the correct solution. In particular, the problem of estimating a regression function - or even an unknown density function - from sparse data is ill-posed. It was shown by Tichonov, [54], [55],that minimal expected risk was obtained by minimising not the empirical risk, but a functional of the form

$$R^*(\alpha) := R_{emp} + \gamma(\delta)\Omega(\alpha)$$

where γ is a constant and Ω some function, both chosen according to the problem to be solved.

In the realm of neural networks the regularisation term may correspond to the sum of the weights in a multi-layer perceptron or some other structural parameter. In this way regularisation theory provides a method for adapting the "structural" parameters of a learning machine to minimise expected risk.

The theory of statistical inference that is usually used to explain and justify the use of a regularisation term is *Bayesian statistics.* In Bayesian theories of model selection a prior probability distribution $P(\alpha)$ over the set of all possible classifiers is assumed. Once the training data, $[Y, X] := (y_1, x_1), \ldots, (y_l, x_l)$ is available the regression function (or at least the parameter α that defines it) can be estimated by applying Bayes' rule

$$P(\alpha|[Y, X]) = \frac{P([Y, X]|\alpha)P(\alpha)}{P([Y, X])}.$$

While several methods exist for approximating the regression given this expression, let us consider the case where α^* is chosen to maximise the conditional probability with respect to the data. If, for example, noise is distributed normally over the data the value of α required minimises

$$\Phi(\alpha) := \sum_{i-1}^{l} \ln P(y_i - f(x_i, \alpha)) - \frac{2\sigma^2}{l} \ln P(\alpha).$$

This is in a form immediately recognisable from regularisation theory: the empirical error plus a regularisation term.

This regularisation information comes in the Bayesian model from the assumptions underlying the choice of prior distribution. In the case of neural networks or radial basis function classifiers in which the model parameters are real-valued, the prior is often taken to be a Gaussian distribution. Because non-smooth output functions require a wide distribution of weights, this has the effect of favouring smooth solutions: the same aim as many pre-processing and regularisation techniques. Most Bayesian methods yield a result that could be construed as a regularisation method, and at the same time provide a theoretical basis for using it. In this sense both methods, Bayes' rule and regularisation, can be seen as trying to minimise empirical risk simultaneously with another term. This other term can usually be seen as representing some structural parameters of the learning machine in which case the analogies with SRM become clear. Most importantly it is important to have enough data to continue the minimisation until the regularisation term becomes small and this again leads us to the problem of small data sets.

Another concept offered by the Bayesian approach is that of *evidence*. This is the probability of the training data given a particular model (or model parameter, α), ie. $P([X,Y]|\alpha)$. This also can be estimated in the learning process and the evidence for the data from a range of models can be compared. However the evidence need not be well correlated with the expected error since the two quantities simply denote different things. Empirical work has been done by MacKay [31] which suggests that the two quantities are correlated, but in certain cases they certainly are not.
Strictly speaking, a Bayesian learning machine must include the regression function exactly in its supply of hypotheses. If not, its prior probability will be zero and by Bayes' rule, so will its posterior probability. However, careful selection of the a priori distribution can ensure that this is not a problem. Moreover, it is claimed by Neil in [37] that good selection of the priors can ensure that the Bayesian method always produces optimal generalisation, whatever the complexity of the model. However,

such a "hands-on" approach to the selection of the function set is not consistent with the principle of no a priori knowledge that this has been held so far in this thesis. The results of chapter 3 show that optimal generalisation is certainly a function of model complexity unless external knowledge about the problem is known. For this reason the Bayesian formalism, and the method of regularisation, have more in common with the methods in the section below in which information about the data can be used to improve classification performance than with the data independent analyses of the ERM and the SRM principle.

7.2.4 Data-Dependent Methods

Of course there are as many data-dependent methods for minimising error are there are data sets, and this thesis has been mainly concerned with considering the bare-bones learning problem without knowledge of the underlying model. However, given the size of the error bounds found by such people as Vapnik and Geman, it is worth considering the role of the data itself, both to improve performance and to explain experimental successes.

Effective VC Dimension

Effective VC dimension was described in section 6.4 as a way of improving VC dimension bounds in cases where the distribution of the input is known. In the case of the n-tuple classifier it has been seen that the improvement in the generalisation bounds can be substantial. A problem with EVC dimensions is that they are often difficult to calculate theoretically, but the experimental methods of chapter 6 can be used to obtain an estimate. Thus if the input data from a known distribution or if there is sufficient training data, experiments similar to those in section 6.4.3 may be performed. This will give an indication of the effective VC dimension of the classifier on that input distribution which will allow tighter generalisation bounds to be made.

Unluckiness

Introduced by Shawe-Taylor et al. in [49], the concept of *unluckiness* is used to encode a priori assumptions about the training data and how well it will be fitted by the classifier. The training data itself is then used to choose a learning machine from a structure with a bias towards the elements of the structure with lower VC dimension that are expected to fit the data well. This allows the error bounds for the SRM principle to be improved on average (assuming the bias has been selected well). This is an example of "designing bias" as recommended by Geman et al..

The "High Art of Engineering"

What Vapnik has called "the high art of engineering" (see [58, page 157]) has undoubtedly been the most common method of choosing appropriate learning machines for learning problems. Either trial and error over various values of structural parameters or "heuristics" based on knowledge of the data or experience of previous problems have allowed results to be obtained which are far better than they would be if know external knowledge were introduced into the problem. Concepts such as effective VC dimension and unluckiness are attempts to formalise this engineering process and allow theoretical predictions to be made based on all the information available to the user of the learning machine.

7.3 The N-tuple Classifier Explained

The title of this section is in deference to the similarly named book by Dennett [19] which, in the same way as this section, analyses every aspect of the object of interest but never quite pins it down. The first two sections of this chapter have described in turn the particulars of the n-tuple method and the generalities of solving problems with learning machines and non-parametric methods. The aim of this section is to complete the syllogism and explain as much as possible about the n-tuple classifier given the current state of the art in the study of learning machines.

Some of techniques used for training and optimising n-tuple classifiers have a clear explanation in terms of the SRM principle. Others, however, are not so obvious and may even seem to run counter to the principle. We shall examine these cases and see if by not heeding the principle they make assumptions about the data which could, if required, be more formally stated.

7.3.1 Empirical Risk Minimisers

We shall first consider the n-tuple classifier's performance as an empirical risk minimisation machine. For any particular architecture the best upper bound on expected risk (when the structure of the data is totally unknown) is obtained when the error on the training data is minimised, see 3.2.4. As has been mentioned before, neither the original nor the Tarling-Rohwer algorithms described in section 7.1.4 guarantee to minimise error on the training data. The possibility of saturation is always present because of the fact that the algorithms can replace 0s in the RAM locations with 1s but cannot do the reverse. The stochastic minimisation algorithm was intended to replace the older algorithms with a process which would actually try to reduce error on the training sample. Experiments were performed with identical classifiers trained on the same data with all three algorithms in turn. The error on the training sample is plotted against the number of training samples drawn from the uniform distribution (figure 7.1) and the Gaussian training set of section 6.4.4 (figure 7.2). It can be seen that there is better accuracy on the training sample when trained with the SMA. However the SMA takes far longer and many passes through the data to run.

When they introduced their new algorithm, Tarling and Rohwer showed that on their classification task their new algorithm gave both lower saturation and lower generalisation error and state that the error of the trained classifiers on the training set was zero. They do not say if the error on the training set was zero with the original algorithm, but given the size of the training set and the fact that overall accuracy was high we may assume that this probably was the case. Hence the Tarling-Rohwer results are not showing enhanced empirical risk minimisation so the benefit of their algorithm must lie elsewhere. Indeed, in practical work it is often

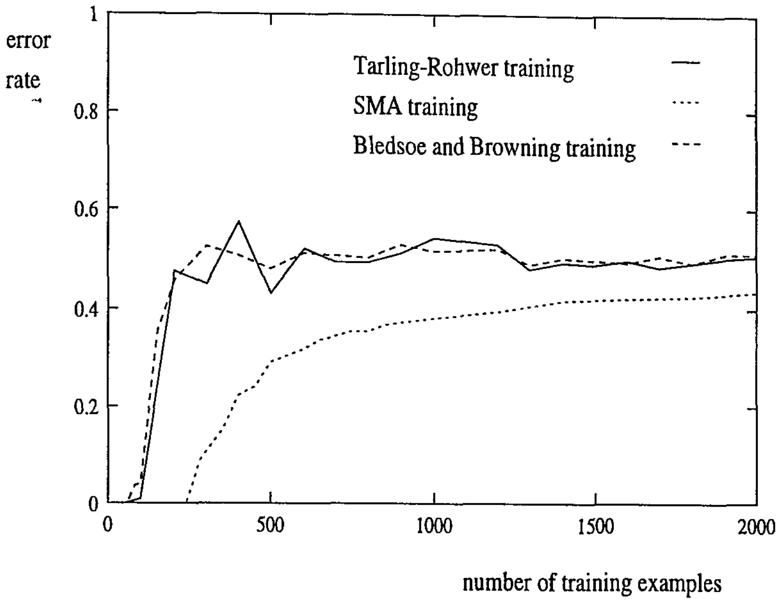


Figure 7.1: Error on the training set when trained with three different algorithms. Uniform distribution.

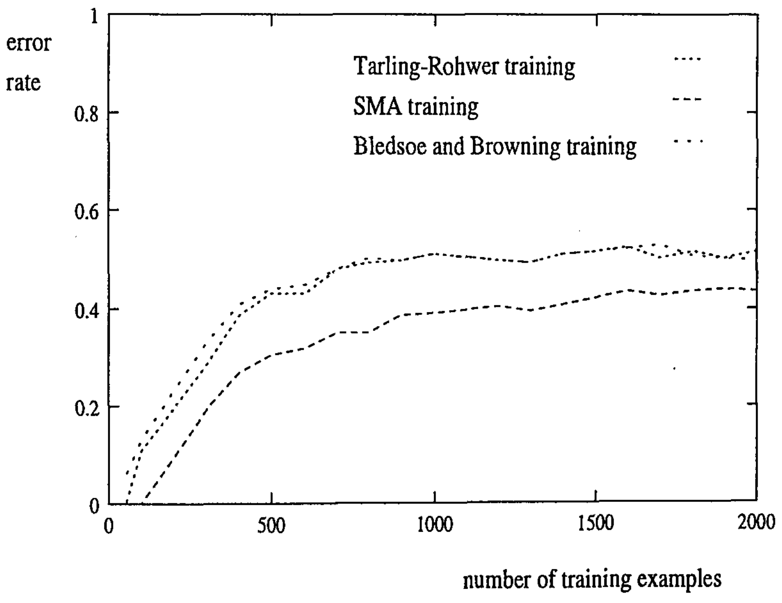


Figure 7.2: Error on the training set when trained with three different algorithms. Gaussian distribution.

the case that the empirical risk is zero. When this is the case the key to good generalisation lies in a small architecture (with small VC dimension) or else an inbuilt bias towards fitting the data. It will be argued that the latter is responsible for the success of the classifier in most of its practical applications.

Since the rest of the methods for optimising n-tuple classifier performance deal not with empirical risk minimisation but with choose the best architecture, the next subsection will consider the classifier in the light of the SRM principle.

7.3.2 Structural Risk Minimisers

All methods which modify the architecture to improve performance, but which do not make any assumptions about the structure of the data, can be considered in light of the SRM principle. This includes variation of the classifier's parameters and also the use of algorithms which tend to restrict the classifier to a subset of the class of functions that the classifier can theoretically realise. We shall first consider how certain structural considerations control generalisation by controlling the VC dimension and then look at the effect of algorithms.

Structural Control by Varying n

The n-tuple size controls the VC dimension and, in at least two cases, the effective VC dimension. Changing n is the most direct way of changing the number of functions realisable and the size of the maximal shatterable set. It is not possible to verify the VC dimension bounds empirically without testing the classifier on every possible data distribution. However the curves generated by Ullman [56] and those in chapter 6 show clearly the trade off between empirical error and over-fitting that SRM aims to solve. What is made clear in chapter 6 though is that although the VC dimension is governed by a term exponential in n, the effective VC dimension for common data distributions can be much lower than either $\log_2 3N(2^n - 1)$ (upper bound) or $N(2^n - 1)$ (lower bound). Because of this, the distribution-independent requirements on training set size for good generalisation are much higher than is commonly required. This is very much in line with Geman et al. above who claimed that truly non-parametric learning, which is equivalent to distribution-independent learning, will always require an number of examples which is larger than is usually feasible, or often necessary.

Another practical problem with the formal control of generalisation using n, is that there is no progression of n-tuples which yields a sequence of machines whose functionalities form an increasing sequence of the form

$$\mathcal{W}_{n_1} \subset \mathcal{W}_{n_2} \subset \cdots \subset \mathcal{W}_{n_i} \subset \ldots$$

where the sequence $\{n_i\}$ represents the sequence of increasing n-tuple size and the $\mathcal{W}_{n_i}$ are some n-tuple machines with a tuple size of n_i. It is fairly obvious that no such $\mathcal{W}_{n_i}$ exist for most $\{n_i\}$, but in fact even the case $n_i = 2^i$ (with n_i-tuples amalgamating to form $2n_i$-tuples) is not possible. Because of this limitation the SRM principle can only be informally applied to a set of n-tuple classifiers with strictly increasing n. Perhaps the easiest way around this problem if a structure based on n is desired is to add new RAM-nodes rather than replace ones with smaller values of n. Then the increasing sets of realisable hypotheses must be contained in each other, although the supports to the various will not all be of the same size.

Altering n may also have other, data-dependent, effects. In Rohwer and Morciniec [45], the pre-processing of real-valued data into binary codes may lead to problems as the metric induced by the CMAC/Gray encoding becomes non-linear for moderately high Hamming distances. Another problem can occur when the input pattern is large. If bits are not to be ignored, the product nN must be equal to the size of the input pattern. If only a simple machine is needed to classify the data, even the $n = 1$ classifier may have too high a VC dimension to solve the data from the training sample, even though the classification function (or at least the regression function) are realisable by the machine.

A real-world example of this is the performance of the n-tuple classifier on the ECG data used by Manintveld in [32] and mentioned in section 7.1.4. Since the full explanation of the results requires consideration of the training data it can be found in section 7.3.2.

Increasing the Range of Values

This subsection deals with those methods of controlling generalisation by increasing the range of stored values in the RAMs. The experiments of chapter 6 show that effective VC dimension increases with the range of values i for the distributions tested, although upper bounds have not been calculated theoretically for the distribution-independent case. Moreover, the classifiers with increasing output range do form a structure. Any hypothesis realisable using an output alphabet from 1 to i can certainly be realised if the output alphabet is extended to include $i + 1$. Hence there is a real possibility of applying SRM techniques. It will be necessary, however, to find the best rule for selecting the correct range of output values, and possibly, tuple size, and has not been attempted.

However this is for general data. In practice such systems have performed worse than similar classifiers with simpler output alphabets using Bledsoe and Browning training, eg. [11]. In the main these experiments have used far fewer data than the upper bound suggests and yet good performance is still obtained. This can only be because the classifier is a good model of the underlying data. In this case increasing the capacity can only be expected to harm the generalisation unless by chance it is a better model.

Maximum likelihood interpretations of the n-tuple classifier come across the same problems. If many of the components of the training examples occur only once, because of the paucity of training data, the analysis cannot be expected to work properly, see [35], [45].

The Role of the Training Algorithm in SRM

In certain cases the over-capacity caused by the size of the input space may be alleviated by the choice of algorithm. An obvious course of action given an over-sized input space is to preprocess the input data, if only by under-sampling it and ignoring some of the input bits. An alternative is to use an algorithm which does not fully search the space of functions of the classifier, at least not on all input distributions, but can be expected to contain the classification function or a close approximation.

The ECG data experiments of section 6.4.6 showed that in some circumstances the original training algorithm would out-perform an ERM algorithm (in this case SMA). Why should there be such a discrepancy? There are two reasons, of which only one strictly belongs in this section. The first is that the Bledsoe and Browning algorithm allows the classifier to model better the data. This is an example of designing bias as discussed in the next section. The second reason has to do with the limiting effect of saturation on the classifier. The SMA always managed to train the classifier with only a few bits set in the RAMs. This suggests that far more training data could have been fitted. This was much less true of the Bledsoe and Browning training algorithm. Many more training patterns, unless they shared all the same components, would have resulted in errors on the training set. This is because the effective search space of the original algorithm is smaller than that of the SMA. Although both algorithms can produce all hypotheses when given specially chosen training sets, as the training sets get larger the original algorithm tends towards those which are close to saturation. Thus the classifier returned by the algorithm is not necessarily a global optimum. Even if it is, it has been found without considering all the possible ways of fitting the training data. Although this argument is informal,it illustrates the effect of the algorithm on the control of generalisation.

It should be noted that similar effects are suggested to explain the performance of the multi-layer perceptron. Far fewer training patterns are needed to achieve good generalisation than even the tightest bounds seem to demand, and it is suggested (see [20]) that the existence of local minima into which the training machine can fall serve to limit the effective functionality, and hence the effective VC dimension of the machine. No formal method for controlling this effect has yet been put forward.

7.3.3 Designing Bias

According to Geman et al. in [20]

> ... the bias/variance dilemma *can* be circumvented if one is willing to give up generality, that is, *purposefully* introduce bias. In this way variance can be eliminated, or significantly reduced. Of course, one must ensure that the bias is in fact harmless *for the problem at hand* ...

Theoretical frameworks for introducing bias into structural risk minimisation have been introduced in section7.2 in the form of effective VC dimension and unluckiness. Empirical methods for doing so were classed as the"the high art of engineering" and it is fair to say that all n-tuple classifier research has been of this kind. Which approach is the better one to take?

The answer, of course, depends on the context in which learning problem is found. Constraints external to the whole problem, such as time and resource limitations, have not been considered so far. There is no doubt that the n-tuple method can be implemented to run very rapidly with satisfactory performance and in such situations engineering constraints are paramount. In more relaxed conditions such as off-line data analysis or recognition of static images, it may be possible to make better use of theoretical predictions. At the current state of the art, the models based on theoretical predictions improve relative to highly engineered models the less a priori information is available. As more sophisticated data-dependent methods such as unluckiness appear it may be possible to tip the balance more in favour of theoretically guided approaches. For those interested in accurate problem solving with n-tuple classifiers, neural networks or non-parametric statistical models generally, this will be a Good Thing.

7.4 What Results from these Results?

Statistical learning theory is in the first instance concerned with estimating expected error on a training set with information derived only from the training set. For a given learning machine the ERM principle is the best way to do this and the training algorithm should attempt to implement this principle. However a learning machine can be decomposed into increasing subsets of functions, and by considering these subsets in increasing order of size, it may be possible to pick a function with acceptably low training error from a subset of a small enough size that the probability of good generalisation is higher. This is the trade-off at the heart of the learning theory problem.

These increasing subsets of functions will certainly increase in cardinality, but most crucially they will increase in complexity. It is this complexity, as measured by the growth function and approximated with the VC dimension that is responsible for the effectiveness or otherwise of a classifier. It is not necessarily related to the number of parameters of the system, see section 6.5 but in the case of the n-tuple discriminators it is very close. The precise result for the two discriminator n-tuple classifier is not available but the VC dimension is within a factor of $\log_2 3$ of the log of the number of functions.

Given a particular structure on a learning machine, and as before given known prior knowledge of the problem to be learnt, the SRM principle says that the training algorithm should choose a function from a structure so that the sum of the training error and the expected generalisation error is minimal. Thus it specifies and tries to resolve the trade-off. Two problems remain, namely how to define the structure and how to perform the minimisation. From the point of view of the SRM principle, however, these are just implementation details and should be easily solved. In practice these are important problems, but decomposition and analysis along the lines of the Unluckiness function (section 7.2.4) may provide a principled answer. There is one other detail, however, which tends to make the bare ERM and SRM principles unsuited for practical use, and this is the large number of training samples required to guarantee low error. For this reason methods which make some assumptions about the data are almost always applied in practise. In the words of Geman et al. in [20] these methods "design bias".

How such bias is to be introduced has been a theme of this chapter, but there is no general answer unless the nature of the a priori information is specified. Many neural networks make assumptions about the smoothness of the output and that allows regularisation methods to be employed. In chapter 6 it was assumed that the input distribution was known and the effective VC dimension was calculated taking this into account. For both the normal and the Gaussian distributions the effective VC dimension was significantly lower than the VC dimension. By its nature the EVC dimension must be lower than its distribution-independent analogue, but can we see a reason for the extent of the difference? Perhaps the best reason we have for expecting that the VC dimension will be too high for most input distribution is the specialised nature of the training sets developed in chapters 4 and 5. Only very few such large sets are shatterable, so if the input distribution is such that those sets appear with very low probability it is not unreasonable to expect a significantly lower EVC dimension.

The other way in which bias is systematically introduced into the n-tuple classifier is through the training algorithm. Although only very few possible node value sets are actually ruled out by the algorithm, the predisposition is to set node values to 1 – a predisposition which when there is too much training data, and which is too noisy, leads to saturation. This clearly suits some data sets because it is possible for the same classifier, with the same structural parameters, the same training set and the same error (zero) on the training data, to perform close to optimally well and maximally poorly on the same test set as in the case of the ECG data in section 7.3.2. In this case the algorithm does not split the learning machine up into a structure in a deterministic way, but it does make certain output functions more probable than others. It would be interesting to know which functions these are, although previous theoretical work and experimental successes suggest that functions which are smooth in Hamming space are most easily implemented.

7.4.1 Future Directions

Future work in this subject is dependent on two factors, namely progress in learning theory and progress in the analysis of weightless systems. The data-dependent methods described in section 7.2.4 such as the unluckiness function offer possible ways of encoding a priori knowledge in a way that makes it amenable to statistical learning theory techniques. This may help to quantify the expected amount of training data required to obtain a required level of generalisation in some particular case. Progress in the analysis of weightless systems could take the form of the mainly theoretical results in this thesis or a more case-based experimental approach as in [45]. Specific areas for further research will be highlighted in the after the review of the thesis in the next chapter.

Chapter 8

Conclusions and the Future for the N-tuple Classifier

This thesis has combined two strands of research, n-tuple classifiers and statistical learning theory. It is intended as exposition of the capabilities and shortcomings of the n-tuple method and as an attempt to place the problems facing this method in the context of a more general learning situation. In this way it has been possible to suggest ways of improving recognition performance while making clear which facets of performance (whether accuracy or generality) have to be sacrificed or at least traded off against each other.

8.1 Fulfilling the Aims of the Thesis

The stated aim of this thesis was to apply the methods of statistical learning theory to the study of RAM-based systems, and in particular the n-tuple classifier. The point of this was to obtain a measure of the performance of the classifiers under consideration which would allow them to be compared with other classification methods. The method chosen to begin this analysis was the calculation of the VC dimension for the various classifiers which also entailed a formalisation of the learning problem and of the classifiers so that they attempted to solve this problem. The aim was then to obtain bounds on generalisation error from the VC dimensions. Ways would then be sought to improve the bounds in cases where some knowledge of the problem domain was available. The following section outlines the work presented in the thesis towards these aims.

8.2 A Recap of the Thesis

8.2.1 Chapter 1

Chapter 1 served as an introduction to the aims and contents of the thesis and described the nature of the work as being an application of learning theory to weightless systems – something previously little attempted. It served as a quick preview of the aims and contents of the thesis.

8.2.2 Chapter 2

Chapter 2 gave a brief history and overview of learning machines emphasising the rather independent lineage of the n-tuple classifier. The term learning machines covers more than simply neural networks (and indeed some neural networks such as Hopfield nets hardly count as learning machines at all.) The invention of many learning machines was inspired by physical biological models, especially the function and structure of the neuron. The analysis of such systems has in general proceeded quite separately from the analysis of learning machines whose inspiration is drawn mainly from statistics. Methods in non-parametric statistics such as Parzen windows [10, pages 53] had similar aims to those of many neural networks, namely to model data when very little is known about the underlying structure of the data. Many neural network methods have close parallels in statistics and the distinction between neural nets which learn from examples and statistical methods which learn from examples is largely arbitrary or due to external factors such as "biological plausibility."

8.2.3 Chapter 3

Chapter 3 was devoted to an overview of the statistical learning theory. The formal learning problem was discussed and the theoretical basis for solving it was explained.

The growth function and VC dimension were described along with their relationship to generalisation error. The aim of this chapter was to describe the tools with which learning in n-tuple systems is analysed in this thesis. The importance of both the ERM and SRM principles is stressed. The limitations of the theory, especially in terms of the failure to consider known facts (prior knowledge) about the data, were emphasised. Analysis of weightless systems has only occasionally been performed from the point of view of statistical learning theory. De Souto uses the concept of VC dimension in [18] to show that the problem of training a PLN pyramid (see section 2.4.4) is NP-hard. VC dimension was also used by Penny and Stonham in [39] to predict the storage capacity of PLN pyramids. The results were inconclusive, partly because they rested on some untested assumptions about the relationship between functionality and VC dimension, and partly because they were only tested on the uniform input distribution.

This chapter stressed the numerical distribution-independent bounds on generalisation that can be derived from the VC dimension of a system. Although in real-life cases there exists more knowledge about the problem which can be used to improve performance, these bounds remain the baseline performance of the machine in the absence of all prior knowledge and for that reason are important.

8.2.4 Chapter 4

Exact theoretical calculation of the VC dimension has not previously been performed for weightless systems. Unfortunately, as with so many other learning machines the calculation is extremely difficult in all but the simplest case. The VC dimension of a single discriminator with a maximal threshold and no nodes with overlapping supports is calculated exactly. This is achieved by exhibiting sets which are shatterable and then by showing no sets of greater size can exist. This the first original contribution of the thesis. The results are then extended to the case where node supports may overlap. In general the result only consists of an upper and lower bound, but for the case where each support overlaps on exactly one input bit or not at all a precise value for the VC dimension is calculated. This result is then extended to general neural units acting as auto-associators. All these results are novel and represent a baseline for assessing the abilities of weightless systems as pattern recognisers which learn from examples.

8.2.5 Chapter 5

More complicated classifiers such as non-maximal threshold classifiers and two-discriminator classifiers are considered in chapter 5. Precise VC dimension bounds are very hard to calculate for such systems because of the large number of ways that the classifier can be trained to realise a particular dichotomy on a small input set. Thus the functional capacity of the classifiers is used to obtain an upper bound on the VC dimension. The number of possible node value sets is shown to be a good approximation of the number of available hypotheses and this approximation is used for the VC dimension bound.

It is also shown that the two-discriminator n-tuple classifier can be expressed as a single, thresholded discriminator with three possible stored values instead of just two. This simplifies some of the calculations and also offers a way of defining a structure on a set of weightless classifiers which is not easily done by other means.

8.2.6 Chapter 6

The generalisation bounds generated from the VC dimensions of the previous two chapters are novel, but not very tight in many real cases. This is illustrated by a brief look at some experimental evidence and a brief discussion on the looseness of the bounds. A quantity called the effective VC dimension is used to cope with the looseness due to lack of knowledge of the input distribution. This is calculated for several different n-tuple classifiers with varying parameters and for two different input distributions: uniform and Gaussian. This allows better a priori estimates of generalisation to be made and is all novel work. Finally, for sake of comparison, the VC dimensions of some other systems are given.

8.2.7 Chapter 7

The effective VC dimension makes use of some a priori information, but often this information takes forms other than knowledge about the distribution of the input patterns. This chapter discusses previous attempts to optimise or improve the performance of the classifier and reviews other techniques for studying learning machines. These techniques are used to explain those aspects of the previous n-tuple work which are not explained by statistical learning theory. Amongst other things the role of the training algorithm was compared with the ERM principle and the reasons for its success despite not adhering to this principle are considered. Statistical learning theory has not previously been applied so comprehensively to weightless systems so it is important to separate the difficulties due to lack of research from the intrinsic limitations of the method. More advanced methods of applying learning theory, such as *unluckiness* [49] are suggested as ways of overcoming the limitations.

The practical usefulness of the learning theory analysis is also discussed, both as it applied to weightless systems and to learning systems in general. The problem of learning from examples is hard in many senses (computational, statistical, mathematical) and success can only be expected after either a lot of examples or with a lot of inbuilt bias which represents a priori assumptions of the system. It is clear that the usefulness of the standard learning theory increases the less is known about the data. For these cases the bounds on required training set size are most useful.

8.2.8 Chapter 8

This chapter is a summary of the work done in the thesis along with a look at its originality and usefulness and some directions for future work.

8.3 Were the Aims Achieved?

The answer to the question in the heading is "Yes, up to a point". The learning machines were easily put into the form required to calculate VC dimension and in the simplest cases (the maximum threshold discriminator and the self-connected GNU) the calculation was performed successfully. In more complex cases the VC dimension was bounded within a factor of $\log_2 i$, where i represents the range of stored values. These results were sufficient to obtain bounds, albeit looser than preferable, on the generalisation error of the classifier as required.

To try to obtain tighter bounds, a variant of the VC dimension, known as the effective VC dimension was introduced. This quantity depends on both the learning machine and the distribution of input patterns and is never greater than the VC dimension. However it is possible to use it in place of the VC dimension in the error bounds if the input distribution is known. The effective VC dimension can be estimated experimentally, and estimates of it were made for various weightless classifiers under both uniform and Gaussian input distributions. These values were substantially lower than actual VC dimension values and so if the input distribution is known, or can be guessed, fewer samples are needed to guarantee good generalisation. Alternatively this means that for the same number of samples a more complex machine can be used which may make it possible to reduce the error on the training set, thereby decreasing overall error.

8.4 What Wasn't Done

A first omission was the lack of exact values for the VC dimensions of many of the n-tuple classifiers. It becomes very difficult to find such VC dimensions because the number of ways in which a classifier may be trained to perform a particular dichotomy increases rapidly as the threshold is moved from the extremes – 1 or N – to the middle. However this omission is not too worrying from a practical point of view since the bounds obtained by considering the functionality of the classifiers are fairly tight and only differ from the actual VC dimension by at most a multiplicative constant. Moreover the generalisation bounds obtained from the VC dimension are true for any input distribution while many real problems use much more ordered data. Hence a bound which can incorporate a priori knowledge is usually more useful than a tighter bound which is independent of the problem.

A second obvious failing is in the treatment of the Bledsoe and Browning training algorithm (the "standard training"). The error term in the generalisation bounds is expressed in two parts, an error from the training plus an error from incorrect generalisation. Bledsoe and Browning training does not in general attempt to minimise error on the training sample. That it often does give low training error is due to properties of the data. If the data is assumed to be entirely unknown then there is no statistical reason to use the B & B algorithm (practically, however, it's a lot faster). Hence the decision about which algorithm to use is data-dependent and, as in the case of Manintveld's ECG data, can even be crucial.

8.5 Directions for Future Work

Several research ideas spring almost immediately from this work, while others are more tangential but equally interesting. If we consider the theoretical work first, it would certainly be satisfying to derive analytically the VC dimension of the n-tuple classifier rather than to simply bound it. However if learning theory is to be usefully applied to weightless systems it must take into account the sort of assumptions make, often implicitly, when systems such as the n-tuple classifier are used. The distribution-independent results, although interesting theoretically, are not realistic from the point of someone wanting to use weightless systems to solve a practical problem. Formalisation of implicit assumptions, as is done for example when setting Bayesian priors or a regularisation term, is important both for weightless systems and for learning theory in particular. Only then can all the information available be used for inference. Incorporating prior knowledge into the statistical learning theory framework and obtaining that knowledge for weightless systems are the two key areas which should be researched.

Learning theory approaches to incorporating prior knowledge are only just emerging, but since other fields of statistics make use of it ("classical" stats use models which are assumed a priori up to just a small number of parameters) it is reasonable to hope that such work will be fruitful. In a similar way, the fairly large number of techniques used for "tweaking" the n-tuple classifier should give insight into how assumptions (about data clustering or noise tolerance) are made and allowed for. A formalisation may follow from an investigation of such work (such as that outlined in chapter 7). In particular explicit use of the SRM principle may be made, perhaps by defining a structure based on the $\mathcal{F}_i$ of section 5.6.

More immediately it is possible to calculate effective VC dimensions for other input distributions and maybe estimate the accuracy of the calculations. One could also ask how effective VC dimension changes with small changes to the distribution. This is important if some actual input data is not exactly the same as a distribution for which the effective VC dimension is known.

8.6 A Take Home Message

This thesis has presented a study of the learning problem as faced by weightless, n-tuple based systems. The tools of statistical learning theory have been applied to make a number of predictions about the learning capabilities of such machines. Where the theory has been unable to explain the experimental results other theories of statistical inference have been used to supply a qualitative explanation. Thus novel results which may be of use to the designer of a n-tuple system have been presented while a review of the tricks used when more information exists has also be given. The message is that the a system cannot be analysed until it is formalised, but complex assumptions about data are very often left implicit when learning machines are applied to a learning problem. Making these assumptions explicit and allowing the resulting bias to be built into the machine is what is required to avoid the bias/variance dilemma in a principled way.

Bibliography

[1] P.J.L. Adeodato and J.G. Taylor. Recurrent neural networks with pRAMS. In F. Fogelman-Soulie and P. Gallinari, editors, *Proc. ICANN95*, pages 607–611, 1995.

[2] M.A. Aizerman, E.M. Braverman, and L.I. Rozonoer. The Robbins-Monroe process and the method of potential functions. *Automation and Remote Control*, 28:1882–1885, 1965.

[3] A. Al-Alawi and J. Stonham. The functionality of a multi-layer Boolean network. *Electronics Letters*, 25(10):657–658, 1989.

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

[5] I. Aleksander, W.V. Thomas, and P.A. Bowden. WISARD: a radical step forward in image recognition. *Sensor Review*, pages 120–124, 1984.

[6] N.M. Allinson and A. Kolcz. Application of the CMAC encoding scheme in the n-tuple approximation network. *IEE Proc. on Comput. Digit. Tech.*, 141(3), 1993.

[7] N.M. Allinson and A. Kolcz. Enhanced n-tuple approximators. In N. Allinson, editor, *Proc. Weightless Neural Network Workshop*. University of York, 1993.

[8] J. Austin and T.J. Stonham. Distributed associative memory for use in scene analysis. *Image and Vision Computing*, 5(4):251–260, 1987.

[9] A. Badr. N-tuple classifier for ECG signals. In N. Allinson, editor, *Proc. of the Weightless Neural Network Workshop 93*, pages 29–32, 1993.

[10] C.M. Bishop. *Neural Networks for Pattern Recognition.* OUP, 1995.

[11] W.W. Bledsoe and C.L. Bisson. Improved memory matrices for the n-tuple recogntion method. In *IRE Joint Computer Conference,11*, pages 414–415, 1962.

[12] W.W. Bledsoe and L. Browning. Pattern recognition and reading by machine. In *Proc. Eastern Joint Computer Conf.*, pages 232–255, 1959.

[13] N.P. Bradshaw. An analysis of learning in weightless neural systems. Technical report, Imperial College, London, 1996.

[14] A.P. Braga and I. Aleksander. Geometrical treatment and statistical modelling of the distribution of patterns in the n-dimensional Boolean space. *Pattern Recognition Letters*, 16:507–515, 1995.

[15] A. Bryson, W. Denham, and S. Dreyfuss. Optimal programming problem with inequality constraints. *AIAA Journal*, 1:25–44, 1963.

[16] C. Cores and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[17] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals ans Systems*, 2:303–314, 1989.

[18] M.C.P. de Souto. Learning and generalization in pyramidal architectures. Technical report, Imperial College, 1995.

[19] D. Dennett. *Consciousness Explained.* MIT, 1992.

[20] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[21] F Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1992.

[22] G.P. Goutos. The classification for alpha-numeric characters with a random access memory system. Master's thesis, University of Kent, 1978.

[23] S. Grossberg. Adaptive pattern classification and universal encoding. *Biological Cybernetics*, 23:121–134,187–202, 1976.

[24] G. Howells, D.L. Bisset, and M.C. Fairhurst. A new paradigm for RAM-based neural networks. In D. Bisset, editor, *Proceedings of the Weightless Neural Network Workshop*, pages 11–16, September 1995.

[25] T.M. Jorgensen, S.S. Christensen, and C. Liisberg. Cross-validation and information measures for RAM based neural networks. In D. Bisset, editor, *Proc. Weightless Neural Network Workshop*, pages 87–92. University of Kent, 1995. (have:Igor's copy).

[26] W.K. Kan and I. Aleksander. A probabilistic logic neuron network for associative learning. In *Proceedings of the 1st IEEE International Conference on Neural Networks*, pages 541–548, 1987.

[27] P. Kanerva. *Sparse Distributed Memory.* MIT, 1988.

[28] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(59-69), 1982.

[29] Y. LeCun. Learning processes in an asymmetric threshold network. In *Disordered Sytems and Biological Organisations*, pages 233–240. Springer, 1986.

[30] W. Maass. Vapnik-Chervonenkis dimension of neural nets. Technical Report NC-TR-96-015, NeuroCOLT Technical Report Series, Jan 1996.

[31] J.D.C. MacKay. A practical Bayesian framework for back-propagation networks. *Neural Computation*, 4(3):448–472, 1992.

[32] W.A.Th Manintveld. Automated ECG analysis using weightless neural networks. Technical report, Imperial College, London, 1996. (Ir. thesis at TU Delft and IDIC at Imperial).

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

[34] D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine Learning, Neural and Statistical Classification.* Prentice-Hall, 1994.

[35] M. Morciniec and R. Rohwer. Good-Turing estimation for RAM based neural networks. In D. Bisset, editor, *Proc. Weightless Neural Network Workshop*, pages 93–98. University of Kent, 1995.

[36] C. Myers. Learning with delayed reinforcement in artificial neural networks. Phd thesis, Dept. of Elec. Eng., Imperial College, London, 1990.

[37] R.M. Neal. *Bayesian Learning for Neural Networks.* PhD thesis, University of Toronto, 1994.

[38] A.B.J. Novikoff. On convergence proofs on perceptrons. In *Proc. of the Symposium on the Mathematical Theory of Automata*, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

[39] W. Penny and T.J. Stonham. Storage capacity of multilayer Boolean neural networks. *Electronics Letters*, 29(15), July 1993.

[40] M.J.D. Powell. Radial basis functions for multivariate interpolation: A review. In *IMA Conference on Algorithms for the Approximation of Functions and Data*, pages 143–167, Shrivenham, UK, 1985. RMCS.

[41] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.F. Flannery. *Numerical Recipes in C.* CUP, 1988.

[42] J. Rissanen. Modelling by shortest data description. *Automation*, pages 465–471, 1978.

[43] H. Robbins and H. Monroe. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[44] R. Rohwer and A. Lamb. An exploration of the effect of super large n-tuples on single-layer RAMnets. In N Allinson, editor, *Proceedings of the weightless neural network Workshop*, pages 33–37. University of York, 1993. nohave.

[45] R. Rohwer and M. Morciniec. A theoretical and experimental account of n-tuple classifier performance. *Neural Computation*, 8:657–670, 1996.

[46] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, Washington D.C., 1962.

[47] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In *PDP*, pages 318–362. Bradford Books, Cambridge, MA, 1986.

[48] N. Sauer. On the densities of families of sets. *Journal of Combinatorial Theory (A)*, 13:145–147, 1972.

[49] J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, and M. Anthony. A framework for structural risk minimisation. Technical report, NeuroCOLT Technical Report, May 1996. NC-TR-96-032.

[50] M.J. Sixsmith, G.D. Tattershall, and J.M. Rollett. Speech recognition using n-tuple techniques. *Br Telecom J*, 8(2), April 1990.

[51] K. Steinbuch. Die lernmatrix. *Kybernetic*, 1:36–45, 1961.

[52] J. Stonham. *The classification of mass spectra with adaptive logic networks.* PhD thesis, University of Kent, 1974.

[53] R. Tarling and R. Rohwer. Efficient use of training data in the n-tuple recognition method. *Electronics Letters*, 29, No.24:2093–2094, 1993.

[54] A.N. Tikhonov. On solving ill-posed problems and the method of regularisation. *Doklady Akademii Nauk USSR*, 153:501–504, 1963.

[55] A.N. Tikhonov and V.Y. Arsenin. *Solution of ill-posed problems.* W.H.Winston, Washington D.C., 1977.

[56] J.R. Ullman. Experiments with the n-tuple method of pattern recognition. *IEEE Transactions on Computers*, pages 1135–1137, 1969.

[57] J.R. Ullman and P.A. Kidd. Recognition experiments with typed numerals from envelopes in the mail. *Pattern Recognition*, 1:273–289, 1969.

[58] V. Vapnik. *The Nature of Statistical Learning Theory.* Spinger-Verlag, 1995.

[59] V. Vapnik, E Levin, and Y LeCun. Measuring the VC-dimension of a learning machine. *Neural Computation*, 6:851–876, 1994.

[60] V.N. Vapnik and A.Ja. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Applications*, 16:264–280, 1971.

[61] V.N. Vapnik and A.Ja. Chervonenkis. The necessary and sufficient conditions for consistency of the method of empirical risk minimisation. *Pattern Recognition and Image Analysis*, 1(3):284–305, 1991.

[62] B. Widrow. Adaptive switching circuits. In *IRE WESCON Convention Record*, pages 96–104, 1960.

[63] B.A. Wilkie. *A stand-alone, high-resolution adaptive pattern recognition system.* PhD thesis, Brunel University, 1983.

[64] J. Wray and G.G.R. Green. Approximation theory and finite precision computing. *Neural Networks*, 8(1):31–37, 1995.

Appendix A

Identical Hypotheses from Different Node Value Sets

It was shown in chapter 5 that two different node value sets for a single two-valued discriminator give identical hypotheses if and only if they yield the same constant hypothesis. That is, either the hypothesis which always returns **1** or that which always returns **0**. As shown in chapter 5 the number of constant hypotheses is given by

$$\binom{N}{N-\Theta+1}2^{(N-\Theta+1)(2^n-1)}+\ldots\binom{N}{N-\Theta+k}2^{(N-\Theta+k)(2^n-1)}+\ldots\binom{N}{N}2^{N(2^n-1)}+$$
$$\binom{N}{\Theta}2^{\Theta(2^n-1)}+\ldots\binom{N}{\Theta+k}2^{(\Theta+k)(2^n-1)}+\ldots\binom{N}{N}2^{N(2^n-1)}.$$

Figure A.1 shows the logarithm of the proportion of such constant hypotheses among all hypotheses for various values of the threshold, Θ. It is clear that the proportion is miniscule and reaches a minimum when $\Theta = N/2$.

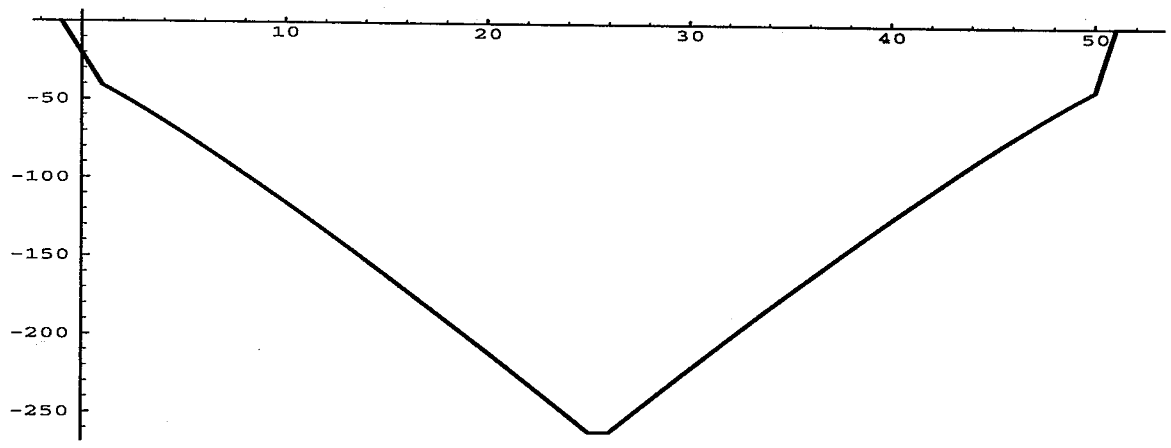


Figure A.1: The logarithm of the proportion of constant hypotheses for a $n = 4, N = 50$ discriminator thresholded at each value of Θ from 1 to 50.

Appendix B

Features and Relatedness in GNUs

Published in *Proceedings ICANN95*

B.1 Introduction

An important question for Neural State Machine[1] design is how to be sure that the trained states will give the kind of generalisation wanted. In this paper we consider the more general problem of an arbitrary training set, and measure the relationships between each input and output feature contained in it. We then measure how well they are preserved by a learning system. We assume the system has no prior knowledge of the training set and we define a function to determine how much information the training set alone contains about the relatedness between features. The use of this **relatedness function** is then extended to assess learning performance and to define certain problems. These terms are now defined precisely before the relatedness measure is proposed.

Let the input space, X, be $\{0,1\}^N$ and let the output space, Y, be $\{0,1\}^M$, where N and M are integers, large enough that the normal approximation for the distribution of patterns [2] is valid. The training set, T, is a subset of the Cartesian product $X \times Y$. If $(x, y) \in T$ then T **associates** x with y. Each $x \in X$ may only be associated with one $y \in Y$, so T is a partial function from X into Y. The learning system is supposed to extend T to a complete function $s : X \to Y$ in a principled way. The aim of this paper is to give a more precise notion of how the principles governing the preservation of related features may be formulated. To do this the idea of a feature pair is introduced.

An **input region** of size n is any of the $\binom{N}{n}$ subsets of X of the form $\{0,1\}^n$; it is identified by its associated projection function, $\pi_n^f : X \to \{0,1\}^n$. An **input feature**, f, is defined to be the ordered pair (π_n^f, x_f), that is, the pair containing a projection and some $x_f \in X$. The input region of f, $X - Ker(\pi_n^f)$, is denoted by $\overline{f}$. The **size** of the feature f is denoted by $|f|$ and equals the size of the region of f, that is, n. An **output feature** is defined analogously for Y. Let $F(X)$ be the set of all input features and let $F(Y)$ be the set of all output features. A **feature pair** is an ordered pair $(f, g) \in F(X) \times F(Y)$.

Almost all feature pairs will be irrelevant to the training set. It will be useful to be able to decide which pairs are relevant, and to get some measure of how relevant they are, given only the information in the training set. In this paper two functions from the set of feature pairs into the interval $[0, 1]$ are defined. One measures the relatedness of two features in the training set while the other measures the amount of information in the training set about the input feature.

B.2 The Relatedness Function

Suppose T, the training set, is finite with N_T members, $\{t_1, t_2, ..., t_{N_T}\}$ and each $t_i = (t_i^X, t_i^Y)$.

If f_1 and f_2 are defined on the same region, that is, their associated projection functions are the same, then define $s(f_1, f_2)$ to be the number of bits on which x_{f_1} and x_{f_2} match in the region $\overline{f_1}$ (which is the same as $\overline{f_2}$). Let $d(f_1, f_2)$ be the number of bits that differ ($= |f_1| - s(f_1, f_2)$). We call s the **feature similarity** and d the **feature distance**. We now give the formula for the relatedness of some given feature pair, (f, g).

Let $s_i^f := s(\pi^f(t_i^X), f)$ and let $s_i^g := s(\pi^g(t_i^Y), g)$. We also specify the properties required of the nearness functions, n_i, which measure similarity between features. (A particular choice of functions is given in the next section.) Let n_1 and n_2 be functions of two integer variables, monotonically increasing in the first, which take real values in the interval [0,1].

The relatedness between f and g according to the training set is then defined by

$$C_T : F(X) \times F(Y) \to [0, 1]$$

$$C_T(f, g) = \frac{\sum_{i=1}^{N_T} n_1(s_i^f, |f|).n_2(s_i^g, |g|)}{\sum_{i=1}^{N_T} n_1(s_i^f, |f|)}.$$

The confidence associated to each *input* feature is given by

$$conf : F(X) \to [0, 1]$$

$$conf(f) = \sum_{i=1}^{N_T} n_1(s_i^f, |f|)$$

Since the n_i are monotonically increasing, $n_1(s_i^f, |f|)$ is a measure of the closeness between f and the projection of the ith training example onto $\overline{f}$. By multiplying this with a similar function on the output space we get a measure of the linkage between f and g in that training pair. This is then summed over the whole of T and normalised by $conf(f)$, the "total nearness" of f with the training examples. Thus feature pairs linked strongly on a few training pairs, but for which the training data containing f is sparse, are still assigned high levels of relatedness.
The comprehensiveness of the training data is indicated by $conf(f)$.This gives a measure of how many times the input feature, f, or close approximations to it, occur in the training data.
The relatedness function differs importantly from covariance in several ways. For instance if the output feature under consideration is linked to input features other than the one under consideration, this does not affect the value of the function. Moreover, there is flexibility in the choice of nearness function function which allows different ideas of closeness of features to be used.

B.2.1 The Nearness Functions

The choice of the **nearness functions** n_i is crucial to the interpretation of the relatedness function. They depend only on the distance between features and the size of the features, not on the features themselves, so they have very little "knowledge" of the inputs.

Since no prior knowledge of the training set is assumed, the best model of the training data before it is presented is to assume all patterns to be equally probable. Hence the distances between features are expected to follow a binomial distribution which may be approximated by the normal [2]. For a relationship between features in a training pair to be significant, it must be significantly improbable in such a random distribution.

This suggests that nearness functions which take into account the difference from the expected values of feature nearness would be appropriate. That is, the n_i should be related to the cumulative distribution function for the distribution of distances of features in the feature region of f. This is approximately the normal distribution with mean $\frac{|f|}{2}$ and variance $\frac{|f|}{4}$. To ensure that the values fall in the required range (a feature at distance $\frac{|f|}{2}$ has probability $\frac{1}{2}$) the cdf is transformed by $x \mapsto 2x - 1$ and negative values are mapped to 0. Weight is then given to feature pairs according to the a priori expectation of their distances from the measured feature. Those which are "unusually" close are given high weighting while those at mean distance (and further) are ignored.

A nearness level below which a feature can be said to be insignificant may be stipulated. This is the **significance level.** Note that in a binary space this is equivalent to defining a Hamming radius around a feature outside which the feature is no longer said to be significant; this will be called the **significance radius.** The radius can be derived from the nearness and vice versa by simply considering the probability that a pattern picked at random should fall outside that radius. Similar calculations, and tables of values, can be found in [**2**].

The significance radius for a particular significance level also allow us to determine the minimum possible size for a feature. (For example a feature of size 1 will be found in 50% of a set of randomly chosen patterns, and so any non-zero significance level would make it always insignificant.) It is interesting to note that the minimum feature size depends solely on the significance level and the size of the feature, not on the size of the space containing the feature. However, any particular learning model (eg. the GNU) may have further constraints as to the size of feature it can hope to distinguish.

Note that other choices of nearness functions are possible. For instance they could give different weight to different parts of the feature. However, such a notion of nearness assumes some prior knowledge of the relative importance of the input and/or output bits which is not assumed in this paper.

B.2.2 Finding the *Real* Features

Given a particular training set, all but a few feature pairs have a relatedness of near zero. Moreover, of those with non-zero relatedness some will be contained in, contain or overlap other feature pairs with higher relatedness. How are our human conceptions of *real* features related to the very general definition of features above?

We have two choices: either to think of a *real* feature not as a single input/output pair, but rather as a less well-defined area of input/output space where similar features are correlated, or else to attempt to find feature pairs with locally maximum relatedness and think of these as the actual feature pairs in the data. It is not clear how such a local maximum should be defined.

B.2.3 Uses of the Relatedness Function

Extracting features from training sets is what a learning system is often required to do. The learning machine can be seen as a statistical model of the data which is trained on the (supposedly "real world") training data and which outputs a best guess on the rest of the input space. This is what we call generalisation.

There are two other ways the coefficient can be used: as a measure of the relatedness between input and output features in a trained system, and as a means of specifying a problem. The first case is similar to that in Section 2 except that the sum is taken over all possible input patterns, not just those in the training set, and the pattern in the output space is the pattern output by the system. A comparison between this and the values for the training set will give a measure of how well the system maintains links between associated features. Values calculated this way, from a trained system, will be referred to as C_{SYSTEM}, for example, C_{GNU}.

Specifying a problem by its C_T values inverts the analysis described above. Instead of trying to determine what sort of behaviour our training set should induce, we can try to discover what sort of training set will induce a specified behaviour. The pre-specified values of relatedness used to specify a problem will be referred to as C_P. In the remainder or the paper we shall discuss the generalisation behaviour of GNUs [**1**] in terms of C_P and C_{GNU}.

B.3 Brief Analysis of GNU behaviour

Here we look at how a simple classification problem may be specified in terms of its C_P values and how well the GNU can approximate the right answer from training data. We isolate three specific areas of difficulty for the GNU, and ways of overcoming these problems are put forward.

The problem is known as the LR problem. The input space is divided into two subsets, L and R. The system is required to output *TypeL* if and only if it receives an input which has one of some set of predefined features in L. The *TypeR* output is defined analogously, and if neither condition is fulfilled the required output is undefined. We have thus defined which feature pairs are to have $C_P = 1$. Further conditions are considered later. We assume that the training set and GNU connections are unknown.

Suppose r_c is the significance radius, and suppose f is some feature which is required to output *TypeL*. Let p_j be a pattern such that $s(\pi_f(p_j), f) = i$. To evaluate $C_{GNU}(f, \mathit{TypeL})$ we need to find the nearness of the output of p_j to *TypeL*. This depends on $|f|$, on the number of nodes nodes that output the same for both *TypeL* and *TypeR* and also on

Prob(A node given a pattern with a feature at nearness i to f outputs TypeL*).*

Only the last is unknown (until the training set and the GNU connections are specified) and the output nearness will be written simply as $n_2(Prob(...))$. Then the relatedness in the GNU between f and *TypeL* is evaluated as follows.

$$C_{GNU}(f, \mathit{TypeL}) = \frac{\sum_{i=|f|}^{|f|-r_c}[2^i.2^{(N-|f|)}.n_1(i,|f|).n_2(Prob(...))]}{\sum_{i=|f|}^{|f|-r_c}[2^i.2^{(N-|f|)}.n_1(i,|f|)]}.$$

Let us suppose initially that the GNU is fully connected without feedback and hence acts a look-up table, then the probability can be re-written as

Prob(A pattern with a feature at nearness i to f is nearer a TypeL *training input than a* TypeR*).*

This expression encapsulates a lot of what is "hard" about certain problems and training sets. If its value varies wildly for large i, the value of $C_{GNU}(f, \mathit{TypeL})$ will be very close to $C_{GNU}(f, \mathit{TypeR})$. Hence the correlations that the GNU can infer will be weak and flawed even if the feature has size N, that is, the feature covers the whole input space. These problems are not well-suited to solution by GNUs in their current form. The worst example of such a problem is the parity problem which alternates completely for each i. However, if when the problem is specified it is required that C_P decline smoothly within the significance radius, then this difficulty is avoided. (Although functions like parity cannot be specified with this stipulation unless the confidence radius is 0.)

The other way in which the problem can be hard is if the feature is small, and many exemplars of its class are very distant in Hamming terms, since they may differ on that part of the pattern which is not part of the feature. In this case even though a training example is close in one feature region, it is not close in the whole input space. The non-feature part of the pattern causes confusion. The GNU can only be expected to make a judgement on a small feature if that feature is strongly linked with the correct output feature in the training set.

If a training set is chosen such that both $C_T(f, \mathit{TypeL})$ and *conf(f)* are high, the training inputs containing close approximations to the feature mostly give the required output, and moreover there are lots of such training pairs in the set. In this case there will probably be a positive exemplar near to the test input, and the fully-connected GNU will find the correct output in most cases.

The memory and/or processing requirements of a fully-connected GNU increase exponentially in N, so let us now consider the case of the sparsely-connected feed-forward GNU [1]. Suppose, as is usual, that the GNU connections are randomly generated and fixed thereafter. Assume L and R are spatially distinct, so we can assume that a node x has l_x connections to the L part of the input space and r_x to the R. The total number of inputs per node is $l_x + r_x = k$ which is the same for each node.

In lieu of a full analysis, the obvious difficulties will be outlined. Clearly the two previous difficulties will occur, but compounded by the effect of partial sampling. Some of the output nodes will have the same output value for both sorts of input and these can be ignored. Of the others, some have opposite values for each category, while some are only defined for one or other of the outputs. It is not hard to see that those with opposite values would perform optimally when $l_x = r_x$, while those which only need to fire for, say, *TypeL*, will perform optimally for $l_x = k, r_x = 0$. It is possible, given a statistical distribution of the training pairs and the connections, to estimate the accuracy of the response to nearby patterns, as required for the C_{GNU} calculation.

Improvements have been suggested to the GNU system which aim to optimise performance by extending the system to more layers [3] or by adapting the connections according to the training data [4]. It will be possible to assess any improvements to the GNU's capability for feature-based generalisation by comparing the C_{SYSTEM} values obtained from training with given training sets and comparing them to the C_T values induced by those sets.

B.4 Conclusions

A function has been defined which expresses the relatedness between input and output features which need not span the whole input or output space. Three forms of the function have been proposed: to study training sets, to specify problems and to study trained systems. The capacity of learning systems to preserve feature correlations in training sets can be gauged by comparing C_T with C_{SYSTEM}. The capacity of a learning system to learn how to solve a problem from a set of examples can be gauged by comparing the required values of C_P with the actual values of C_{SYSTEM}.

B.5 References

[1] Aleksander and Morton,(1993) *Neurons and Symbols*, Chapman and Hall

[2] Kanerva,(1988) *Sparse Distributed Memory*, MIT Press

[3] Kan, (1990) *A Probabilistic Neural network for Associative Learning*; PhD Thesis, Imperial College, London

[4] Bradshaw(1995), *Weightless Systems: What, where, how, why?*; Internal Report, Imperial College London

Appendix C

The Future for Weightless Systems

Published in *HELNET International Workshop on Neural Networks, Proceedings Volumes I/II*, eds. M.J. van der Heyden, J.Mrsic-Flogel and K.Weigl.

C.1 Introduction

This paper presents an overview of some of the current problems and sticking points in weightless systems research. The aim is to highlight where progress will need to be made if weightless systems are to play a distinctive role in the future of neural systems research.

There are two main areas which should be examined more closely: firstly the imprecise nature of the use of terms such as "training“, "generalisation" and "learning" and secondly the lack of much adaptation of the systems to optimise them for any particular task. The suggestion in this paper is for a more formal, if not indeed mathematical, set of definitions governing our everyday terms for the capabilities of our systems, together with a more precise knowledge of how well they perform to these formal specifications. This will then allow us to properly assess the effectiveness of any new "learning" or "adaptive behaviour". The inspiration for much of this approach is from Computational Learning Theory[6]. To interpret GNUs and other weightless systems in COLT terms is a long-term goal.

While some of the comments refer to all weightless systems, they are more specifically aimed at the General Neural Unit and Neural State Machine of Aleksander [**9**]Note that most of the GNUs referred to are feed-forward only and are operating in single steps. There is no discussion of attractors. To analyse more complicated systems, the basic machinery must be properly understood.

C.2 Formalisation

In this paper we shall only *describe* the degree of rigour and formality with which undefined terms should be used. The rigorous theoretical treatment demanded will be given elsewhere. Examples are [2] and [**3**].

A good point to start is 'training'. What differentiates 'training' from 'programming' apart from the connotations placed upon them by workers in different fields?

This question is fundamental to any claim about trained machines. Could we, for example, refer to 'iconic programming' rather than 'iconic training' (see [9]) without altering the sense? In so far as the iconic paradigm defines exactly what internal representation should be stored for a particular input this is indeed an example of programming. More exactly it is a specification of the data structures that the processor, in this case the GNU, should use. It is training and not programming in so far as adding another iconic state transition to the GNU is not a clear declarative programming act. It depends on the previously trained transitions, and the nature of their interaction is largely opaque to the 'trainer'.

Of course, the fact that the training is iconic is not really the issue here; any one-time occurrence which is determined externally to the machine must be subject the same examination.

This is not to say that the GNUs are not being trained. Of course they are. The point is rather to look past the terms we commonly use to notice whether what we are doing is substantively different to what has gone before, that is to say, programming. If it is different, then in what way? Only by making this sort of analysis will we ever see in what way work with weightless neural systems is making an original contribution.

C.3 The Generalisation Game

Now what does generalisation mean? In its broadest sense we can examine the generalisation of some function $T : X \to Y$. Suppose $X \subset Z$, then any function $S : Z \to Y$ such that $S|_X = T$ is a generalisation of T. In a sense S fills in the gaps. But how? Whenever we feel we intuitively know what we mean by generalisation it is because we have an idea of our domain of interest (Z in the example), and of a pattern in the required function which seems in some sense obvious. Whenever we are designing a system to perform generalisation we must be aware of our hidden assumptions and deep understanding of the patterns in our data which may make a particular generalisation problem seem easier than it is.

The generalisation performed in most weightless systems is based on the same principle. The function T (ie. the training data) consists of point to point mappings (ie. input/output training pairs), and the underlying idea is to eliminate noise by recalling from an unknown input the output pattern corresponding to the trained input pattern that is nearest in Hamming distance. That is, nearest neighbour lookup (NNLU).

Now this is one generalisation function that GNUs have been shown to be capable of performing. Their performance at such tasks has been analysed, for example in [**10**]. To see what else they can usefully do, it is instructive (as a thought experiment) to substitute a perfect NNLU machine for a GNU in any proposed system. If the NNLU performs better in its position, then the usefulness of the GNU in that situation is clearly dependent simply on its approximation to a NNLU. In this case it has well-defined competitors such as Sparse Distributed Memory and a direct technical comparison can be made.

Of course the behaviour of a GNU is by no means exactly that of a NNLU machine. It is where these differences are of benefit that any GNU-specific research should be directed. The differences with the GNU as currently formulated stem from the fact that no node has more than a fractional view of the input space. This restricts the number of functions that a GNU can perform (parity for instance cannot be measured by a sparsely connected GNU), but also appears to give potential for more interesting generalisation. The "partial blindness" of GNUs forces them to make generalisations about the training data. The question is, are these generalisations the right ones?

C.4 What they *can't* do

In the section above it is claimed that a sparsely connected GNU cannot learn to solve the problem of the parity of its inputs. This is not in itself too worrying (it isn't the sort of problem *people* can solve easily, either) but it is worthwhile considering why it is the case, and precisely what sort of problem the sparse GNU cannot solve.

The easiest way to see that a non-fully-connected GNU cannot solve the parity problem is to consider the response of any one node. There are 2^{N-f} inputs which yield any given f-tuple at any particular node. Of these, half have zero parity and half have one. Thus there is no way for the node to select which to output. The power of the GNU to perform generalisation assumes that at least some of the nodes can make the assumption that, given a particular input f-tuple, one output is more likely than another. Thus this particular problem cannot be solved by any GNU, whatever its learning, training or spreading algorithm, unless at least one node takes input from the whole of the input space. The result is analogous to the equivalent result for linear perceptrons in [**1**]. The question for GNU research is, having given up the ability to solve parity problems, what have we gained?

This is not a terminal problem for GNUs. The parity problem and similar problems yield easily to other techniques (eg. counting) and do not occur regularly in the domains which have been selected for experiment with GNUs. Perceptron workers, although set back temporarily by the inability of the one-layer perceptron to solve parity [**1**], soon recovered their confidence, and learning to solve parity-like problems became a benchmark for new learning systems. Moreover, it is clear than a GNU has enough information to make the decision (if we assume each input pixel is connected to at least one node); it is the feed-forward nature of the GNU which makes it impossible. Could a GNU with feedback in principle succeed where the feed-forward net failed? This is a question for further research.

C.5 Feature Sensitivity

Again this needs a fairly formal definition if we are to talk meaningfully about it. If a machine is given nothing but a set of training examples, it will not be able to use any "deep principles" to analyse the data. Hence a measure of relatedness based purely on the training set has been proposed in [2] which associates a value in [0,1] to every pair of input and output sub-patterns. The stronger any such association is, the more sense it makes to refer to the two sub-patterns as features. Thus by "feature sensitivity" I am referring to the ability to detect such i/o correlations in the training data, and use them as a basis for generalisation. Whether or not this is the sort of generalisation behaviour required in a particular system is another question. It is just a way of extracting meaning from a set of training examples and as such is something that a learning machine could plausibly hope to discover.

This is traditionally a property more associated with weighted-sum type networks rather than weightless systems. There is, however, no reason not to investigate the performance of GNUs in feature-sensitive situations. The relatedness measure defined in [2] and expanded upon in [3] can be used to determine the relatedness between sub-patterns either on the basis of the patterns in the training set or the state function of a trained system. By comparing the two we can see how well the trained system retains the feature-sensitivity in the training set. Perhaps we can also find ways to enhance it.

C.6 Enhancement Speculation

A GNU is specified by just a few characteristics: the size of the input and output fields, the size of the feed-forward and feedback to each node, the exact distribution of the (usually randomly chosen) connections and (when trained) the contents of the VRAMs and the spreading radius. The training procedure already deals with setting the contents of the VRAMs, so we need not investigate that in such detail. The i/o sizes are a function of the problem to be solved, and varying the spreading radius does not fundamentally affect the GNU capabilities. A non-maximum spreading radius forces the GNU to use *less* information, so the decision to alter it must be external to the GNU system itself. It may in some cases prevent unwanted information from being used, but the functionality of the GNU is fundamentally the same. To upgrade the GNU's capability we are therefore left with the possibility of altering the number and distribution of the connections, or else extending the model in some way, possibly by adding extra layers of GNUs as in [**4**].

C.6.1 Ways and Means

Two possible ways to improve on random connections are presented next. Both, however, require some (preferably local) method of determining useful (ie information rich) lines and rejecting unhelpful poor lines.

Method I:

Start with a random system of connections of fixed size f. Use some kind of evolutionary process to remove poor lines and test other lines at random. This would be better if we could pick potentially useful lines rather than at random, but this is even harder.

Method II:

Instead of fixed f-tuple size we fix the total memory available for storage. The f-tuple size is then set to be the maximum possible that will fill the available memory. When training threatens to overflow the memory the f-tuple size for each node is reduced by removing the least helpful input line. This is highly dependent on order of training and does not allow for recovery of lines which later training makes useful again. Of course, we could initially train like this until some "base" f-tuple size is reached and then continue with the birth/death processes of Method I.

Extensions of the model to extra layers can have several uses. Kan trains with an algorithm which separates the images of trained patterns so that they interfere less. This is very useful for unrelated pattern recall, but unhelpful for feature sensitive data. It may well, however, be possible to devise an algorithm to deal sensitively with training examples which are correlated. A procedure could start by assigning a random pattern to each input pattern and adapting it according to the output and input data in the training set. Thus a pattern which made best use of the random connections could be interpolated between the input and output data with an extra layer of VRAMs to perform the mapping. Perhaps this could be done so as to give a generalisation process in some sense better?

C.6.2 Optimal Mappings and Connections

A useful thing to know would be, given a fixed set of f-tuples and connections, what are the "easiest" training sets for the GNU. That is, what is its optimal performance over all possible i/o mappings. We can see that the parity problem is the most difficult possible, and a constant map (where each VRAM always outputs either 1 or 0) is the easiest. Can we measure this sort of "easiness" and use it to convert "hard" representations of problems to "easier" ones? Then we could hope to come up with intermediate patterns which help us to approximate this best mapping at each level. We would need to find a set of patterns which, given the input set, form an "easy" output set but which form an "easy" input set given the output set, and preferably do this adaptively during training. There is no obvious reason to believe any such set could give performance above the average, but maybe ...

Another sort of extension is to add more parameters to the basic GNU. This has already been done in at least two ways. Aleksander et al. in [5] suggest a "discrimination coefficient" to modulate each input line to a G/VRAM according to an off-line analysis of the training set. This has the advantage of adapting the GNU to the training set and allowing each VRAM to become differentially sensitive to its input lines, thus allowing the GNU to be feature sensitive. An extension to this work has been proposed by Ruijterman [8] who uses the algorithm to analyse medical data. Sales [7] has implemented a system by which the spreading of values in the VRAMs varies over the input fields. These both provide cases for analysis.

The important point about any of these extensions and innovations is not how they perform on a few convenient training sets, but how they match up to some formal criteria for learning and generalisation. Their limitations, and hence their domains of applicability must be discovered if the method is going to survive into the future.

C.6.3 Statistical Learning Theory

One candidate for the formalisation of weightless systems is the statistical learning theory of Vapnik and others [**11**]. This formulates the problem of learning from examples in precise terms and offers standard statistical definitions of learning and generalisation. Moreover it provides a framework for the evaluation and comparison of different learning machines. Whenever the learning problem can be precisely stated learning theory methods are applicable, so in most useful cases the theory can be applied. Work to measure the complexity and generalisation ability of certain weightless systems is ongoing.

C.7 Conclusions

There are no conclusions in this paper, only suggestions and speculations. The suggestion made is that without a good look at the real computational capability, and indeed cost, of weightless systems it will be impossible to systematically improve them except for some narrow class of problems. Moreover, unless we decide exactly why we want to use a weightless system for a particular task we cannot properly assess its performance against competitors. Do we see the GNU as a quick-training associative memory in which we may implement a state machine, or does its sparse connectivity allow it a wider scope?

The speculation took the form of some possible extensions to the GNU functionality which may or may not help to solve some of the problems which may or may not be of interest to those working in the field. They attempt to bring some adaptivity into the model to allow it to respond better to a wider range of problems. Of course, similar schemes have been put forward before, what is important is to properly understand and measure what function we want the GNU to perform and measure the improvement with these extensions. If we don't do this, and as formally as possible, we risk having a raft of plausible examples but no general facts about what our systems can do. It is for this reason that the application of statistical learning theory methods to weightless systems may be fruitful. An early attempt to do this is contained in [12] but more analysis is needed.

I hope that this will inspire people to think about the underlying nature of the power of weightless systems, or, more importantly, to formalise and make known what they have been thinking about for a while. Without some taking stock, research in weightless systems could slowly stagnate.

C.8 References

[1] Minsky and Papert,(1969) *Perceptrons*; MIT Press

[2] N. Bradshaw,(1995) *Feature Sensitivity in GNUs*; Proceedings of ICANN'95.

[3] N. Bradshaw,(1995) *Solving Problems with GNUs*; Internal report, Neural systems Group, Imperial College London.

[4] W.K. Kan,(1990) *A Probabilistic Neural Network for Associative Learning*; PhD Thesis, Imperial College London.

[5] Aleksander et al.,(1994) *Binary Neural Systems*; Internal Report, Neural Systems Group, Imperial College London.

[6] Kearns and Vazirani,(1994) *An Introduction to Computational Learning Theory*; MIT Press, 1994.

[7] Nick Sales; Personal Communication.

[8] C.L.M. Ruijterman,(1995) *The CLMR Generalization Algorithm*; Internal Report, Neural Systems Group, Imperial College London.

[9] Aleksander and Morton,(1993) *Neurons and Symbols*; Chapman and Hall

[10] Wong and Sherrington,(1988) *Storage Properties of Randomly Connected Boolean Networks for Associative Memory*; Europhys. Lett., 7 (3), pp. 197-202

[11] V. Vapnik, (1995) *The Nature of Statistical Learning Theory*; Springer-Verlag.

[12] N. Bradshaw, (1996) *A Learning Theory Approach to Studying the N-Tuple Classifier*; Submitted to ICANN96.

Appendix D

Improving the Generalisation of the N-Tuple Classifier using the Effective VC Dimension

Accepted for publication in *Electronics Letters.*

Abstract The VC dimension of an n-tuple classifier predicts poorer generalisation performance than is found in practice. The effective VC dimension of the classifier for a Gaussian input distribution is calculated empirically and shown to be significantly lower than the VC dimension. Thus better generalisation can be predicted without risking over-fitting.

Introduction: N-tuple classifiers of the type pioneered by Bledsoe and Browning [12] and developed by, amongst others, Aleksander and Stonham [4] and Rohwer [45] and have found significant practical application owing to their simplicity and the speed with which they operate in either hardware or software.

The Vapnik-Chervonenkis dimension is a scalar quantity which contains information about the functional complexity of a learning machine. It is possible to derive probabilistic bounds on the expected generalisation error of a learning machine on a problem if the size of the training set and the VC dimension are known. In order to make these bounds tighter in those cases where something is known about the distribution of the input patterns, a second quantity called the *effective* VC dimension is used.

In this paper the lower bound on the size of the training set required to give good generalisation is found to be unrealistically high if based on the the VC dimension. The effective VC dimension is estimated empirically for a Gaussian distribution of input patterns and shown to be significantly lower than the true VC dimension. This reduces the bound on the minimum number of training examples required for generalisation, and allows the system designer to make a better choice of system parameters.

The N-Tuple Classifier: The fundamental unit of weightless systems is the RAM node which is functionally equivalent to a random access memory capable of storing single binary digits. A binary input pattern is presented at the input and is sampled

in N groups of n bits. Each group or *n-tuple* is fed into a single RAM node which either writes or reads at the addressed location depending on whether the classifier is in the training or the test phase. A discriminator consists of N RAM nodes and a device for summing their outputs. Each discriminator (and although there can be any number this paper only considers the case where there are two) has an associated class label. In this paper they are taken to be logical **1** and **0**. Several training methods have been suggested but none affect the VC dimension. For this paper the Stochastic Minimisation Algorithm (SMA) has been developed to find the set of stored values that yield minimum number of misclassifications of the training data. It is described in [13].
Testing requires that the input pattern be presented to each discriminator, whereupon the locations addressed at each node by the n-bit samples of the pattern are read. The output from each RAM node in a discriminator is summed and the output of the classifier is the class label of the discriminator with the highest score. (In the case of a tie the output may be selected by an arbitrary rule.) The speed and simplicity of both training and testing are the classifier's biggest advantages in practical situations.

The VC Dimension: The VC dimension is a scalar property of any set of binary valued functions: for example the set of functions realisable by a two-discriminator n-tuple classifier. Let us say that a set of size l of input patterns is **shattered** by a set of functions if those functions can produce all 2^l dichotomies of the input set. The **VC dimension** of a set of function $\mathcal{L}$ is defined to be *the size of the largest input set that can be shattered by* $\mathcal{L}$. The definition can be extended to functions with a larger output range, but for simplicity we shall only consider binary-valued functions here. Hence we shall only consider classifiers with two discriminators.
The importance of the VC dimension is due to the following bound, (D.1), on generalisation error by Vapnik and Chervonenkis [58]. Given completely unknown input and output distributions, a randomly drawn training sample of size l, a classifier with VC dimension h and a proportion R_{emp} of misclassifications on the training data, then with probability η the expected classification error R can be bounded as

follows.

$$R \leq R_{emp} + \frac{1}{2}\sqrt{4\frac{h(\ln\frac{l}{h}+1) - \ln(\frac{\eta}{4})}{l}}. \tag{D.1}$$

This leads to a trade-off between the first term, error on the training set, and the second, error due to poor generalisation. Both depend on the complexity of the training set, but in different senses. An optimal classifier for a problem must have both sufficient VC dimension and sufficient training data.

The VC Dimension of the N-Tuple Classifier: In previous theoretical work, [13], the VC dimension, $h_{\mathcal{D}}$, of the two discriminator n-tuple classifier has been bounded by

$$N(2^n - 1) \leq h_{\mathcal{D}} \leq (\log_2 3)N2^n. \tag{D.2}$$

For (D.1) to guarantee acceptable generalisation, the size of the training sample must at least be of the order of the VC dimension of the classifier used. Even at the lower end of this bound this would require far more examples than are commonly used.

The problem with the VC dimension result is that because no assumptions are made about the distribution of the data, all the information about the problem must be deduced from the training examples. The *effective VC dimension* defined by Vapnik et al. in [59] takes into account some of the properties of the input distribution (if they are known) and can be used to replace the VC dimension in (D.1). The same paper gives an empirical method for calculating the effective VC dimension of a classifier in which the maximal deviation of the errors on two half-samples must be calculated. This method has been applied to the n-tuple classifier.

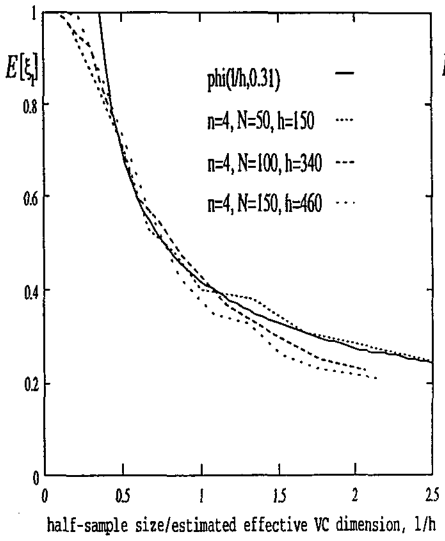
A simulation was designed in which random training sets could be generated according to a Gaussian distribution (of Hamming distance from a fixed point). Each output value was set to 1 or 0 with 50% probability. The training error was then approximately minimised by the SMA and the resulting error converted to a maximal half-sample error deviation as in [59]. These values were then plotted against half-sample size. According to the theory this graph can be approximated by an analytic function, phi(l/h), which depends on the effective VC dimension and the

n	N	VCDupper	VCDlower	EVCdim
4	50	750	1190	150
4	100	1500	2380	340
4	150	2250	3570	460
6	50	3150	4990	700
8	50	12750	20,200	2000

Table D.1: VC dimension lower bound, VC dimension upper bound and effective (Gaussian) VC dimension of n-tuple classifier.

half-sample size. By these means an estimate of the effective VC dimension can be derived.

Experimental Results First it was shown that the choice of assignment of output values did not affect half-sample error deviation. These are shown in Figures 1 and 2. Then the expected deviation $E[\xi_l]$ was estimated for different values of l, different classifiers and for both input distributions. The best-fit effective VC dimensions were estimated and the graphs of $E[\xi_l]$ against l/h were shown to fit with the theoretical predictions justifying the approximate estimated values of VC dimensions. These plots are shown in Figure 2. The discrepancies for the larger values of l may be due to the failure of the SMA to find the global minimum of the error.



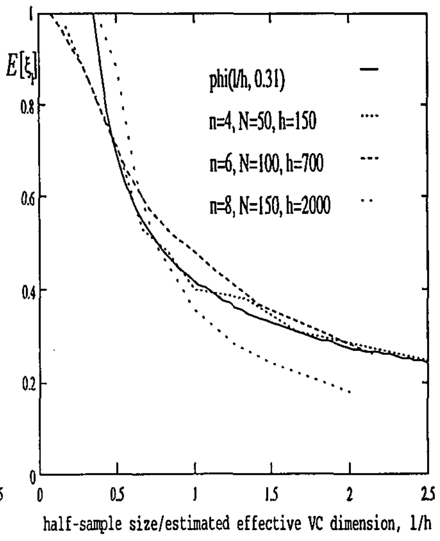


Figure D.1: Max half-sample error against l for three different output distributions.

Figure D.2: Empirical $E[\xi_l]$ against estimated effective VC dim over l plotted against a best-fit estimate.