CLASSIFICATION OF COMPLEX TWO-DIMENSIONAL IMAGES IN A PARALLEL DISTRIBUTED PROCESSING ARCHITECTURE

ROBERT GILMOUR SIMPSON, B.A., M.Sc.

A thesis submitted in partial fulfilment of the requirements of the
University of Plymouth for the degree of Doctor of Philosophy

October 1992

University of Plymouth in collaboration with the Plymouth Marine Laboratory

REFERENCE ONLY

90 0166561 3

UNIVERSITY OF PLYMOUTH
LIBRARY SERVICES

Item No. 900 1665613

Class No. T. 006.33 SIM

Contl No. X702778566

LIBRARY STORE

UNIVERSITY OF PLYMOUTH
LIBRARY SERVICES

To all my family, for their encouragement.

ROBERT GILMOUR SIMPSON

Abstract.

Neural network analysis is proposed and evaluated as a method of analysis of marine biological data, specifically images of plankton specimens. The quantification of the various plankton species is of great scientific importance, from modelling global climatic change to predicting the economic effects of toxic red tides. A preliminary evaluation of the neural network technique is made by the development of a back-propagation system that successfully learns to distinguish between two co-occurring morphologically similar species from the North Atlantic Ocean, namely *Ceratium arcticum* and *C. longipes*. Various techniques are developed to handle the indeterminately labelled source data, pre-process the images and successfully train the networks. An analysis of the network solutions is made, and some consideration given to how the system might be extended.

Acknowledgements. page 1
Author's declaration. page 2

Chapter 1. Neural Networks. page 3

1.1 Introduction page 3
1.2 Image Processing page 3
1.3 Parallel Distributed Processing page 7
  1.3.1 Parallel processing page 7
  1.3.2 Parallel distributed processing page 8
    1.3.2.1 Nodes page 10
    1.3.2.2 Connections page 12
    1.3.2.3 Network functions page 12
1.4 Neural Network Implementations page 15
  1.4.1 The artificial neuron page 15
  1.4.2 The perceptron page 16
    1.4.2.1 Rosenblatt's formulation page 16
    1.4.2.2 Minsky and Papert's criticism of the perceptron page 20
  1.4.3 Multiple constraint satisfaction page 23
  1.4.4 Multi-layer networks page 25
    1.4.4.2 Back-propagation page 26
1.5 Summary page 30

Chapter 2. Classification. page 31

2.1 Introduction page 31
2.2 Classification page 31
2.3 Theories of Categorisation. page 33
  2.3.1 The classical model page 34
    2.3.1.1 Defining features page 34
    2.3.1.2 Implementing classical theory page 35
    2.3.1.3 Failures of the classical model. page 37
  2.3.2 Classes of categories. page 39
  2.3.3 Probabilistic models of categorisation. page 40
    2.3.3.1 Prototype models page 40
    2.3.3.2 Exemplar models page 41
    2.3.3.3 Feature list models. page 42
    2.3.3.4 Type and token representation in neural networks page 44
  2.3.4 Similarity of individuals. page 46
    2.3.4.1 Similarity theory page 46
    2.3.4.2 The Ugly Duckling theorem page 48
  2.3.5 Psychological measurement. page 51
  2.3.6 Formal models of categorisation. page 53
    2.3.6.1 Bayesian theory page 53
    2.3.6.2 Fisher's discriminant page 56
    2.3.6.3 Least Mean Squared error method page 57
    2.3.6.4 Nearest Neighbour methods page 58
  2.3.7 Modelling a classification scheme. page 62
2.4 Summary page 64

Chapter 3. Rationales. page 65

3.1 Introduction page 65
3.2 Previous Work in the Domain page 65
  3.2.1 Traditional analytical techniques page 66
  3.2.2 Automation of morphometric analysis page 68
    3.2.2.1 Segmentation of the image page 68
    3.2.2.2 Classifying the single specimen page 69

3.2.3 Other automated techniques .... page 70
3.2.4 Summary of previous work in the domain .... page 71
3.3 Network algorithm selection .... page 72
3.3.1 Unsupervised networks .... page 73
3.3.1.1 The competitive network .... page 73
3.3.1.2 Experiments with an ART 1 model .... page 75
3.3.2 Supervised networks .... page 78
3.3.2.1 The back-propagation algorithm .... page 79
3.3.2.2 Feed-forward networks .... page 80
3.4 Selection of pre-processing .... page 80
3.4.1 The need for pre-processing .... page 80
3.4.2 Pre-processing techniques .... page 81
3.4.3 Spatial frequency trials .... page 82
3.4.3.1 Results .... page 84
3.4.4 Using Fourier information .... page 85
3.5 Summary .... page 86

Chapter 4. The Plankton Data Set. .... page 87

4.1 Introduction .... page 87
4.2 The Problem Domain - Plankton .... page 87
4.2.1 Ceratium plankton .... page 87
4.3 Assessing the Data .... page 89
4.3.1 Ground truth in classifying data .... page 89
4.3.2 Obtaining expert judgement .... page 91
4.3.3 Assessing expert judgement .... page 92
4.4 Pre-processing the Data .... page 96
4.4.1 Source of the images .... page 96
4.4.2 Problems with source data .... page 97
4.4.3 Obtaining homogeneous source data .... page 98
4.4.4 Pre-processing the source images .... page 99
4.4.4.1 Fourier analysis .... page 100
4.4.4.2 Processing frequency information .... page 102
4.5 Summary .... page 103

Chapter 5. Methodologies. .... page 104

5.1 Introduction .... page 104
5.2 Back-propagation Networks .... page 104
5.2.1 The back-propagation algorithm .... page 104
5.2.1.1 Variations on vanilla .... page 106
5.3 Training the Network .... page 107
5.3.1 Selecting training and test sets .... page 107
5.3.2 Training the network .... page 109
5.3.3 Network maturity; error, weights and epochs .... page 110
5.3.3.1 Error of a network .... page 110
5.3.3.2 Change in network weights .... page 112
5.3.3.3 Number of training epochs .... page 112
5.4 Testing the Network .... page 114
5.4.1 Multiple experimentation .... page 114
5.4.2 Correlation .... page 114
5.4.3 Extrapolation .... page 117
5.4.4 Regression analysis .... page 118
5.4.5 Probability measures of performance .... page 119
5.4.5.1 Generalisation .... page 119
5.4.5.2 Quality of generalisation .... page 121
5.4.5.3 Performance probabilities .... page 122
5.4.5.4 A proposed performance metric .... page 124
5.5 Summary .... page 125

Chapter 6. Initial Network Tests. .... page 126

- 6.1 Introduction .... page 126
- 6.2 Implementing Back-Propagation .... page 126
  - 6.2.1 Verifying the simulation .... page 126
    - *6.2.1.1 The Exclusive-Or function .... page 127*
    - *6.2.1.2 Simulating a network .... page 127*
    - *6.2.1.3 Data compression .... page 130*
- 6.3 Parameter Experiments .... page 131
  - 6.3.1 Training set size experiment .... page 132
  - 6.3.2 Training pattern size experiment .... page 133
  - 6.3.3 Input size experiment .... page 134
  - 6.3.4 Signal/noise ratio experiment .... page 135
  - 6.3.5 Within- and between- class correlation .... page 136
- 6.4 Summary .... page 138

Chapter 7. Ceratium Trials. .... page 139

- 7.1 Introduction .... page 139
- 7.2 Ceratium Learning Trials .... page 139
  - 7.2.1 Raw frequency experiments .... page 139
    - *7.2.1.1 Raw frequency data .... page 139*
    - *7.2.1.2 Raw frequency results .... page 141*
  - 7.2.2 Frequency gradient experiments .... page 143
    - *7.2.2.1 Frequency gradient .... page 143*
    - *7.2.2.2 Frequency gradient results .... page 144*
- 7.3 Network Generalisation .... page 146
  - 7.3.1 Classification in the normal condition .... page 146
    - *7.3.1.1 Normal results .... page 146*
  - 7.3.2 Pattern noise experiments .... page 148
    - *7.3.2.1 Additive pattern noise .... page 148*
    - *7.3.2.2 Additive pattern noise results .... page 149*
  - 7.3.3 Random pattern sequence presentation .... page 151
    - *7.3.3.1 Training set order .... page 151*
    - *7.3.3.2 Training set order results .... page 151*
  - 7.3.4 Combining pattern noise and sequence order .... page 152
    - *7.3.4.1 Order and noise results .... page 152*
  - 7.3.5 Limiting receptive fields .... page 153
    - *7.3.5.1 Receptive fields .... page 153*
    - *7.3.5.2 Receptive fields results .... page 154*
- 7.4 Summary .... page 155

Chapter 8. Network Analysis. .... page 156

- 8.1 Introduction .... page 156
- 8.2 Solution Analysis .... page 156
  - 8.2.1 Assessing generalisation .... page 156
  - 8.2.2 Performance characteristics .... page 158
  - 8.2.3 Analysis of network solutions .... page 162
  - 8.2.4 Feature detection .... page 163
    - *8.2.4.1 Fully-connected networks .... page 163*
    - *8.2.4.2 Limited receptive fields .... page 164*
  - 8.2.5 Hidden node discrimination .... page 165
  - 8.2.6 Weight symmetry .... page 167
  - 8.2.7 Arbitration of multiple solutions .... page 168
  - 8.2.8 Nearest Neighbour trials .... page 171
- 8.3 Summary .... page 173

Chapter 9. Discussion. .... page 175

- 9.1 Introduction .... page 175

9.2 Ceratia Classification .... page 176
    9.2.1 Applying neural networks .... page 176
    9.2.2 Network classification .... page 177
    9.2.3 The human case .... page 178
    9.2.4 Assessing network success .... page 179
    9.2.5 Networks versus Neighbours .... page 181
9.3 Network Observation .... page 182
    9.3.1 Network observation .... page 182
9.4 Improving Performance .... page 183
    9.4.1 Selecting training data .... page 183
    9.4.2 Arbitration .... page 184
    9.4.3 Default classes .... page 184
    9.4.4 Building large classifiers .... page 185
9.5 Conclusion .... page 186

Appendix I - Pop-11 code. .... **page 188**

Appendix II - Plankton specimen images and task instruction sheet .... **page 195**

References. .... **page 215**

List of Illustrations and Tables.

List of Illustrations

Figure 1: a–d Activation functions. 11

Figure 2: Logical functions, (a) NOT A, (b) A AND B. 15

Figure 3: An elementary perceptron with n A-units and m R-units. 18

Figure 4: An elementary perceptron, 22

Figure 5: Sigmoid functions $y = 1 / (1+e^{-\eta x})$, 26

Figure 6: A feed-forward network architecture. 27

Figure 7: a) A network with a threshold value in the node function. b) A equivalent network, but with the threshold considered as an additional input (bias). 28

Figure 8: A simple universe, consisting of two starting predicates, big and dark. 47

Figure 9: An Adaptive Resonance Network. 76

Figure 10: Binary images of *Ceratium* plankton; a) *C. longipes*, b) *C. arcticum*. 91

Figure 11: Numbers of images eliciting a minimum consensus. 95

Figure 12: a) A grey-level image. b) Power spectrum of a). 101

Figure 13: a) Low frequency image of Figure 12a. b) Frequency histogram of a. 102

Figure 14: RMS curves above, and the weight dynamics below. 113

Figure 15: The probability performance metric vs. number of trials. 125

Figure 16: RMS error in a network on the EOR problem. 128

Figure 17: Weight change in a network on the EOR problem. 129

Figure 18: Weight and RMS error change in a network on the EOR problem. 129

Figure 19: A feed-forward data compression network, with $n = 3$. 130

Figure 20: Learning data-compression, for $n = 1, 2, \ldots 5$. 131

Figure 21: Trials with increasing size of training set: a) $\eta = 0.5$, b) $\eta = 0.2$. 133

Figure 22: Trials with increasing length of input patterns. 134

Figure 23: Trials with increasing gain of input patterns. 135

Figure 24: Trials with increasing addition of noise. 136

Figure 25: Trials with increasing within-class correlation. 138

Figure 26: a) Grey-level images of (top) *C. longipes* and (bottom) *C. arcticum*, b) Reconstructed images of a) using lowest 16 frequencies. 140

Figure 27: A feed-forward network with 16 input units, 3 hidden units and 2 output units. 140

Figure 28: RMS error of a network on a *Ceratium* frequency trial. 141

Figure 29: Classification error of a network on a *Ceratium* frequency trial. 142

Figure 30: A feed-forward network with 15 input units, 3 hidden units and 2 output units. 144

Figure 31: RMS and classification errors of a network on a *Ceratium* frequency gradient trial: a) 0–1000 epochs b) 0–20 epochs. 144

Figure 32: RMS error of 10 networks on a *Ceratium* frequency gradient trial. 145

Figure 33: RMS error of 54 networks on *Ceratium* frequency gradient trials. 147

Figure 34: Classification error of 54 networks on *Ceratium* frequency gradient trials. 147

Figure 35: Generalisation error of 54 networks on *Ceratium* frequency gradient trials. 148

Figure 36: Generalisation error of 54 networks on *Ceratium* trials with added noise. 150

Figure 37: Generalisation error of 54 networks on *Ceratium* trials with random pattern presentation order. 151

Figure 38: Generalisation error of 100 networks on *Ceratium* trials with random pattern presentation order and added noise. 152

Figure 39: A feed-forward network with 15 input units, 3 hidden units and 2 output units. 153

Figure 40: Generalisation error of 100 networks on *Ceratium* trials with limited receptive fields. 154

Figure 41: Performance error of 100 networks on *Ceratium* trials with random pattern presentation order and added noise, 157

Figure 42: Generalisation error of 100 networks on *Ceratium* trials with random pattern presentation order and added noise. 158

Figure 43: Mean training and generalisation performance of 98 patterns. 160

Figure 44: Mean network and human generalisation certainty of 98 patterns. 162

Figure 45: The weights going into the 3 hidden nodes of the 28 networks: 163

Figure 46: The weights going into the 3 hidden nodes of the 28 networks: 164

Figure 47: Five networks with three hidden nodes showing responses to input, 166

Figure 48: Hidden node activations for 5 networks, 167

Figure 49: Performance gain after pairwise arbitration. 170

Figure 50: Generalisation error of 1000 nearest neighbour *Ceratium* trials. 172

Figure 51: Generalisation performance of neural network and nearest neighbour *Ceratium* trials. 173

List of Tables

Table I. Self-inconsistency of subjects on *Ceratium* classification trials 93

Table II. Consistency between subjects on *Ceratium* classification trials. 94

Table III. Frequency and orientation information in a power spectrum. 101

Table IV. The Exclusive-Or function. 127

Table V. Within- and between- class correlations. 137

Table VI. Network performances on *Ceratium* frequency trials. 142

Table VII. Network performances on *Ceratium* frequency gradient trials. 145

Table VIII. Performance of networks on *Ceratium* frequency gradient trials. 148

Table IX. Performance of networks on *Ceratium* frequency gradient trials. 150

Table X. Performance of networks on *Ceratium* frequency gradient trials. 152

Table XI. Performance of networks on *Ceratium* frequency gradient trials. 153

Table XII. Performance of networks on *Ceratium* frequency gradient trials. 154

Table XIII. Response to three *Ceratium* patterns. 161

Table XIV. Mean arbitration results. 169

Table XV. Mean matched arbitration results. 171

Table XVI. Performance of networks and nearest neighbour on *Ceratium* frequency gradient trials. 172

Acknowledgements.

I am very grateful for the support of a number of people. I would like to thank my supervisors, Rob Ellis and Phil Culverhouse for their employment of me, intellectual input and enormous patience, and without whom this work would have been a lot less fun to do. I also thank Bob Williams for his help and enthusiasm, and the various staff of Plymouth Marine Laboratory who have given their time.

Finally, I thank those friends who have provided me with the *joie de vivre* necessary to any creative pursuit, especially Phil and Hannah and family, for being there.

Author's declaration.

Various results in this thesis have been previously reported in:

Simpson R., P. F. Culverhouse, R. Ellis and R. Williams (1991) Classification of Euceratium Gran. in neural networks, *Proceedings of Conference on Neural Networks for Ocean Engineering*, Washington D.C., USA, August.

Simpson R., P. F. Culverhouse, R. Williams and R. Ellis (1991) Classification of Dinophyceae by artificial neural networks, *Proceedings of the Fifth International Conference on Toxic Marine Phytoplankton*, Rhode Island, USA, November.

Simpson R., R. Williams, P. F. Culverhouse and R. Ellis, (1992) Biological pattern classification in neural networks, *Marine Ecology Progress Series*, *79*, 303–308.

Chapter 1. Neural Networks.

1.1 INTRODUCTION

This thesis is about an attempt to use a relatively new type of technology, namely neural networks, in order to produce a learning device that is capable of learning sophisticated classifications of complex scenes. The prime motivation for the completion of that task is the production of a working device that has applied use in the field. As a piece of research, however, we wish to investigate the principles behind such a device in order to increase our knowledge about neural networks in general. As the title of this thesis spells out, classification within the context of this new technology is the primary concern.

This first chapter introduces the two relevant fields of image processing and neural networks. Section 1.2 discusses briefly some problems encountered in image processing, and refers to the idea of the biological case informing the technological. Section 1.3 introduces various notions within the field of neural networks, and section 1.4 discusses some actual implemented systems, and some related theory concerning the possible abilities of such devices.

1.2 IMAGE PROCESSING

The exponential increase in the information processing power of machines over the past few decades has enabled an increasing sophistication of computing techniques that has equipped us with a wide range of hitherto unforeseen computer applications. While large large improvements in the efficacy and efficiency have been made in the collection, storage, retrieval and analysis of data, there remains an impenetrability of certain information processing tasks to analysis and simulation. Vision, in particular, can be very well-defined task, with well-defined inputs and desired outputs. For some visual tasks it is possible to define what is sufficient data for a useful analysis to be done, and what the analysis will actually be. Typically, the input data will be a grey-level image, which would be available to a human as a photograph perhaps, and to a machine as an array of light-intensity levels. The task might be that of scene description, something an unimpaired adult would find trivially easy. How then is it that we are unable to provide a technique of automating this?

The first attempts at automated image analysis relied on statistical interpretation of the data. Such techniques can be used successfully in limited domains, such as aerial imagery, but lack the ability to be able to be make useful distinctions in less simple, often more natural, domains. Since then, more sophisticated techniques have been applied successfully to areas as diverse as medicine (e.g. Ballard and Sklansky 1965), inspection of industrial components (e.g. Agin 1980), and control of robotic devices (Page and Pugh 1981). In general though, the effectiveness of such applications is due to the highly constrained world that the machine has to deal with. In an industrial context, for example, the operator can ensure that variations in lighting conditions are minimised, and viewpoint maximised, which can often reduce the task to a trivial level. Such preparation is expensive though, and makes each application little more than a one-off, so that the whole developmental process has to be repeated for each new and slightly different application. There is a need for a general vision system that can perform with the same robustness as is exhibited by natural vision systems.

Vision lends itself to a hierarchical description. That is, the processes that vision consists of can be grouped according to their mutual dependencies upon each other. Such a stratification produces a hierarchy that has at the lowest (earliest) level, the most bottom-up and least top-down analysis, while at the highest (latest) level, the least bottom-up and most top-down analysis. Marr (1978, 1982) and Barrow and Tenenbaum (1981) suggest that the first stage of the processing of an image is into edge and line segments, and terminations. This stage of processing is known as early vision, and is characterized by local computations in the image such as filtering, edge-finding and simple feature detection. In Marr's terminology this is the level of the primal sketch (Marr 1982). Such models are ideally domain independent. It is presumed that in any particular environment such methods would be of use. It has been said that practically anything could happen to an image and furthermore practically everything does. However, the physical laws of nature do constrain the transformation from light source to image, and it is such laws that enable us to say (for instance), that the part of an image that is occluding another is an image of something positioned in front of that the image of which is occluded. There are further likely constraints that arise from the nature of matter itself. We know that in general surfaces are smooth, that is the variation in distance of two neighbouring points from us is small compared to their distance from us. It is knowledge of underlying regularities like these that enable sense to be made of the optic array. Such underlying principles apply to any visual task, human or machine, except in the most alien of worlds. It is the exploitation of such constraints that enable an analysis of an image to be done.

In the light of the fact that we have such a paradigm of satisfaction of these constraints as the brain, it is not surprising that research has looked at natural vision systems in order to inform the manufacture of artificial systems. Psychophysics can contribute to such a process. Julesz showed that stereopsis can occur in the absence of object information (Julesz 1960). This discovery shows that stereopsis is a low-level process that operates with low-level constructs. This type of finding is important, showing that machine algorithms that rely on high-level constructs to perform stereo-matching are unfeasible as *natural* simulations, and thus indicates where a different approach might be made. More detailed neurological work was made possible with the development of electrical techniques such as amplification, which enabled the state of activity of a single cell to be recorded (single cell recording). Hubel and Wiesel's work on the properties of neurons in the striate cortex of monkeys illustrate the possibilities of complex sub-units made up of simple cells (see Hubel and Wiesal 1977). The discovery of orientationally tuned edge, slit and line detectors is an example of the explication of the visual process by showing functionally useful properties of collections of neurons. Campbell and Robson showed that independent spatial-frequency tuned channels existed in the visual pathway (Cambell and Robson 1968). Evidence like this at the way natural systems organise visual analysis are taken up quickly by those wishing to manufacture artificial systems, even if the theory is not slavishly followed into practice. There are many suggestions as to what exactly the channels might be used for (e.g. Marr 1982), but it is not suggested that the brain does anything like a full Fourier analysis on incoming data, although exactly this is done in many artificial systems. Such work has led to various interesting results, for example the 'hand-detector' (Gross, Rocha-Miranda and Bender 1972), but such high-level results are less useful in exposing the functional mechanisms that enable such properties. As Marr pointed out, “finding a hand-detector did not allow us to program one” (Marr 1982).

The above is not meant to be read as stating that biological plausibility is the be-all and end-all in vision systems. Many commercial systems operate perfectly well doing the task they were designed for with disregard for the actualities of the brain. Much ground can be made by exploiting constraints that are artificially imposed on a machine’s environment. What is suggested is that the existing natural systems are a potentially rich source of ideas. This combined with the fact that they are the only systems that can do certain things, those most useful and high-level things that we would wish to be able to simulate but cannot, must suggest that to achieve advances in machine vision examination of natural systems is necessary. Only in this way can we hope to enlarge our conceptual apparatus to the extent that must be required to explain a process that transforms an image into a description or memory. How complicated this is is indicated by the problems we have in simulating even low-level vision. It is possible however to research into tasks at high visual levels without waiting for the lower ones to be sufficiently analysed. Artificial intelligence research traditionally has adopted this approach to a notable extent, hence the joke “first assume a ladder”. A lot of work has been done in investigating the analysis of ‘toy worlds’. The idea is that one assumes that a prior problem has been solved, and concentrates on the further issues that this raises. Roberts (1965) worked with a world composed of outlines of prismatic solids, which present straight lines and flat surfaces to a viewer. This has been dubbed a ‘blocks world’. Work by Waltz (1975) on the possible structures of line intersections has enabled coherent description of a scene in terms of its constituent objects. Such work enables problems to be solved before the solution to some, logically prior, problems.

Computationally, much can be done with an image. Simple features can be extracted, surfaces resolved from shading and edges, depth from texture cues and stereopsis and so on. But brute force methods of programming artificial vision systems do not look promising for developing higher-level processing. The introduction of high-level knowledge is essential at some stage in the process of object recognition and scene analysis. Purely data-driven processing can only be useful up to a point. In order to make sense of the world some top-down knowledge is essential. However introducing such top-down data to influence lower processes leads to a combinatorial explosion of complexity. Which high-level knowledge is to be accessed and how used is a complicated decision. There has long been seen the need to discover the knowledge storage, retrieval and usage mechanisms of the brain. Conventional attempts at knowledge representation often deal with high-level constructs and ignore the functional mechanisms that such constructs consist of, their implementational detail (e.g. Minsky's 'frames', 1975, or Hayes' 'commonsense', 1985). As a result of these attempts, many useful tricks can be performed on high-level data, but we still have very little idea of how such processes work on less constrained data (i.e. the real world).

1.3 PARALLEL DISTRIBUTED PROCESSING

1.3.1 Parallel processing

Our experience of effortless and immediate categorisation and access to high-level descriptions of a visual stimulus belies the complexity of the analysis. Workers in the field in the last few decades may have been surprised that access to more and more powerful machines did not significantly ease their burden. Machines that are many thousands of times faster than the brain at the local level are extremely slow and clumsy at the most basic of visual tasks. The operational elegance and efficiency of the brain by contrast has suggested to some researchers that it is the computational architecture of the brain that distinguishes it from serial computers. The conventional Von Neumann model of computing architecture has a single central processor sequentially manipulating data drawn from, and returned to, a memory store. A consequence of this is that only a tiny portion of the data structure is doing anything at any one time. This computational wastefulness has been attenuated to a degree with the introduction of *parallel processing*. Parallel processing describes a range of approaches to computation, which all share the notion of temporally parallel activity, that is, different processing activities taking place simultaneously in time in different parts of the machine (Ullman 1984).

Low-level problems in vision lend themselves to parallel algorithms due to two main reasons: Firstly, an image generally consists of a vast amount of information that needs to be considered, and secondly, there is (generally) no information contained in one point (pixel) of a scene that is logically prior to any information to be extracted from another part of the scene. In other words whilst there is much redundancy in an image generally, there is no specific redundancy that can be used to *easily* limit the amount of information that needs to be considered. Image processing can be considered as the formulation of such redundancies so as to construct a smaller set of information that adequately (for some purpose) represents the cause of the image.

At one extreme of the parallel processing spectrum is the technique of utilising a number of what are effectively CPU-like units, each with its own, or access to a common, memory store. This is often referred to as *message-passing* parallelism. The units are linked together such that a task can be split up and divided out amongst them, in the way that one might use several computers, with a task shared between them, to speed up computation. The extent to which this type of message-passing parallelism is useful is determined by the particular task in question. Certain tasks lend themselves to division while others do not. In planning problems, consequences of actions at one time need to be know at a later time, since they affect the state of the world at that later time. Hence tasks with temporal constraints (such as planning) are often difficult to effectively divide into computable modules.

1.3.2 Parallel distributed processing

There is a emergent (and vocal) school of thought that a radically different computational architecture from that of Von Neumann machines might be utilised to enable the abilities of machines to approach a level more comparable to that of humans performing simple tasks. One of the most visible deficiencies of the symbolic approach to programming brain function is the contrast between the operational speed of the brain and the speed of the brains processors.

> Neurons whose basic computational speed is a few milliseconds must be made to account for complex behaviors which are carried out in a few hundred milliseconds. This means that *entire complex behaviors are carried out in less than a hundred time steps.* (Feldman and Ballard 1982)

This observation has been paraphrased as 'the hundred step rule', viz., a biologically plausible algorithm must perform cognitive processes in less than one hundred steps

(Hofstader 1985). It seems that the only possible explanation for this speed is the massive parallelism of the brain. Thus programming a computer with many thousands of lines of code and relying on the relatively fast speed of electrical connections compared to biological ones to produce 'real–time' behaviour is bad practice for cognitive scientists, however useful it might be practically. The efficiency of the brain is supposed to suggest more efficient methods for performing such tasks than brute force.

At the other extreme of the spectrum of parallel processing from message passing systems is a wide range of systems known variously as *parallel distributed processing* (PDP) systems (e.g. Rumelhart and McClelland 1986), *connectionist* systems (e.g. Hebb 1949, Rosenblatt 1958), *neurocomputing* systems (e.g. Anderson and Rosenfeld 1988) or *artificial neural networks* (e.g. Kohonen 1988). Various of these terms will be used below, loosely, without prejudice or attendance to specific technical, political or hermeneutic interpretations of each. Some of these terms bring to mind the similarities between the functioning of these networks and the functioning of neuronal structures in the brain. Indeed consideration of the structure of the brain (the most sophisticated information–processing device we know of) is one of the motives for considering such systems. Some authors see the development of PDP systems as an opportunity to exploit specific features of brain function (such as massive parallelism, e.g. Feldman and Ballard op. cit.) others have expressed a more encompassing desire for the consideration of brain function:

> "Artificial neural networks" are massively parallel interconnected networks of simple (usually adaptive) elements and their hierarchical organizations *which are intended to interact with the objects of the real world in the same way as biological nervous systems do.* (Kohonen 1988, p. 4, my emphasis)

Commonalities between these systems centre around a structure, or *architecture*, consisting of an arrangement or topology of *nodes*, connected by weighted connections, or just *weights*. A network typically consists of many simple processors (nodes), with many interconnections between them. Each of these nodes, far from being a CPU–like device with all the versatility and complexity that that entails, is a simple device with a very limited local memory store and (strictly speaking) no access to any form of global storage. This local store typically records the level of excitation of the unit. A modicum of calculation is performed within these nodes, typically some calculation of the level of activation of the unit, depending on the signals it is receiving. The signals received by a node are dependent upon the activations of those nodes it is connected to, and the nature of those activations. Such processors are sometimes referred to *value-passing*. A specific instantiated network will consist of such a structure, along with specified weights, and rules for the processing of information there within. Generally networks perform some *encoding* of information, and some *recall* of information. Such networks have been, and are currently being investigated with a view to using them for a very wide range of data analysis and control tasks. In general the types of task that are attempted are those that are performed by the low-level unconscious mechanisms of the brain (e.g. McClelland and Rumelhart 1986b), although there is work being done in applying neural networks to more traditional work in artificial intelligence which concerns itself with high-level (conscious) skills such as planning (Ali *et al.* 1988), sorting (Gray *et al.* 1988) and expertise (Ghosh *et al.* 1988) et cetera, without the manipulation of inappropriately sophisticated symbolic representations.

1.3.2.1 Nodes

Many different forms of processing have been postulated for the nodes to perform. Again generally, three distinct types of node can be identified; *Input* nodes, which take information from some environment. These are typically provided with *input vectors* (representing patterns to be processed by the network) by some operator, although input might come from another network or device; *output* nodes, which are deemed as such by their use as such, in that their activations are taken as output of the network by an operator or some other device; and *hidden* nodes, which neither receive input values directly nor are treated as outputs, but process information from other nodes in the network. Often input nodes are treated in a rather different manner from other nodes in the network, and are, strictly speaking, not nodes at all. They are often operated in deviation from the summation and activation rules imposed on other nodes, and are instead *directly* activated (i.e. their activations set, or clamped, to some value) by the elements of the input pattern, rather than processing these input patterns through some transform (thus serving a distribution function to other nodes in a network). Where this occurs it is shown figuratively by a different symbol for these input nodes (see for example Figure 6). A network structure may consist of *layers* of nodes, with connectivity between layers defined, and each layer incorporating only one of the above types of node (hence *input layers*, *hidden layers* and so on). It should be noted that in the neural network literature usage of the term layer can refer to nodes (either inclusive of the 'dummy' input nodes, or not) or weights between nodes, hence expressions such as "a *n*-layer network" suffer from terminal ambiguity. The present author will adopt the usage of *n-layers* to represent n layers of processing nodes (that is without counting distributing inputs), which in the simple feed-forward case also implies n layers of weights. Any deviation from this convention will be appropriately flagged.

The response of a node to its input is called its *activation* or *transfer* function. Many types of activation function have been used in the past, but most represent some transformation or mapping from an infinite range of a summed input value to a finite range of output. Common activation functions include threshold functions such as the Heaviside function, range {0,+1} and the bipolar function {-1,+1}, and sigmoid (squashing) functions such as the hyperbolic tan [-1 +1] and the logistic sigmoid function, $(1+e^{-x})^{-1}$ [0 +1], see Figure 1. The significance of and differences between these functions is discussed in various places below, primarily in section 5.2.1.1. Figure 1 illustrates the functions mentioned above. In terms analogous to the biological case, a node's activation is often thought of as equivalent to the momentary firing frequency of a neuron. Associated with a node is often a *threshold*, some value that acts as a lateral offset for the activation function, enabling a node to respond to a variety of ranges of inputs. Very often the analysis of the threshold is in terms of another input to be summed, incoming from a node with an activation of 1, through an appropriate connection strength. This reductionist step loses no generality of a model and simplifies the mathematical and algorithmic analysis (Figure 7 illustrates this).

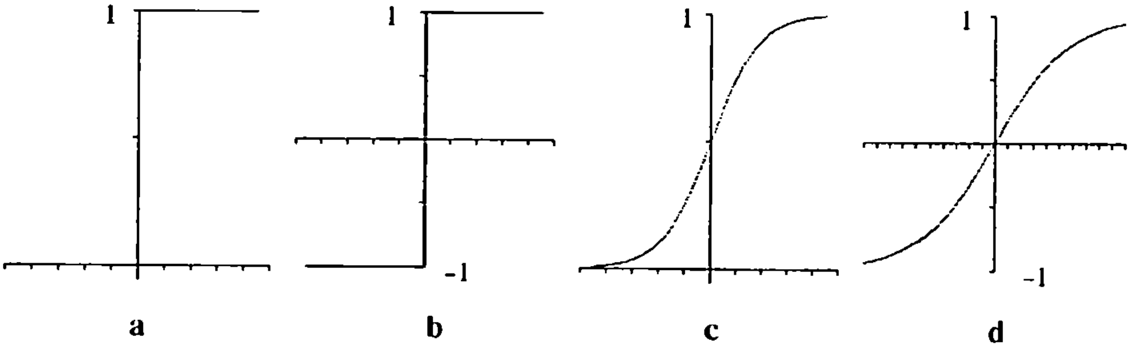


Figure 1: a-d Activation functions. a) Heaviside function, b) Bipolar function, c) $(1+e^{-x})^{-1}$, d) tanh(x)

1.3.2.2 Connections

Connections (or weights) between nodes *propagate* (pass) activations from one node to another. There is an simple linear transformation associated with this propagation, and this may be of absolute magnitude greater than one (increasing the strength of the signal being propagated), less than 1 (decreasing the signal strength) or equal to 1 (leaving signal strength unchanged). The transformation may also be negative, in which case the sign of signal is changed. The value of an input to a node is a function (typically the product) of the activation of another node and the strength of connection between them, in that direction. A node's total input is generally calculated as a simple summation of all the inputs to that node. It is this value that is passed through the activation function.

Depending on the structure of a particular network these connections may be between nodes in different layers, running in a particular direction (*feed-forward* connectivity), or between nodes in the same layer (*interfield* connectivity), feeding back in loops to other nodes (*recurrent* connectivity), or a mixture of the above. A general thesis of neural networks as a field of study is that the weightings of the connections are analogous to the long-term memory (synaptic efficacies) in the biological case (Alkon 1989), and activation of a system is therefore analogous to short-term memory (neuronal activation, or firing frequency) - these terms are sometimes used explicitly (e.g. Grossberg 1988). Given that the functions of the nodes remain unchanged in a network, then any change in the *behaviour* of a network (i.e. its response to specific inputs) can only come about by means of a change in the weights. Since learning must at least be construed as a change in (potential or actual) behaviour, generally, any network learning must be a result of weight change. Hence a particular *state* of a network consists of its architecture (the topology of the nodes and their various connections), and its set of weights (connection strengths), along with the various activation and propagation rules (its operational algorithm).

1.3.2.3 Network functions

What can networks do? As computational devices, an answer is that "(networks) ... can compute only such numbers as can a Turing machine" (McCulloch and Pitts 1943). Given that a digital computer is a collection of Boolean logic gates, it follows that since such gates can be instantiated by a network, any function computable by a digital computer is computable by a (but not necessarily any) network (see section 1.4.1). Function approximation abilities of networks have been shown to be universal (Hornik *et al.* 1989), in that a network can approximate any function *arbitrarily well* (as can a Fourier series), that is to within some given degree of error.

A more sensible answer to the question of what networks can do addresses itself to the particular uses of network computation aside from theoretical questions of computability et cetera. One distinct class of network structures use *relaxation* techniques to obtain a solution. The term relaxation is used where an initial state of activations in a network, due to recurrent connectivity, undergoes a process of change and flux (usually reducing some cost function further at every stage), until a stable state is reached. Such algorithms are often used to find the solution to problems requiring multiple constraint satisfaction. These problems consist of an optimisation task over a number of parameters, which interact in some specified manner. Perhaps the most common of this class of problem is the 'travelling salesman problem' (e.g. Garey and Johnson 1979), which has a number of application outside of sales, for instance in numerous physical design problems. The solution to this problem requires specifying a route passing though each of a number of specified locations, minimising the distance travelled (or some other cost function). A network can be set up as a dynamic analogy to this problem, with connection strengths in a network corresponding to some cost function (distance, say), and activities in the network representing certain variables of the problem. An initial (sub-optimal, perhaps random) state is given, and the network *relaxes* into a stable state, which corresponds to some solution to the problem. This final solution may be optimal, or perhaps only a better solution than the initial given state (e.g. Hopfield and Tank 1986).

In image processing, the calculation of surface orientation from shading has been achieved by using parallel relaxation techniques (Horn 1975). The orientation and reflectance of a point are in all likelihood constrained by the orientation and reflectance of neighbouring points. This is inferred from the optics and the physics of matter. Another example of modelling physical constraints on an image is given in Marr and Poggio's algorithm for calculating the depth of points in a stereo image (Marr and Poggio 1976, 1979). A random dot stereogram (see Julesz 1960) is used as a test image, and coherent correspondences are found between points on either side of the stereogram by using the relaxation of a network. Activity of the network's nodes represents hypotheses about correspondence between pairs of points in the two images. The constraints of physics are modelled by rules of excitation and inhibition between nodes; For instance, Marr cites two constraints on an image due to the physical world:

> (C1) a given point on a physical surface has a unique position in space at any one time,
>
> (C2) matter is cohesive, it is separated into objects and the surface of objects are generally smooth compared with their distance from the viewer. (Marr 1974)

These constraints can be modelled by rules for combining the descriptions of the left and right image:

> (R1) *Uniqueness*. Each item from each image may be assigned at most one disparity value, ...
>
> (R2) *Continuity*. Disparity varies smoothly almost everywhere. .... (Marr 1979)

These rules in turn can be explicitly modelled by the construction of various inhibitory and excitatory connections between nodes representing particular disparity hypotheses. The adapting of parallel processing to obvious mutual constraint satisfaction tasks and to the highly parallel tasks of vision processing has led to a many successful applications (see also Kirkpatrick *et al.* 1983, Hummel and Zucker 1983 and Carnevali *et al.* 1985).

However, more typical of use of neural networks today is in problems of recognition and classification of stimuli, where a particular pattern is the input to the network, and a particular output represents some item or category. The general structure of network used for such tasks is the *feed-forward* network, with activations being passed from a set of input nodes to a separate set of output nodes. A distinction can be made between *spatial*, and *spatio-temporal* pattern recognition. An input pattern to a network might be static, and a desired response be static (say, a category in image recognition) or dynamic (e.g. motor control), or the input pattern may have a temporal component, and the desired response may again be static (e.g. classification of sonar returns) or dynamic (e.g. speech recognition). A network performs a mapping from an input pattern to an output pattern. The input may be part of, or a marred version of, a desired output (for use as a content-addressable memory or in image restoration, both tasks generally referred to as *autoassociation*), or the desired output may have a more subtle relationship with the input, and represent a classification, prediction or action (*hetroassociation*). Some of the much touted advantages of networks are declared to be the ability to work with noisy, or stochastic data, to be damage resistant, to be able to *generalise* from a set of given input-output pairs to new examples of input patterns, and to fail in a graceful manner when either damaged or operating conditions are less than perfect. The most significant ability of neural networks is adaptive, to be able to learn. Discussion of these traits and abilities occurs below, and some pertinent points can be made concerning these issues when discussing specific network structures.

1.4 NEURAL NETWORK IMPLEMENTATIONS

1.4.1 The artificial neuron

The first mathematical model of an artificial neuron is generally accepted to be due to McCulloch and Pitts (1943), treating it as a logical device. Their model of a 'logical' neuron consisted of a processing element with an all-or-nothing activation function - a binary step-threshold function {0 +1} (see Figure 1a). McCulloch and Pitts demonstrated that such a device was logically complete, i.e. that for any logical function a network of such neurons could be devised (see Figure 2a and b for example functions). Simple functions could be instantiated in one single node with an appropriate threshold, more complex functions require a number of nodes. Any logical expression can be constructed out of AND, OR and NOT. McCulloch and Pitts posited this type of neuron as the basic computational device of the brain. Whilst networks of such nodes can indeed perform any

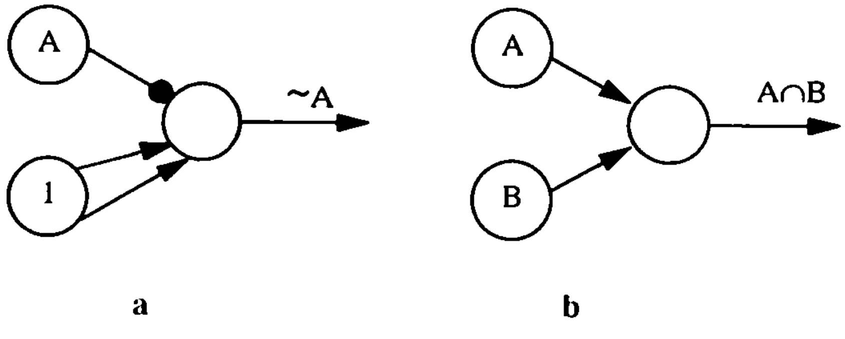


Figure 2: Logical functions, (a) NOT A, (b) A AND B. A node fires if it receives a total of two or more excitatory inputs, (—● = inhibition, —► = excitation).

logical function, there is no clear method to enable one to construct a network to perform a desired function. Indeed, it is often the case that we do not know precisely the function we wish to instantiate.

McCulloch and Pitts' networks have to be hand-wired, constructed in every detail by the operator. The problem of their inability to learn was indirectly addressed by a number of people. Hebb (1949) postulated a process of change in a (biological) network that account for memory. The idea was that structural and metabolic changes would occur in and between neurons, primarily at the synaptic junctions. The operative conditions for this to happen would be when pre-synaptic activity accompanied post-synaptic activity (i.e. the synaptic junctions between neurons are strengthened when both neurons attached to the junction were simultaneously active).

> When the axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes part in one or both cells such that A's efficiency, as one of the cells firing B, is increased. (Hebb 1949, p.62)

A later formulation of this principle is that the increase in the efficacy of a synapse is in proportion to the correlation between the pre- and post-synaptic activities, or that when two nodes are repeatedly co-active then there will be a tendency for there activations to become even more correlated. This is often refer to as the Hebb rule, although it moves somewhat beyond Hebb's own formulation. This type of rule for adaptive behaviour gives the general ground for learning systems. The most significant early development of a learning system was Rosenblatt's perceptron, a feed-forward network with a then astonishing ability to learn input-output mappings. It deserves a section of its own:

1.4.2 The perceptron

1.4.2.1 Rosenblatt's formulation

Hebb's vague reinforcement rule was adopted by Rosenblatt as an adaptive technique in a device of his invention called a 'perceptron' (Rosenblatt 1957). The term perceptron covers a family of network architectures, some with recurrent connectivity and multiple layers, but the elementary (or "simple" - Rosenblatt 1962) perceptron is a one-layer, hetroassociative binary threshold device where each output node sums a set of weighted inputs and responds if this sum exceeds a threshold. The output is a two class response $\{-1, +1\}$. Thus a perceptron dichotomises an input set of patterns into two groups. A randomly set perceptron will do this, but the results are unlikely to be interesting, and it may be that the whole input set is classified as belonging to one group. What is needed is a method whereby the perceptron can *learn* interesting classifications. Ideally a device should be "capable of reorganizing its own logic, to correspond to a logical organization which already exists in the universe around it, ..." (Rosenblatt 1959, p.424).

Rosenblatt illustrated his ideas by postulating a sensory surface or retina (S-units), connected to a set of association units (A-units), which were in turn connected to the response units (R-unit) (see Figure 3). This type of associative learning structure is in contrast to the logistic units of McCulloch and Pitts, which Rosenblatt described as "logical contrivances" (Rosenblatt 1958, p.387). Rosenblatt felt that the plausibility of the logical threshold model of brain function was limited:

> Models which conceive of the brain as a strictly digital, Boolean algebra device, always involve an impossibly large number of discrete elements, or else a precision in the "wiring diagram" and synchronization of the system which is quite unlike the conditions observed in a biological nervous system. (Rosenblatt 1959, p. 422)

Rosenblatt envisaged random connections between the retina and the associative units, so that the actual predicate or feature detected by an A-unit was randomly determined. The response at the R-units comes from consideration of these randomly generated predicates, and constitutes the separation of the input space into two classes with a hyperplane. To investigate the possibility of an unsupervised learning procedure Rosenblatt instantiated Hebb's learning rule in a perceptron, (the C rule). This led to the perceptron learning by experience. Unfortunately, as active lines got more and more weight added to them, they began to dominate the classification of the input stream until all overlapping patterns were classified together. A development of the Hebbian rule was to restrict the growth of weights on lines, such that all the lines would share a set amount of weight, and this was merely redistributed each time the perceptron was to learn a pattern - the C′ rule (Rosenblatt 1959). This prevented lines from becoming so heavily weighted as to dominate the summation of all the lines.

The perceptron will now be considered formally. Figure 3 represents a perceptron with n A-units and m R-units (outputs). Associated with each R-unit is a threshold Θ. This

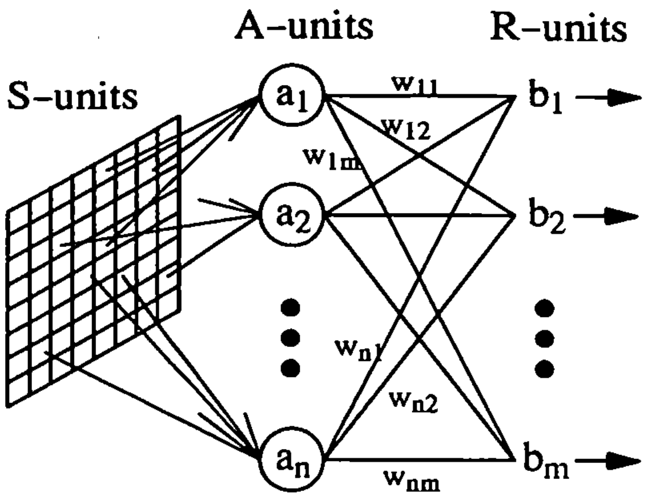


Figure 3: An elementary perceptron with *n* A-units and *m* R-units.

threshold acts as a lateral shift on the inputs to that particular R-unit, so that the threshold of the activation function can be positioned anywhere on the input values, rather than just at zero. A perceptron is initialised by assigning each weight w_{ij} and each threshold Θ_j a random value in the range [-1 +1]. Consider a set of *p* input patterns, giving rise to *p* patterns across the A-units, $\mathbf{I} = \{\mathbf{i}_1, \mathbf{i}_2, \dots \mathbf{i}_{p-1}, \mathbf{i}_p\}$, and a set of *p* output patterns, $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \dots \mathbf{o}_{p-1}, \mathbf{o}_p\}$. The task is to associate the *k*th pattern on the A-units, represented by an vector $\mathbf{i}_k = \{a_1^k, a_2^k, \dots a_n^k\}$, with a bi-polar valued (each term either -1 or +1) output vector $\mathbf{o}_k = \{b_1^k, b_2^k, \dots b_m^k\}$.

A pattern arising from an input stimulus $\mathbf{i}_k$ is presented at the A-units, and the output values at R calculated using the equation

$$b_j = f\left(\sum_{i=1}^{n} w_{ij}\, a_i - \theta\right), \qquad \text{(eq. 1-1)}$$

$$\textit{where } f(x) = \begin{cases} +1 & \text{if } x > 0 \\ -1 & \textit{otherwise} \end{cases}.$$

The required changes in the weights are calculated as

$$\Delta w_{ij} = \alpha\, a_i\, (b_j^t - b_j)\ . \qquad \text{(eq. 1-2)}$$

i.e. in proportion to the product of the activation at an input node and the error at the corresponding output node. α is a term controlling the rate of learning. This process is repeated for all *p* patterns, until the error term

$$E = \sum_{k=1}^{p} \sum_{j=1}^{m} (b_j^t - b_j) \qquad \text{(eq. 1-3)}$$

is zero, or below some desired amount. Rosenblatt proved that this algorithm would converge on a solution for any classification task *provided some such solution exists.* (Rosenblatt 1962). This algorithm is called the perpectron convergence procedure.

Rosenblatt discusses various aspects of the perceptron (Rosenblatt 1958, 1959, 1962), recognising some of its faults as well as its virtues. Rosenblatt recognised the distributed nature of the perceptron's memory, and that damage to the network would cause a graceful degradation in performance, involving all the learned associations. The main problem with the perceptron is that it is really a 'likeness' detector. In an environment with distinct and differentiated categories consisting of similar stimuli, the perceptron does well. The higher the degree of correlation between members of a class the more likely it is that correct discriminations will be learnt. As might be expected, the learning of randomly generated input-output mappings is poor, due to the possible unlikeness of members of a particular class. The ability of the perceptron to generalise from learnt data is poor. The training time was lengthy, since learning was not*forced*, that is weights were only changed if the outputs were erroneous. However, adapting Hebb's descriptive account of learning via synaptic change, remaining consistent with (although simplifying) the then known biological facts, Rosenblatt manage to specify precisely a learning algorithm, thus enabling rigourous testing of his theories to be carried out, against predicted behaviours for instance. It demonstrated "the feasibility and fruitfulness of a quantitative statistical approach to the organization of cognitive systems" (Rosenblatt 1958, p.407). The perceptron was complex enough to exhibit interesting behaviour, yet simple enough to admit to analysis. Rosenblatt himself wrote

> .... it seems clear that the Class C′ perceptron introduces a new kind of information processing automaton: *for the first time, we have a machine which is capable of having original ideas.* (Rosenblatt 1959, p. 449, my emphasis)

The impact this model made on the pattern recognition community was enormous, and many people (more than 100 groups world-wide according to Barrow 1989) set to analysing various aspects of the perceptron, i.e. what can it learn, how big need the retinal field of the A-units be, how big need the weights be, et cetera.

The near-fatal flaw of the perceptron is that it can (of course) only find *possible* solutions to classifying its input, when the very structure of the simple perceptron excludes a whole range of solutions necessary for some problems. It is the (single-layer) perceptron's inability to deal with *non-linearly-discriminable* classifications that is its Achilles' heel. A perceptron divides the input space with a hyperplane, that is, a plane of n−1 dimensions in a n-dimensional input space. Not all desirable classifications can be performed with such a hyperplane. Given that the perceptron acts as a likeness detector, or nearest-neighbour classifier, the problem arises with classes consisting of unlike members. The most simple (if well-worn) example of such a problem is the Boolean exclusive-or function (EOR, or sometimes XOR). The class 'true' consists of the members {(0 1) , (1 0)}, and the class 'false' of {(0 0) , (1 1)}. Each pattern is more like (i.e. closer in Hamming distance) any member of the other class than it is like the other member of its own. The patterns that are least alike are expected to generate identical outputs. No information about one point on the retina gives any evidence as to the correct classification of the retina. Information as to the value at one point on the retina (say, knowing it is a '1') provides no evidence *by itself* as to the correct classification of the input pattern. In general parity problems, knowing even 99 out of 100 of the elements of the input does not enable one to calculate the overall class of the input (i.e. even/odd. Note that knowing 99 out of 100 elements gives one no more information regarding parity than knowing 1 out of 100 elements). Although Rosenblatt did discuss multi-layer systems (which would be able to discriminate non-linearly), his convergence procedure only held for single layer systems. The most damning critique of these systems was to come.

1.4.2.2 Minsky and Papert's criticism of the perceptron

The initial optimism shown over single-layer linear threshold devices was somewhat dampened by the publication of *Perceptrons* (Minsky and Papert 1968). This is a rigourous analysis of the limitations of perceptrons. Today the notion of a perceptron is inextricably link to Minsky and Papert (M&P), as often cited as having irresponsibly claimed limitations of perceptrons and thus (temporarily) putting paid to that line of research, as they are recognised for the sterling analytical work on parallel systems. Modern readers of Rumelhart and McClelland's *Parallel Distributed Processing* (1986) might come away with the impression that *Perceptrons* was not the deserved indictment that some claim it to be, but just a somewhat dated, and now irrelevant, look at the limitations of what is rather a limited device anyway. Much of the presentation of more modern texts gives the unwarranted impression of a distancing from M&P's analysis. In the epilogue to *Perceptrons* (1988) M&P write

> .... it is wrong to formulate our findings in terms of what perceptrons can and cannot do. [....] perceptrons of a sufficiently large order can represent *any* finite predicate. A better description of what we did is that, in certain cases, we established the computational costs of what perceptrons can do as a function of increasing problem size. The authors of [*Parallel Distributed Processing*] show little concern for such issues, and usually seem content with experiments in which small multi-layer networks solve particular instances of small problems. (p.265)

Indeed, perceptrons can represent any finite predicate. A grandmother detector[1] may be constructed by having the coefficients of the perceptron retain the information that dictates every instance of a grandmother image appearing on the retina. This is a very inefficient use of parallel computing, and could equally be done by having a look-up table in which every possible arrangement of pixels was stored along with the Boolean value of such an arrangement being a likeness of my grandmother. When M&P (among others) state that such-and-such a problem is 'impossible', it is meant to be read as stating that *no efficient* perceptron can solve some problem (see section 1.3.2.3).

Minsky and Papert's (M&P's) analysis treated perceptron as computing *predicates*, rather than simple classifications. The A-units calculate simple predicates ψ, which are combined by the R-unit to obtain a value of a more complex predicate Ψ (See figure Figure 4.) Thus rather than trying to get a network to respond with the correct classifications most of the time, M&P asked what predicates *could* be computed. A trivial answer to this is that any predicate can be computed (see above), but the more significant question is what can be computed given some limitations. M&P investigated a number of limitations, but primarily order-limited and diameter-limited perceptrons (which are a special case of order-limited perceptrons).

A predicate is said to have an order of k if it can be computed by linearly combining evidence from a number of predicates which rely on k (or less) points on the retina. For instance the predicate $\Psi_{\text{HAS A LIT CELL}}$ applied to the retina can be calculated by a linear threshold function of predicates each relying on one cell of the retina, and thus is a predicate

---

[1] A detector that fires only when one's grandmother comes into view.

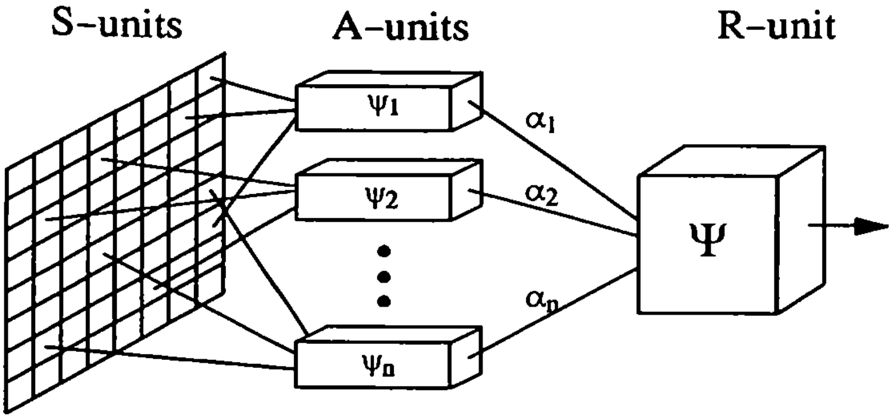


Figure 4: An elementary perceptron, with simple predicates ψ, combining through weights α, to form a complex predicate Ψ.

of order 1. The predicate Ψ_{CONVEX} is of order 3, at least three points of the retina need to be examined for evidence for (or against) its validity. This measure of order of a predicate is important, since any predicates that have an order dependent on the size of the retina will scale up badly when applied to larger retinas. If a predicate is of order k, then the number of simple predicates ψ required scales to the order of n^k. $\Psi_{\text{HAS A LIT CELL}}$ and Ψ_{CONVEX} are of orders 1 and 3 respectively regardless of the size of the retina. Ψ_{PARITY} has an order dependent on the size of the retina, thus the cost of computing this predicate increases dramatically with the size of the retina. In fact at least one of the A-units must receive input from every cell in the retina.

This type of analysis can indicate when success in an application is simply a result of the 'toy' size of the problem, and that an increase in retina size would lead to unfeasible size of predicates. Order can be calculated for multi-layer linear threshold networks as well as simple perceptrons. The order of a network is calculated as the product of the orders of each successive layer. No improvement in the (unbounded) order of a perceptron can be made by having further linear-threshold layers. Multi-layer perceptrons do have some advantage over the single-layer variety, though. Some computational efficiency can be gained, and since multi-layer networks can have recurrent connections such systems may be able to compute predicates of unbounded order, in effect computing serially.

The second problem associated with scaling up perceptrons is that of unbounded weights. For instance, M&P discuss the predicate $\psi_{\text{SYMMETRICAL}}$. If we have the centre of symmetry in advance this problem is fairly easy, requiring a perceptron of order 2 that simply detected differences between symmetrically paired points on the retina. One solution to this problem in the absence of a given centre is *stratification* (a phenomena found to arise spontaneously in networks, see Rumelhart and McClelland 1986, p.340) The use of masks to detect various features is combined with a 'scanning' phenomena that arise when suitable weight coefficients are used. The problem with the use of masks is that success with one mask is interfered with by the failures of other masks. Stratification uses a series of weights such that success at one level of strata overwhelms any lower-level failures. The scaling of the weights is such that the addition of retinal cells requires exponentially larger weights than before. Doubling the size of the retina requires doubling the number of significant digits needed by the weights. The problem is not that large coefficients are required, rather these must be represented such that a node can distinguish between a value of (say) 1 and 2, and still operate successfully with weights many of orders of magnitude greater. What might seem a successful application on a small toy-sized retina may become unfeasible when scaled up.

These problems, together with the proofs of what predicates were 'impossible' (in M&P's sense) for the single-layer perceptron, constituted a crisis for neural network research. Many authors claim that the publication of *Perceptrons* was responsible for the subsequent lack of interest, and funding, in the field (e.g. Rumelhart and McClelland 1986, Barrow 1989 and Simpson 1990). Minsky and Papert describe the dilemma of connectionism as this;

> Perceptrons could learn anything that they could represent, but they were too limited in what they could represent.
>
> Multi-layer networks were less limited in what they could represent, but they had no reliable learning procedure. (Minsky and Papert 1988, p.256)

It was partly the partial resolution of the second horn of this dilemma that was responsible for the re-emergence of neural network research in the 1980's.

1.4.3 Multiple constraint satisfaction

Since the publication of "Perceptrons" many attempts at a general learning rule for multi-layer and recurrent systems have been made. One of the most significant early papers in this period was by Hopfield (1982), in which he introduces a model for an associative memory, which recalls patterns through a process of *energy minimisation*. Based on an analogy to physical systems (i.e. spin glasses) Hopfield showed that a dynamical process of asynchronous and stochastic updates of neurons leads from the initial network state to a *local* energy minimum (instead of oscillating). This technique has been applied to the Travelling Salesman problem (Hopfield and Tank 1986, see section 1.3.2.3). The whole process is one of multiple constraint satisfaction. A similar technique has been used in Hinton and Sejnowski (1986), in a stochastic device called the Boltzmann machine. The training of a Boltzmann machine attempts to avoid entrapment in local minima by using probabilistic decisions to change the location on the error surface that a weight vector occupies. The process is analogous to hill-climbing, although inverted. The technique of gradient descent is to alter parameters in such a way as to reduce some cost function (usually some error measure) at each step, hopefully leading to a *global* minimum (a point of lowest cost function in the parameter space). However, it is not always possible to ensure that a global minimum is found, on occasions local (or false) minima may be found, in which case no further parameter change takes place, in spite of the fact that the point reached is not the best solution. The elementary perceptron can be envisaged as performing a gradient descent technique in a weight space (which due to the nature of the device is) entirely free of local minima. Introducing more than one layer into a network alters the shape of the weight space, possibly introducing local minima (see McInery *et al.* 1988 for proof of *high* local minima).

The Boltzmann machine is started with an set of input vectors and a corresponding set of desired output vectors, which combined with the weights vector result in a certain amount of error between the desired and actual responses. This is what is to be minimised. At the start of the gradient descent the system is 'hot', that is there is a high probability that a jump across the gradient space will be made although this may increase the error. This amounts to saying that the search for a minimum is somehow reckless, the system is being shaken to prevent the search falling into shallow local minima. Progressively the 'temperature' of the system is cooled down, that is, the probability of jumps across the gradient space becomes progressively less and less. This is done slowly. The procedure is analogous to annealing, where hot components are cooled slowly to enable the constituent molecules to naturally align themselves with each other (and this computational strategy is often referred to as 'simulated annealing'). It has been shown that this is a very effective gradient descent technique, although not fool-proof. However it requires a very large number of iterations on a given set of data to work satisfactorily, and this makes it unsuitable for networks where fast learning is required. There is no biological plausibility for such a technique.

1.4.4 Multi-layer networks

1.4.4.1 Linear, linear threshold and semi-linear activation functions.

Probably the most direct challenge to the critique of *Perceptrons* was the development of learning algorithms for multi-layer networks. Rosenblatt's perceptron convergence procedure has been adapted by Widrow and Hoff (1960) for linear activation networks (without thresholds), using the weight update rule given in section 1.4.2.1,

$$\Delta w_{ij} = \alpha \, a_i \, (b_j^t - b_j) \, . \qquad \text{(eq. 1-2)}$$

but with the value of the output b_j calculated without applying a threshold function during the learning phase, thus

$$b_j = \sum_{i=1}^{n} w_{ij} \, a_i - \theta \, . \qquad \text{(eq. 1-4)}$$

This means that learning is still carried out even when the network is giving the correct responses during the recall phase (as according to the threshold function in equation 1-1). This procedure leads to faster learning, since learning is not slowed down as more and more input patterns are correctly classified. This procedure has been called, variously, the Widrow-Hoff procedure, the Least Mean Squares (LMS) procedure, and the delta rule. However, whilst having a learning procedure for linear units is useful, it is not possible to increase the power of a linear network by having additional layers. This is because the linearity of the nodes activation functions mean that the calculation of the outputs of one layer is equivalent to a straight-forward multiplication by a weight matrix. Further layers of units only introduce further such multiplications. Since a series of any number of multiplication of matrices can be replaced by the resultant matrix, any multi-layer linear network can be replaced by a functionally identical single-layer network. So the introduction of further layers serves no purpose. Hence for linear units we have a learning algorithm, but hidden units are no use, whilst for linear-threshold units we have an algorithm, but not for hidden units. What is needed it seems, is a function somewhere between linear, and linear-threshold. Non-decreasing and differentiable, or *semilinear* (see Rumelhart and McClelland 1986, p.52), activation functions have been widely utilised to solve this problem. An example of such a function is the logistic sigmoid function, $(1 + e^{(-\eta x)})^{-1}$, where η is a parameter determining the steepness of the curve. This is illustrated in Figure 5 for various values of η. It can be seen that at $\eta = \infty$ this function acts as a threshold function, whilst for small values of η this function approximates to a linear function.

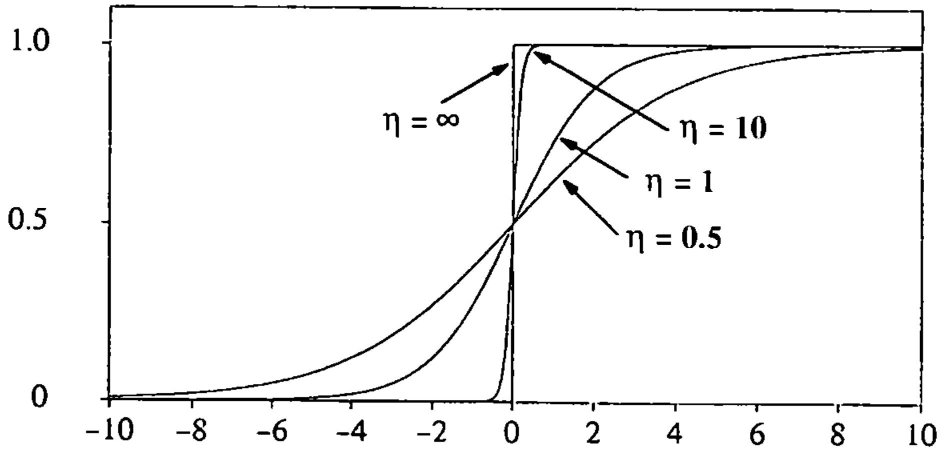


Figure 5: Sigmoid functions $y = 1 / (1+e^{-\eta x})$, for $\eta = 0.5, 1, 10$ and ∞.

The formulation of learning rules for multi-layer systems was made possible by the introduction of these semilinear activation functions. The most famous and widely used of these learning algorithms is the backward error-propagation algorithm.

1.4.4.2 Back-propagation

The backward error-propagation algorithm (commonly called back-propagation) is best characterised as an extension to the Widrow-Hoff rule (see section 1.4.4.1), referred to as the delta rule in Rumelhart and McClelland (1986). The delta rule itself is a generalisation of Rosenblatt's Perceptron convergence theorem, which (as described above) guarantees a convergence on a solution where one exists for the perceptron. However, use of the back-propagation algorithm *does not* guarantee any such convergence, but in practise is often capable of finding solutions. The generalisation of the delta rule for multi-layer perceptrons effectively solved the so-called *credit assignment* problem, in allowing portions of the output error to be assigned to nodes in layers previous to the output layer. The discovery of this solution is generally credited to a number of people working independently. The first use of gradient descent in training multi-layered perceptrons is credited to Amari (1967), but the first development of a true back-propagation algorithm came in a doctoral thesis by Werbos (1974), and was independently discovered by Parker (1982). Credit for the massive current awareness and utilisation of this technique must however go to Rumelhart, Hinton and Williams (1986a) whose experimentation with the technique (see Rumelhart *et al.* 1986b) has led to many other researchers using similar algorithms. The equations for the back-propagation algorithm are given in section 5.2.1.

Consider an input to a network consisting of p elements. Each element is treated as an activation of a node in the network's first layer, such that the input layer consists of an *ordered* list of such activations. These activations are to stand for the first layer of the network (although, strictly speaking, this first layer of activations are not node activations but raw inputs). These activations are then propagated through the various connections to the next layer, where the propagation of each activation passes through some weight, i.e. some multiplying factor on the connection between two nodes. Thus any particular node (not in the input layer), will receive activation from a number of nodes below it, the activation of each amended somehow by the connection strength or weight. Thus the input to any particular node j (not at the input layer) is typically given by the summation of these

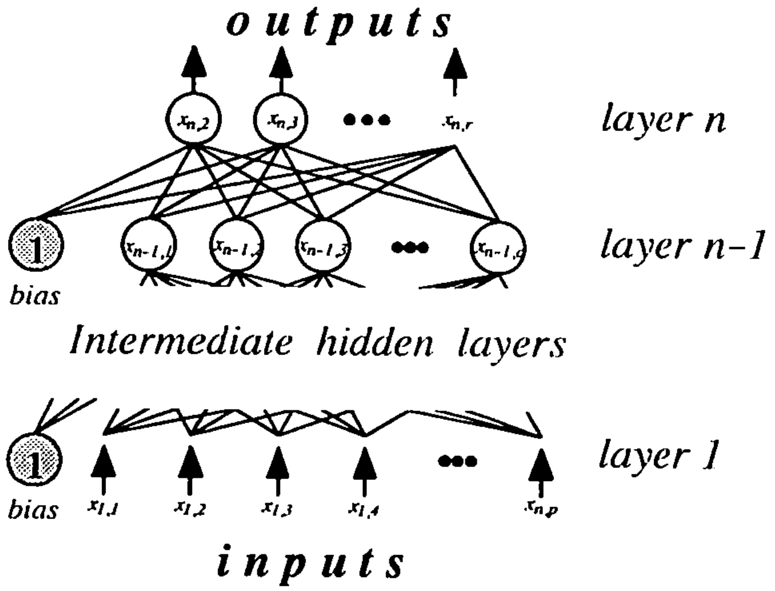


Figure 6: A feed-forward network architecture.

incoming activations and an extra term, θ, termed the *bias*, which plays a similar role in a network as does the threshold in the perceptron (section 1.4.2.1). This extra input consists of an additional activation to a node, and is necessary so that the total input to a node can be shifted from the actual summed inputs, thus allowing a node to distinguish between two high valued inputs, say 100 and 101. Without this term the actual inputs would dictate the output of the node, and thus make various desired outputs unobtainable. The bias term can be thought of as being able to shift the dynamic range of the inputs to a node to within that nodes operation (or, acting as a variable lateral shift on the node's activation function). For instance, if it is desired that a node respond with a high activation to a null input, this can only be achieved by using the bias. The bias can be regarded as another weighted connection to an addition node that always has an activation value of 1 (see Figure 7). The weight from a bias node to another node is updated as is any other weight.

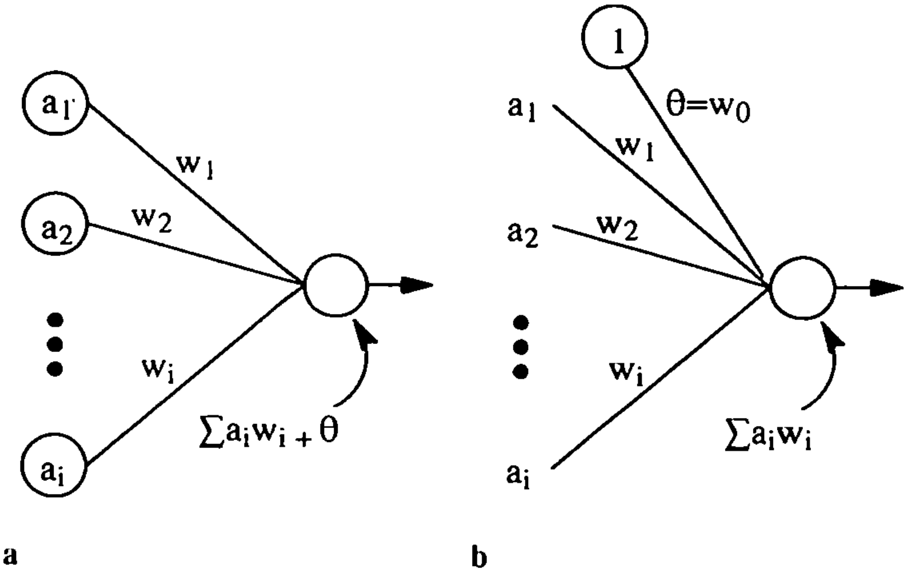


Figure 7: a) A network with a threshold value in the node function. b) A equivalent network, but with the threshold considered as an additional input (bias).

Associated with each node is an activation function which dictates the mapping from an input to a node to that node's activation (and thus output), and which serves to constrain the possible outputs of a node to within some particular range. The most important characteristic of any activation function used in the back-propagation algorithm is that it is non-linear. A linear transfer function, although useful in single-layer (i.e. no hidden layers) perceptrons, renders the additional layers in the case of a multi-layer network useless, since any linear multi-layer function can be replaced by an identically mapping single-layer function (see section 1.4.4.1). It is the non-linearity of the multi-layer perceptron that gives it its power to form complex mappings.

Given (taught or environmental) input activations are propagated through the first layer of weights, resulting in impinging inputs upon the subsequent layer of nodes, for which further activation values are calculated. This forward propagation of activation continues until the activations on the final (output) layer are finally calculated. Thus the network has produced an r-dimensional output for the p-dimensional input (see Figure 5). It is this set of values that is desired to be some particular target vector.

The error correction procedure works by calculating the error term for each node, and correcting the weights leading to that node, calculating the error term for the subsequent layer of nodes, and so on until all the weights in the network have been adjusted. The network is repeatedly exposed to the set of training pairs, and update of the weights may be done either individual to each training pair (*on-line* training), or once after exposure to the whole training pattern set, by cumulatively storing weight updates (*batch* training). Once the global error has been reduced to some particular amount, training may cease, and the network be used only in the feed-forward (recall) operation, to give an output for any given input.

This process is not guaranteed to find the global minimum in error space (as is Rosenblatt's perceptron convergence procedure), only a local minimum. Indeed, theoretic proofs exist showing that there are cases where a back-propagation algorithm will fail whilst a perceptron does not (where the data are linearly separable), (see Brady, Raghavan and Slawny 1989, but also Sontag and Sussmann 1991 for an alternative error function that ensures that this does *not* happen). However, in practice a suitable minimum is often found, although it is not possible to predict this in advance (a criticism made in Minsky and Papert 1988). Given the lack of theoretical tools for calculating when this procedure is appropriate to the given problem domain, most of the users of this procedure assess its suitability by trial and error. While many toy (although perhaps significant) problems have been shown to be solvable by backward error-propagation (for example parity and the 'T-C' problem, Rumelhart *et al.* 1986b), experimentation has proceeded in more complex domains, for instance, medical data analysis (e.g. Silverman and Noetzel 1990, Rayburn *et al.* 1991) and signal processing (e.g. Lapedes and Farber 1988, Gorman and Sejnowski 1988). Inevitably, attention is focussed on successful applications of this technique, while failures remain unpublicised. Entering novel domains with one's tool in hand, it is hard to assess the likelihood of success, and further, the significance of any success encountered. In the current experimentation a novel domain is approached, and techniques developed for the successful application of neural networks to a limited problem in that domain. The fact that limited success was attained in a relatively complex domain is further evidence that this technique does have real-world applicability. Despite the increasingly diverse use that back-propagation is being put to, there is still no available theoretical framework to suggest where and when back-propagation will succeed, and where fail. Given that there is no guaranteed method for solving any particular mapping problem, any available evidence that this technique has widespread application is welcome.

1.5 SUMMARY

In this chapter the theoretical framework for neural networks is introduced. The notion of parallelism is introduced, and various network models and their capabilities discussed. It is suggested that the utilisation of massively parallel processing techniques may enable progress to be made in various artificial intelligence domains where the traditional algorithmic approach has failed. By borrowing ideas from the architecture of the brain, it is hoped that learning machines can be developed enabling the discovery of solutions requiring the adaption of many parameters.

Chapter 2. Classification.

2.1 INTRODUCTION

This thesis attempts to investigate the workings of a particular class of classifiers known as connectionist models, the examination of which may serve to illuminate existing notions of categorisation and concepts. Connectionist models (also known as artificial neural networks, or parallel distributed processing systems) have been shown to be able to learn particular classificatory schemes through exposure to the data. An ability to learn some classification might be regarded as an ability to form concepts. The naked deterministic mechanism of a neural network theoretically allows us to examine its workings in their entirety, to enable post-hoc analysis of the developed classificatory schemes. However, the very complexity of those workings can make a complete explanation and elucidation of such a device difficult, if not impossible, to come to. More significantly, the simulation of a theoretical model of categorisation forces particular design decisions within that theory to become apparent, and exposes under-determined areas of that theory. This is the working premise of the study of artificial intelligence: that the interaction of theory and simulation will benefit the former by applying the rigourous analysis of the latter to it, and the latter by couching the former in an explicit computational vocabulary (Boden 1977).

This chapter will examine the two key areas of this thesis, connectionist models and classification, by reference to previous theories of classification, and existing connectionist models. An attempt will be made to show where and how particular connectionist architectures are valid computational representations of certain theoretical models of categories.

2.2 CLASSIFICATION

"There is nothing more basic to thought and language than our sense of similarity; our sorting of things into kinds.".(Quine 1969, p. 116). The ability to recognise that a number of entities are, in some respect, similar, is the single most important cognitive ability that the thinking animal possesses. The process of abstraction from the raw percept to some representation of that percept is necessary for the functioning of perception itself, and hence memory, problem solving and language. Without abstraction, our perceptual world would be "... one big blooming buzzing Confusion. .... potentially resolvable, and demanding to be resolved, but not yet actually resolved into parts." (James 1894/1961, p.29).

Exact perceptual reoccurrences are certainly rare if indeed possible, and it has even been suggested that generalisation, the ability to handle novel percepts, should be the basis of psychology's first general law (Shepard 1987). Being able to judge two things as alike in some respects enables an epistemic bound to be made, one that disengages the perceiver from the ever-changing flux of sensation, and allows for conceptual groupings to be made that make possible the interpretation and recognition of a scene, event or entity. *Representation* is the means by which we are able to do this. To represent something, means that another thing is to stand for the first. For this other thing to serve some useful purpose we might reasonably expect it offer some advantage over that which it represents. The most apparent benefit of a representation is that it is in the realm of the mental, and is thus able to be stored, carried, retrieved and utilised in mental processes. Great cognitive economies can be made by recognising that two different (i.e. not having an identity relation) items are members of a particular category. Rather than each new percept being regarded as a novel entity about which nothing is known, identification of a percept, as (in all likelihood) a member of a category about which something *is* known, enables inferences to be made about it. The formation of categories, concepts or classes (1) is data compression at its most basic level. Useful data compression will bring to the fore relevant information by rejecting irrelevant information. To be able to predict some change in our world we need to be able to locate, retrieve and apply, small parts of our knowledge, which are initially buried in a mass of knowledge about the world. If we perceive a tiger in the forest, (for instance), we need access to information regarding the danger of the scene, and not (in this context) irrelevant information about leaves, fur and sunshine. To apply our knowledge of tigers to the present scene we need to be able recognise the percept as 'a tiger', that is, something which shares a

---

[1] The terms category, concept, and class will be used interchangeably, as I feel like, throughout, as will their associated verbs.

great many properties with other causes of 'similar' percepts. "Concepts", said Johnson-Laird and Wason (1977), "are the coinage of thought."

2.3 THEORIES OF CATEGORISATION.

Perhaps the most immediate distinction to be made is between recognition of an object as *itself* (i.e. classifying the cause of the percept as a single unique entity), and classifying it as a member of a particular class (i.e. one of *these*). In actual fact these processes will often occur concurrently, with a particular percept being (perhaps) recognised as Lottie (an individual), classified as a dachshund, dog, mammal, animal and so on, depending on the context. Wittgenstein (1953) distinguished between "seeing something_1 as something_2", giving two usages of the word 'see': see_1 as in "I see this rose", and see_2 as in "I see this picture *as* a rabbit" (of an ambiguous duck / rabbit picture, my emphasis). Bruner *et al.* (1956) distinguished the process of discrimination of stimuli, thereby allowing their identification as unique entities - *seeing one as one* - from the process of grouping of similar entities together and thus rendering them equivalent (in some sense) - *seeing many as one*. In this thesis, *identification* will be used to refer specifically to the former sense of recognition, and *classification* and *categorisation* will refer to the latter sense, that is, the allocation of a class or category to a percept.

Formal theories of categorisation attempt to provide an account of the structure of categories (classes), i.e. how they are represented (logically, psychologically or even physically), and the various processes that come to bear on those representations, in terms of the formation of categories, calculation of membership functions, and update of categories. Logicians would define a category by its *intension* and its *extension*. The intension of a concept is the list of predicates which *define* the concept (what a category is defined to be, however this definition is to be achieved). The extension of a concept is the list of all entities that are within the definition of that concept. To use an over-simplifying example, the intension of 'cat' might be a list including such things as furry, mammalian and so on, while the extension of 'cat' would be a collection of numerous (mewing) physical objects, perhaps also including various images, sounds, sculptures et cetera. Quite how inadequate such an over-simplification is will become clearer on considering various posited theories of classification. Typically, existing theories are regarded as being either *classical*, or *probabilistic*, this latter term applying to a wide range of differing theories (for example, Smith and Schaffer 1978, Rosch and Lloyd 1978).

2.3.1 The classical model

2.3.1.1 Defining features

The classical view of categories it that a category consists of a set of defining features. Each one of these features must be present in an entity for it to be classed as a member of that category - the features are '*singularly necessary*'. Any entity with all of the defining features of a category is a member of that category - the features are '*jointly sufficient*'. All the members of a category have all these features, and no non-members have all of these features. All possible entities are either a member of a specific category, or not. There is no membership / non-membership overlap. Hence there are sharp member / non-member boundaries to a category. Thus of the features (or properties) 'three sided', 'straight-sided' and 'internal angles adding up to one hundred and eighty degrees' any entity with all these properties will *necessarily* be a member of that category thus defined - a triangle. Nothing without *all* of these properties is a member of that category. Thus the properties are critical in determining an entity's membership of a given class. They are often referred to as "critical attributes". A consequence of this is that all the features of one class are inherited by any subordinate class, for example, a triangle has the attributes discussed above, and (a subordinate class) an equilateral triangle has all those attributes, plus one of its own (namely that of having equal length sides).

According to the classical view, the intension of a class completely determines that classes extension. Or, the set of attributes (that make up, or rather define, the class) determine uniquely the membership of the class in the world. No other information is required to decide whether an entity is a member of the class or not. There is, therefore, no element of gradedness in category membership, an entity either *is*, or *is not*, a member of a particular class. Any one example member of a class is therefore as good (an example) as any other.

Another idea associated with the classical view is that the classes used in the world are essentially arbitrary, i.e. the world could be cut up an indefinite number of different ways.

That we happen to do so in the way we do is due to our culture, and our language. For instance, Brown and Lenneburg (1954) found that colours most easily named by a particular culture were the colours that were most easy to recall. Since from an objective (physical) viewpoint the spectrum is a continuous scale (of electromagnetic wavelengths) with no significant demarcations, Brown and Lenneburg's results have been taken to be a predictable consequence, and partial confirmation, of the Whorf-Sapir hypothesis of the effect of culture and language on the way we perceive the world, namely that language determines one's conceptual scheme (Carroll 1956).

2.3.1.2 Implementing classical theory

Such a categorisation could be directly implemented in a neural network, say by using an binary input Adaptive Resonance Theory (ART1) model (Carpenter and Grossberg 1987). The ART1 model is a competitive feed-forward network that classifies given input patterns according to their similarity to one another. Rumelhart and Zipser (1985) characterize the competitive learning paradigm as consisting of

i) inhibitory clusters which perform the classifications. In such a cluster only that node which receives the largest input is allowed to output a value, (typically of 1) – hence a 'winner-takes-all' cluster. This is achieved by mutual-inhibition and self-excitation of the nodes. There can be several such clusters, each performing independent classifications, but these must not overlap.

ii) learning on a set of lines only when they are abutting the winning node of a cluster. Only the winning node in a cluster will 'learn' from the input pattern. Thus a response to a pattern will tend to reinforce the likelihood of that same node winning when the pattern is next presented.

iii) a fixed amount of weight on a set of lines. Thus increasing the weights on those active lines of a set will cause a decrease in the weights on inactive lines of that set. This is to prevent the weights reaching such a magnitude that more and more patterns fall within that category, as happens with the Rosenblatt's 'C rule' (Rosenblatt 1959).

Grossberg has shown that such a model will learn stable classifications give sparse data inputs, and sufficient competitive nodes, (i.e. that there are not too many input patterns relative to the number of classifications that can be made, and that there are not too many clusters in the input pattern set), (1976a). However, he has also proved that a competitive net cannot learn a stable classification of an arbitrary input stream. Grossberg's ART1 model attempts to provide an internal supervisor in the net to dictate when learning is to be suspended, when to trade off plasticity for stability. The net is then self–organizing, and can distinguish between relevant codes that must be learnt, and irrelevant codes that lead to unlearning of previously learnt patterns. The two major differences between this model and the typical competitive net (as described in Rumelhart and Zipser 1985) are firstly, the top–down set of weights (or LTM) to the input layer, feeding back from the competitive layer, and secondly the orienting subsystem, which is responsible for supervising the switch between the stable and plastic states of the system. It is the interaction of these two devices that enables the system to detect when learning will be unlearning, that is when to develop a new representation for a pattern rather than recode an previously learnt one.

How variant members of a particular category are allowed to be is determined by a *mismatch tolerance* parameter. With this parameter set to zero, only identical patterns would be assigned to any one category. With each bit of the input pattern representing some defining feature, such a model would class an input precisely as belonging to that category consisting of all such (identical) inputs. The slightest change in an accepted input pattern (say, an additional or missing feature) would result in the new input being classed as belong as a different category from the original. It is interesting to note that Grossberg calls the individual bits of input information in his model "critical features".

It must be noted, however, that this use of the ART1 model, in deviation from the classical model, would not be able to represent the situation where one category was a sub–category of another – a sparrow is a bird is an animal – by virtue of having all the critical attributes of the higher category (as well as some extra ones of its own – as in the triangle / equilateral triangle example). In addition, the ART1 model of classification makes no provision for the classified entities having additional, non–critical attributes. Our previous classical definition of a triangle does not limit any member of that class to only those attributes. Any actual member of that class (with the possible exception of Platonic forms) will have an indefinite number of additional attributes – being blue, drawn in pencil, seen by Geoffrey, and so on. Whilst the ART1 model can allow contingent features in a pattern by allowing an increase in the amount of mismatch tolerated, this toleration cannot be applied selectively to only some (contingent) features without also being (incorrectly) applied to the (necessary) critical features (thus rendering them singularly unnecessary).

2.3.1.3 Failures of the classical model.

A major short-coming of the classical theory is that it fails to account for a number of experimental results. An instance of this is the typicality effect, in that subjects often judge certain members of a class to be more or less typical than other members of the same class, (whereas classically the item either *is*, or *is not* a member of the class, with no degree of gradedness allowed). Not only are items able to be assessed as more or less typical, but there is a high degree of correlation between the assessments made by various subjects (see Rosch 1985, and Lakatoff 1987). Also, subjects appear to use non-necessary features to judge the category membership of test items, for instance, something flying would be a strong cue for it being a bird, although not all birds fly.

Another failure of the classical view is the inability to explain many categories in the world of natural and man-made objects. Natural concepts seem not to be well-defined. There are many common correlations in the real world – such as feathers and wings – and it would rather undermine the enterprise of cognitive economy if each of these single features were represented separately. This would amount to a redundancy in the information being stored. More importantly, what would be the critical features of a category like 'tiger', say? Can you imagine a tiger without stripes? It seems that people can have their categories 'tampered' with to a much greater extent than this and yet still retain an item's membership (a three-legged albino dwarf tiger for example). This is partly due to the inadequacy of defining features. We have all seen, and recognised as such, a three-legged dog. No amount of sophistication in the critical features (replacing "has four legs" with "has four legs, unless has three legs and a stump, unless has ...") will leave the feature list sufficient and necessary. A classic example of this difficulty is the construction of a set of defining features for the idea of "game" (due to Wittgenstein, e.g. 1953). It seems impossible to come up with a list of necessary and sufficient properties of such categories. Anything that can be manifest in such a variety of different ways seems to undermine the principles of denotation by critical features. There is nothing that all games have in common - some are about competition, with winners and losers, some about mutual amusement, and so on. Wittgenstein (1953) writes "We see a complicated network of similarities overlapping and criss-crossing". He uses the term *familienähnlichkeit* - family resemblance - to describe this disjunction of subgroupings that constitutes a category. Various attempts have been made to encompass these disjunctive groupings in a theory of classification, and will be discussed below.

The above assumption of defining features – with singular necessity and joint sufficiency – being the underlying structure of categories, are partly due, say Rosch and Lloyd (1978), to the presumption that

> "the 'mature western mind' ... employed Aristologian laws of logic, ... to know a category was to have a clear-cut, necessary and sufficient criteria for category membership."

A result of the pervasiveness of the classical view is that much work has been done using artificial stimuli such as 'red squares', blue triangles' et cetera, to elicit information about the structure of such categories, search strategies and so on, (For example, Bruner *et al.* 1956) in the belief that such concepts were (in implementation) similar to the more natural concepts we use every day. A major rethink of classification was caused by work in natural domains by Eleanor Rosch and her colleagues. While studying cultures with different colour words to our own Rosch found that recognition of previously seen colours was independent of the ease of naming these colours, suggesting that the different classes of hue were non-arbitrarily determined. In contrast to Brown and Lenneburg, who used speakers of their own language, Rosch found that recognition errors were similar across the language divide (see Rosch 1973). This was an empirical refutation of the Wharf-Sapir hypothesis – the linguistics and cultures of the situations appeared to have an insignificant effect on the ease and rate of categorisation of the stimuli. Rosch showed that the Dani tribe (of Papua New Guinea) had the same perceptual preferences across the spectrum as did English speakers, despite only having two colour categories (*mola* and *mili*, approximating to light / warm, and dark / cool, respectively). Thus, Rosch surmised, the classifications that were made explicit by English colour words were the result of "underlying perceptual factors" (Heider 1972). It does seem obvious that our physiology will be implicated in how we classify things. A bat, with little sight but advanced echo-location will perceive differences we miss, and be unaware of similarities we observe. This constraint is a much more immediate one than that of culture and language, since it is the result of the filtering our physiology gives the physical world (attenuating the "blooming buzzing Confusion") that is in turn variously affected by culture and language.

2.3.2 Classes of categories.

How then should we treat those categories that do seem to have formal critical attributes? We would not want an object with only some of the critical attributes of a triangle to be classed as a triangle. Since a triangle (along with many other geometric figures) seems to have necessary and sufficient attributes, is it a class *not* amenable to prototypical representation? One could implement such a strict feature list by the appropriate fixing of weights and thresholds. This is, though, something of a bodge, and it seems that it is more true to represent such critical features as being at another level of categorisation than the categories of things like 'doughnuts' and 'dog'.

Osherson and Smith (1981) suggest that we may have one model of a concept for identifying a member of that category, and another for the explication of the relationship between that and other concepts. The example they give is the concept of 'woman'. We may have a 'core' concept of a woman involving reproductive systems, chromosomes and so on. We might also have an 'identification' function which uses attributes such as length of hair, pitch of voice, and so on. Obviously the identification function is the one which most of us would use all the time, and the core concept would be used only in special circumstances. Armstrong, Gleitman and Gleitman (1983) point out that the core concept is typically that employed by an expert in a particular field [1]. They also (taking a leap of faith) suggest that there might be a further level of categorisation, that of the essence of a category. These are what Putnam would call "natural kind terms", or Kripke "rigid designators". What this classification would consist of is not known. Such terms appear to have no fixed intension, only an extension. Armstrong *et al.* put the case of gold – we all have a rough and ready identification function for gold (gold-coloured, malleable, shiny et cetera), an expert would have a more sophisticated concept (atomic number 79 and all that entails, et cetera) and no-one has the real essence of gold wrapped up. We may not have the concepts full connotation, but we do have its denotation (gold!).

---

[1] In the case of gender verification, these might include the International Olympic Committee. It should be noted that this is not as clear-cut a classification as one might expect, and there is still controversy over various suggested techniques - see Koopman *et al.* (1991) and Ljungqvist and Simpson (1992). It seems that there is *no* universally accepted core concept to gender.

2.3.3 Probabilistic models of categorisation.

2.3.3.1 Prototype models

Rosch (1973) proposed a conceptual representation of a category as a 'prototype'. Unlike the classical view, lists of attributes (in her initial formulation) played no part in the representation. In this scheme a prototype was seen as being some composite of typical members of a class, a type of abstraction. A class had a typicality dimension, and any member of a class had a typicality distance from the prototype. The smaller this distance was, the more like the prototype an item was, and so the more a class member. Hence there was no clear-cut class boundaries. Things can be very good examples of a category, or very bad, or somewhere in between. In this scheme a carrot is a very good example of a vegetable, and a skateboard is a very bad example of a vegetable, and a pumpkin is somewhere in between. Subjects not only find it very natural to award typicality ratings to items, but tend to give similar ratings to other subjects (see Rosch 1975, Table A1).

Posner and Keele (1968) found that patterns artificially generated around a prototype led to subjects classifying the original prototype faster than the generated patterns – this in spite of the fact that the prototype had not actually been seen by the subjects. These results suggested that the prototype itself was being learnt – extrapolated from the seen patterns – as the defining element of the given category.

This initial idea of a prototype being an abstraction or a composite of good class members has problems though. It gives us a bizarre idea of a prototype to imagine – say of 'fruit'; an apple cum banana cum orange et cetera is like no one fruit we know. In fact a composite of different members of a class of homogeneous objects appears to rule itself out of the class.

2.3.3.2 Exemplar models

Rosch attempted to refine the prototype idea as a collection of exemplar items of that class, thus a concept would be regarded as a collection of individual objects, each acting on a single prototype. The disadvantages of this type of representational scheme is that the collection of exemplars would have to contain all the attributes that might be relevant in determining class membership, and such a representation would be very uneconomical due to the redundancy involved in multiple representations of the same attributes. Secondly, which members of the set would be exemplars? Since people have different histories and have had different experiences, might two individuals have a very different set of exemplars for a particular category? This would seem to go against the fact that there is a general consensus as to the membership of categories among individuals (Rosch 1975). In fact the one reason for moving to this exemplar based model is that it enables more attributes to be represented than in the composite model - though at the cost of redundancy. Osherson and Smith (1981) suggested that these considerations apart, the model fails on two counts, namely its handling of complex concepts, and of quantifiers. Arguing that the prototype model was really a variation of fuzzy set theory, their critique of the combinatorial functions of fuzzy set theory (see Zadeh 1965) shows that Rosch's account of simple concepts - such as 'striped', and 'apple' - cannot be extended to handle more complex ideas - such as 'striped apple'. According to fuzzy set theory, the degree to which the predicate *striped apple* is true of an item, is the minimum of the degrees to which *striped* and *apple* are true of that item. In this example, however, it seems that a partially striped apple is firstly, unlike a prototypical apple, secondly, a poor (being partial) candidate for stripiness, but a good candidate for 'striped-appleness'. A similar point can be made for (supposedly) empty-set concepts, such as *apple-and-not-apple*. If anything can have a non-zero and non-unity measure of appleness (a pear perhaps), then the value of *apple-and-not-apple* will not be zero (implying not an empty set). This criticism applies to any model where membership functions are analogue. Osherson and Smith suggest that the failure of any one model illustrates the different purposes for which we categorise things (see section 2.3.2).

2.3.3.3 Feature list models.

Rosch finally moved to a feature list model (Rosch and Mervis 1975). The difference between this and the classical approach is that Rosch viewed each attribute as contributing to the likelihood that an item with that attribute is a member of that category, without making the attribute necessary for membership. Members of a class tend to share some attributes, but not others, and indeed share some attributes with items that have a very low measure of category membership – non-members in the classical view. Between two contrasting classes there would be an overlap of attributes. This was found to be the case in empirical studies (eg. Rosch and Mervis 1975). Also, subjects were asked to rate attributes not just in a binary fashion (true or not true of a class) but as to how important the attributes were to the category. An item could then be given a degree of membership of a category by reference to not only what attributes of the class it had, but to how strongly each attribute was associated with each class. This model is similar in function to feed-forward networks, in particular the competitive net. In such a network, membership of a category is defined by a summation of weighted attributes, where each attribute is represented by an input node, perhaps with an input signal of 1 representing presence, and 0 absence. A one layer feed-forward network (without a competitive output layer) would give the degree of fit of an input for all its categories (nodes) in the output layer.

In this type of scheme a concept is represented as a summary description of its members. Categorisation is done by assessing a measure of 'match' between this description and the item in question. Membership might be construed as either a match over some threshold limit, or as a graded scale in the range [0, 1]. A common interpretation of such models is of an entity being represented as a point in a multi-dimensional space, with dimensions representing the various attributes. A category is then some sub-space, which in the prototype model would be centred around the prototype, with membership possibly related to some distance function. In a neural networks this type of interpretation is often used to describe the function of mapping an n-dimensional input space to an m-dimensional output space. Where one category is of interest $m=1$, and the output is a measure of membership of that category.

How might one go about implementing such a theory? Rosch (1973, 1975) took typicality ratings of various members of a number of categories, and found that minor discrepancies aside, most test items scored approximately similar ratings for category membership across a number of subjects. One might therefore obtain such a ratings table for members of a category, and perhaps collect a list of attributes associated with each member of that category (as was done by Rosch and Mervis 1975) and give them weights accordingly. This would certainly seem a plausible approach. Might it not be, however, that the numbers do not add up – eg. that a high rated category member has few attributes associated with it (and so these are weighted according high) while a low rated member of a category has these same attributes, as well as some others. What this would amount to is that the attributes are not first-order indicators of membership value. Instead, certain combinations have to be taken into account. For instance, 'fried dough' is a good attribute for a doughnut, as is 'hole in middle', and 'jam in middle'. The latter two do not combine to make a stronger attribute. If something has a hole in it, and jam in the middle then it is likely to be something other than a doughnut, (perhaps a 'jammy dodger'?).

This is the crux of the point Osherson and Smith (*op. cit.*) make. Treating prototype theory as a class of fuzzy logic, they show that combinations of attributes lead to inconsistencies, primarily in calculating the membership value for an item, and secondly in determining truth conditions of tautologous statements. In both these cases an attribute is treated as a simple linear measure. However, we have shown that this need not be so. In another parlance, a class may be non-linearly discriminable in terms of the attributes associated with it. This might be due to our limited understanding of how the attributes operate. We might assume that the more 'holey' an item is the more likely it is to be a doughnut – not so for 'jam in middle' items. This does not mean that there are not a set of (complex) attributes such that the category can be defined in in a linearly discriminable way. In a back-propagation network, these complex attributes would be represented by the units in the penultimate layer of the network – the network from then on being a linear discriminator (a perceptron). Considering a network to consist of a mapping layer (mapping a convoluted input space into a simpler representation) followed by a discriminator layer, implies the penultimate representations are in some sense 'discovered' complex attributes, enabling a (henceforth) straight-forward discrimination to occur. Although it may be desirable for powerful attributes to be pulled out of the given primitives, there is no reason for this to be the case. Unless deliberate structuring of levels of representation is involved in the network learning, it is likely that the 'meaning' of the hidden representations is as opaque to us as the multi-dimensional input space was.

2.3.3.4 Type and token representation in neural networks

Above, (section 2.3.3), representations of categories and individuals are thought of as units, or sets of units acting as outputs of a network. For the sake of discussion categorisation is thought of as an input-output mapping, certain inputs generating certain outputs, these representing the category to which the input belongs. This is, however, a great simplification, if only because entities do not generally fit into one category, except by contextually ignoring all but one aspect of an item. In many cases we do wish to limit the significance of an item to a limited selection of predicates, sometimes only one - is pattern *X* an example of an '*A*', or is it not? In the real world we recognise that the complexity of information presented by a stimulus is best represented, not as a series of binary predicate decisions, but as a hierarchy of such decisions, serving to minimise the information storage requirement by maximising the representation of common correlations. 'Type hierarchies', as they are known in the knowledge representation literature, are able to store information about individual entities - tokens - and yet still retain access to information about common properties of clusters of entities (see Shapiro and Eckroth 1987). Property inheritance is the usual means for achieving this. A property true of some item (an individual entity or a class representation) may be inherited by all items subordinate (more specific) in the tree. The structure of the hierarchy itself carries the relation of the individual instance to the (superordinate) concept. A representation of 'bird' might have 'can fly' as a property, and this could be inherited down the tree so as to be automatically true of all items subordinate to 'bird' in the tree. In this way, every individual species of bird - sparrow, wren, robin et cetera - need not specifically carry information about flight ability. Exceptions to the rule can be built in- penguins cannot fly - thus over-riding the bequest of that property in that case. Such links also carry information about sub and superset relations - a sparrow is an instance of a bird, which is an instance of an animal ... (see Brachman 1983 for various other uses of links).

The representation of individuals in neural networks poses problems for property inheritance. How individual tokens are to be represented so as to enable type information to be accessible is addressed by Hinton *et al.* (1986), where type information is represented by common vectors among individuals. Thus common elements of a vector representing three particular dogs would be those dog-like characteristics (perhaps even 'dogness' itself), while individual differences in the vectors might be information regarding individual variance, such as sex, weight, colour et cetera. A consequence of this is that superordinate tokens (such as mammal, or animal) consist of shorter and shorter vectors. This scheme does not offer the computational advantage that type hierarchies do, since individual items are required to carry all the information that is relevant to them. McClelland and Rumelhart (1986b) posit a scheme whereby prototypes are represented in an associative memory, and these representations are generated automatically through exposure to paradigms by a learning process. In fact, this method appears simply average a set of signals, and is unable to cluster significant groups without being primed by the learning or retrieval process, either by giving a part of a prototype as an input to be completed, or by using a name tag associated with a group of input patterns. In the sense that *any* prototypes are formed, meaningless ones are too, such as any amalgamation of three disparate entities. Robins (1989) attempts to represent hierarchies by defining a centrality measure for a single vector element, representing how co-active a particular unit (or feature) is which other active elements in that vector. Thresholding individual nodes develops a prototypical representation of a set of vectors. Centrality is considered to be a measure of 'cue validity', i.e. how well that feature is as a cue for that class of vectors. Relaxing this threshold enables more of these cues to become active, enabling higher and higher representations to be formed. Again this has the consequence of superordinate tokens consisting of shorter and shorter vectors.

In attempting to perform some particular classification task we have the advantage of being able to ignore many issues surrounding the development of an overall representational scheme. Just as in the visual domain, research has tended to focus on specific problems without attempting to develop entire visual systems, in classification it appears that solutions to small problems can be found without attempting to provide a consistent and complete account of human knowledge-representation. This becomes apparent when one considers more complex, multi-layered problems, in which the development of simple input-output mappings does not suffice. In this thesis classification will generally be considered an abstract theoretical level, without too much regard for the intricacies of complete human cognition.

2.3.4 Similarity of individuals.

2.3.4.1 Similarity theory

The inclusion of an item within a category is typically supported, if not defined, by that item's similarity to some other item. This is most explicit where distance measures to some exemplars are used in the categorisation process (see for instance sections 2.3.3.2 and 2.3.6.4). Even where non-comparative methods are used (e.g. hyperplanes) an item is expected to be similar to at least some other members of its class. When inter-pattern distances are used in constructing categories some threshold measure of difference (say Θ) is occasionally used in determining the category boundaries, so that, for instance, $x_i \in C_k$ and $x_j \in C_k$ *iff* $d(x_i,x_j) \leq \Theta$. Difficult concepts such as the Wittgensteinian example of game, may require a combination of disjunctive classes, giving a class $G = \vee C_k$. The enumeration of predicates in order to calculate similarity between things proves to be more difficult than one might suppose. Consider a set of p patterns, each with n items of predicate information, $\chi = \{x^j = \{x_i^j : i = 1, \ldots, n\} : j = 1, \ldots, p\}$. A simple measure of similarity between items x^1 and x^2 might be the distance between the two corresponding vectors - let the elements of each vector be binary (yes or no) decisions regarding some predicate (affirming or denying that predicate), then the Hamming distance between the patterns, i.e.

$$d(x^p, x^q) = \sum_{i=1}^{n} | x_i^p - x_i^q | . \qquad \text{(eq. 2-1)}$$

corresponds to the number of different predicate values they have. A normalised measure, such as

$$s(x^p, x^q) = 1 - \frac{d(x^p, x^q)}{n} . \qquad \text{(eq. 2-2)}$$

gives the similarity between two items. A consequence of this measure is that

$$s(x^i,x^j) \leq s_0 \text{ .} \qquad \text{(eq. 2-3)}$$

$$s(x^i,x^i) = s_0 \text{ .} \qquad \text{(eq. 2-4)}$$

and

$$s(x^i,x^j) = s(x^j,x^i). \text{ [1]} \qquad \text{(eq. 2-5)}$$

The problem with this simple enumeration of predicate similarities is that there is no straight-forward manner in which to determine the predicates to include. The very choice of predicates initially is value-laden, and infects the classification process with a priori value-judgements. We may wish to include all the information we have access to, and so use every given predicate, that is, every element of a pattern. However, it is necessary to determine a complete set of predicates that are mutually independent and compatible. If, for instance, $y_1 \rightarrow y_2$, then we would not wish the presence of both y_1 and y_2 in two patterns to contribute twice the evidence that the two items were similar as the presence of y_1 alone would imply. The inclusion of every logical predicate gives us one method of ensuring an equal representation for all predicates. Figure 8 illustrates the case of a simple universe with two binary starting predicates, big (or not small) and dark (or not light). (It should be noted that continuous-valued predicates can be encompassed by this account since an analogue value can be represented digitally to any desired degree of precision.) From these two starting predicates, given that there are no further constraints or implications between them, it can be seen that there are four possible states for something in this universe to be, represented here by the atomic predicates α_i, i = 1, ..., 4 (small-light, big-light, *et cetera.*

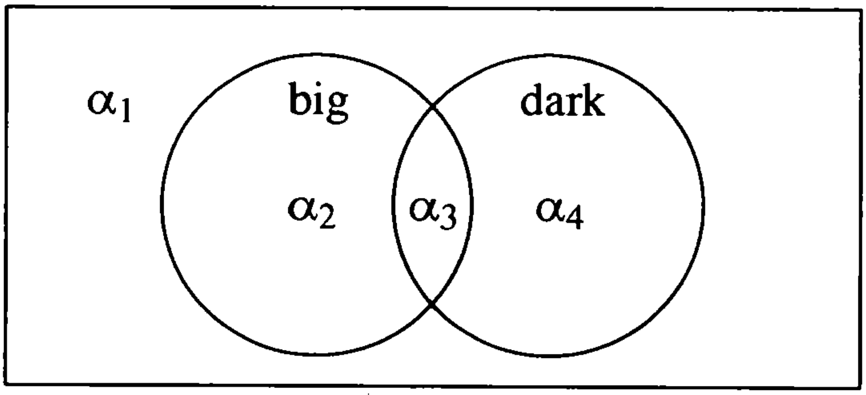


Figure 8: A simple universe, consisting of two starting predicates, big and dark.

---

[1] Given these constraints, then this similarity measure is a true metric if

$$s(x^i,x^j) = s_0 \rightarrow x^i = x^j \text{ ,} \qquad \text{(eq. 2-6)}$$

also

$$[s(x^i,x^j) + s(x^j,x^k)] \times (x^i,x^k) \geq s(x^i,x^j) \times s(x^j,x^k) \text{ .} \qquad \text{(eq. 2-7)}$$

and this latter expression being equivalent to the triangular inequality of Euclidean geometry.

Possible items in this universe might be a golf-ball, an iceberg, an elephant and a beetle respectively. Using only the two given predicates no further distinctions can be made). These predicates are considered to be of rank one, that is, consisting of one single atomic predicate. It can further be seen that there are six predicates of rank two, for example, $\alpha_1 \cup \alpha_2$ (small-light or big-light, i.e. all things light), $\alpha_1 \cup \alpha_3$ (small-light or big-dark), $\alpha_2 \cup \alpha_3$ (big), and so on. There are four predicates of rank three, one of rank four ($\alpha_1 \cup \alpha_2 \cup \alpha_3 \cup \alpha_4 = \square$, the universal set), and one of rank zero, $\emptyset$, the empty set. The rank-four predicate is true of all items in the universe, and may perhaps be thought of as the existential predicate. In contrast, the rank-zero predicate is true of nothing. In this example we have a total of sixteen possible predicates applicable to our universe. Any division of items in this universe - any possible classification - must be in accordance with one of these predicates. Generally, s independent (i.e. no constraints between them) n-state predicates (here n=2, since they are binary predicates) give $a = n^s$ atomic predicates, or possible object types. Predicates can be formed from these basic atoms by combination, there being p^r predicates of rank r, with

$$p^r = \binom{a}{r}\,. \qquad \text{(eq. 2-8)}$$

where

$$\binom{a}{r} = C^a_r = \frac{a!}{r!(a-r)!}\,. \qquad \text{(eq. 2-9)}$$

Hence the total number of predicates is

$$\sum_{r=0}^{a}\binom{a}{r} = 2^a\,. \qquad \text{(eq. 2-10)}$$

There are therefore 2^a total predicates, or 2^a possible ways of discriminating between entities within a universe. These different ways of slicing up the universe are the possible discriminations that can be made.

2.3.4.2 The Ugly Duckling theorem

A possible object, or object type, *is* just an atom according to the above description of a universe (in the above example any object *is* either α_1, α_2, α_3 or α_4. There are just four possible objects in that universe). The various atomic predicates are either true or not true of a class. A class can be defined by its extension (the list of all objects within a class), or its intension (the truth-table of all relevant predicates in a universe). According to similarity theory, a measure of similarity can be derived from the number of predicates with shared truth values between two items. Let us take two objects, say α_1 and α_2 (our golf-ball and iceberg), and enumerate their commonalities. No atomic predicate (i.e. of rank one) is shared by these two objects. One predicate of rank two is, however, namely $\alpha_1 \cup \alpha_2$ (that of being big-light or small-light – in other words, the predicate of lightness). Of predicates of rank three, two are shared by the objects ($\alpha_1 \cup \alpha_2 \cup \alpha_3$ and $\alpha_1 \cup \alpha_2 \cup \alpha_4$), and of course the predicate of rank four, the universal set. In general, of two objects, the number of shared predicates of rank r is

$$shared(r) = \binom{a-2}{r-2} . \qquad \text{(eq. 2-11)}$$

and hence the total number of shared predicates is

$$sim = \sum_{r=2}^{a} \binom{a-2}{r-2} . \qquad \text{(eq. 2-12)}$$

Since the numbers a–2 and r–2 are independent of the choice of objects being compared, it can be concluded that all pairs of objects are equally alike, in that all objects share an equal number of predicates. Watanabe (1985) states this as the theorem of the ugly duckling,

> "Insofar as we use a finite set of predicates that are capable of distinguishing any two objects considered, the number of predicates shared by any two such objects is constant, independent of the choice of the two objects. .... Any arbitrary two objects are equally similar" (Watanabe 1985, p. 82)

For any pair of objects exactly half the truth values of the total logical set of predicates will be shared. This is a consequence of the production of a complete set of predicates from the initially considered simple predicates. The key to overcoming this seemingly impossible result is in the realisation that not all possible predicates are treated equally, i.e. predicates are necessarily value weighted.

An initial, perhaps intuitive, approach might be to argue that the simplest predicates are the most significant, regarding them as the more natural ones in comparison with the seemingly more contrived predicates of higher ranks, such as $\alpha_1 \cup \alpha_3$ (small-light or big-dark) and $\alpha_1 \cup \alpha_2 \cup \alpha_3$ (not small-dark). However, predicate rank does not necessarily correspond to simplicity, since our two starting predicates (size and colour) are both of rank two, while some of the atomic predicates are arguably less simple (small-light et cetera). In fact there are no logical grounds for assessing predicate simplicity in this manner, since a different allocation of atoms to a predicate (perhaps calling $\alpha_1 \cup \alpha_3$ 'foo', and not-foo ($\alpha_2 \cup \alpha_4$) 'bar') gives a different set of classes, big-foo, small-foo, big-bar and small-bar. This re-allocation of starting predicates casts once simple predicates into a more complex form (dark becomes big-foo $\cup$ small-bar). Thus there are no grounds for arguing that one predicate construction is more natural, or more simple than another.[1]

If all predicates are therefore logically equal, some extra-logical considerations must be brought to bear. The most immediate considerations that are forced upon us are those due to our sensory apparatus. Importance of a predicate to an organism is related to survival, and the more directly observed (less contrived) predicates are those that serve the organism best in that sense. Thus it is that given a set of data the simplest interpretations of the data are utilised prior to the more complex conjunctions of elements of the data. This weighting of predicate value gives us

$$d(x^p, x^q) = \sum_{i=1}^{n} w_i \mid x_i^p - x_i^q \mid , \qquad \text{(eq. 2-13)}$$

instead of equation 2-1, where w_i is the importance attached to a particular predicate, or element of a pattern. Selectivity in attention is represented by changes in perceptual saliency.

Two objects x^1 and x^2 can then be defined as belonging to a set C *iff* $s(x^1, x^2) > \Theta$ (see equation 2-2), with disjunctive groups formed between two items x^1 and x^2 *iff* there exists a set of groupings C_i, such that $x^1 \in C_1$ and $x^2 \in C_n$, and $C_i \cap C_{i+1} \neq 0$: $i = 1, \ldots, n-1$.

In the division of an input space in a feed-forward network, s binary inputs can be fed through to 2^S hidden units, each one representing an atomic predicate. A network could then respond to any set of these atomic predicates (i.e. any complex predicate) simply be adopting weightings such that the output cell fires only in the presence of the appropriate set of hidden unit activations. This might however not be the most efficient method of coding the divisions of the input space. Certainly the required number of hidden units grows exponentially with the size of the input space, and regarding continuous input activations as discrete activations can produce a very large number of predicates for a given degree of accuracy. Only if the inputs have a highly non-linear significance – where different discrete activations have entirely different significances – is this quantisation fully justified. More typically, similar activation levels will result from similar patterns, and increasingly smaller changes in activations will have less significance for the overall classification, i.e. the pattern space is smooth, much in the same sense that the physical world is smooth (see Marr's constraint on an image due to the physical world, C2, page 12.)

---

[1] Nelson Goodman's famous grue-bleen paradox (Goodman 1973) illustrates this point well, arguing that 'bleen' - defined as blue before the year 2000 and green from then on - and 'grue' - green before the year 2000 and blue from then on - might seem artificial and contrived, but to grue-bleen speakers our colour vocabulary seems equally ridiculous (e.g. 'blue' meaning bleen before the year 2000, and grue from then on).

2.3.5 Psychological measurement.

Consideration of similarity between things is essential for formulating many theories of psychological behaviour. Classical Stimulus-Response theory requires a metric of similarity between items in order to determine which condition is applicable to a situation. Pavlov, for instance, experimented with variably similar conditions, assessing different pitches of a whistle inducing salivation in dogs (Pavlov 1927). Since then, many attempts have been made to establish reliable gradients of generalisation in a number of fields, e.g. pigeons responding to spectral hues (Guttman and Kalish 1956), shapes and sizes of triangles (Attneave 1950) and Morse code signals (Rothkopf 1957). The use of quantifiable physical measurements (such as pitch, size and so on) as the similarity parameters assumes that the relation between the physical measure and the psychological distance between items is linear – an unwarranted assumption. Non-monotonic relationships exist, for instance in the pitch domain an octave is a natural interval that breaks the linear relationship between pitch and perception. We perceive a middle and an upper C note as more similar than a C and a G, contrary to the physical distance between the wavelengths. Similarly, colour hues are not perceived monotonically (hence we see a number of stripes – although different people may see different numbers of stripes – in a rainbow rather than one continuous spectrum). Psychologists are interested in mapping the relation between the physical parameters and the perception of various events.

Shepard (1966) proposes a method (nonmetric multi-dimensional scaling) of determining the function $f(d)$ that maps the physical distance $d(x_i, x_j)$ of two items to their psychological distance. Shepard sees consequential regions as a result of evolutionary pressure for an organism to survive – "... generalisation is thus a cognitive act, not merely a failure of sensory discrimination" (Shepard 1987). It is the mapping of the consequential regions in psychological space that is of interest here. Of the two functions, *f*, and *d*, for *d* Minkowski r–metrics are generally taken to be the underlying form – where

$$d(x^p,x^q) = \left(\sum_{i=1}^{n} | x_i^p - x_i^q |^r\right)^{\frac{1}{r}}. \qquad \text{(eq. 2–14)}$$

The traditional view has $r = 2$, giving a Euclidean metric (e.g. Shepard 1964). This is generally assumed where dimensions are integral to each other. Where dimensions are considered to be separable $r = 1$ has been taken, giving a Manhattan, or city–block metric (see Garner 1974, and compare with equation 2–13). Indeed, where there has thought to have been a negative correlation between dimensions, values of $r < 1$ have been adopted. It has even been suggested that where the dimensions are separable no distance metric is suitable. Tversky and Gati (1982) provide accounts where the Euclidean triangular inequality rule is violated (as in equation 2–7), implying that there is no appropriate distance metric.

Selective attention can be modelled by supposing that the distance measure of equation 2–14 has weighted significances attached to each dimension, so that the distance measure would be

$$d(x^p,x^q) = c\left(\sum_{i=1}^{n} w_i | x_i^p - x_i^q |^r\right)^{\frac{1}{r}}. \qquad \text{(eq. 2–15)}$$

In this equation, $0 \leq c \leq \infty$, reflecting the overall discriminability in that pattern space. As in equation 2–13, $0 \leq w_i \leq 1$ and $\sum w_i = 1$ (Nosofsky 1986). The weights reflect the amount of attention given to the individual stimuli dimensions. Reed (1972) and Getty *et al.* (1979) suggest that subjects seek to optimally distribute attention across the stimuli dimensions so as to maximise the categorisation performance. Nosofsky (1986) claims that category decisions are based on stimulus similarity to stored examples, that is exemplars, rather than distances to a centroid of previously seen patterns (prototypes). He suggests that the Euclidean metric is the appropriate measure for stimulus similarity (contrary to Shepard *et al.* 1961), and that a Gaussian curve is the appropriate function for *f*, rather than the exponential function usually posited (e.g. Ennis 1988). Staddon and Reid have incorporated these ideas into a neural network model of spreading node activation (Staddon and Reid 1990), suggesting that both Shepard's exponential results (e.g. Shepard 1987) and Nosofsky's Gaussian results (e.g. Nosofsky *op. cit.*) may be reconciled within their theory of the process of activation diffusion between nodes, although without providing a strong framework within which to incorporate a model of classification and/or generalisation.

2.3.6 Formal models of categorisation.

Various schemes have been proposed for constructing the appropriate sub-divisions of the input space, e.g. the sections above on Perceptrons (section 1.4.2) and multi-layer networks (section 1.4.4), where iterative development (and one hopes, improvement) of a randomly selected hyperplane is used to find a solution. All these techniques have in common the use of a number of paradigms, or labelled samples (often called the training set), to fix the appropriate hyperplane in the belief that the paradigms are representative enough of that particular category for a *general* solution to be found (i.e. one where further data not in the training set - the test set - is also appropriately classified). The differences between these plane finding techniques lie in the measures used to assess their performance. Fisher's discriminant and the Least Mean Squares techniques are briefly discussed to illustrate this point, and are contrasted with Nearest Neighbour algorithms.

2.3.6.1 Bayesian theory

The classical Bayesian approach to classification is one which enables an attempted assessment of the probability that some given pattern was caused by a particular cause. Given a particular pattern, we wish in particular to discover which of various possible causes is most likely to be responsible for it, in classifying an image, say, which object is most likely to be responsible for that image. Let x_i, $i = 1, 2, ..., I$, be a set of I patterns, each variously belonging to one and only one of J classes, C_j, $j = 1, 2, ..., J$, each class being mutually exclusive. Various probabilities concerning these patterns can be expressed as below:

$P(C_j) \equiv$ the *a priori* probability that any pattern belongs to class C_j,
$P(x_i) \equiv$ the probability that a pattern is x_i,
$P(x_i|C_j) \equiv$ the class conditional probability that a pattern from class C_j is x_i,
$P(C_j|x_i) \equiv$ the *a posteriori* conditional probability that a pattern x_i is from class C_j,

and $P(C_j,x_i) \equiv$ the joint probability that a pattern is x_i, and is from class C_j.

Under the Bayesian formalism, various conditions apply to these statements of probability. That each pattern x_i is unique, and belongs to one and only one class in C_j is represented by the conditions that

$$\sum_{i=1}^{I} P(x_i) = 1 \ , \qquad \text{(eq. 2–16)}$$

and

$$\sum_{j=1}^{J} P(C_j) = 1 \ . \qquad \text{(eq. 2–17)}$$

It is also held that

$$0 \le P(C_j) \le 1 \ , \qquad \text{(eq. 2–18)}$$

$$P(certainty) = 1 \ , \qquad \text{(eq. 2–19)}$$

and

$$P(C_i \ or \ C_j) = P(C_i) + P(C_j) \ . \qquad \text{(eq. 2–20)}$$

In general, we wish to determine the probability of a given pattern being a member of a particular class. From the definitions given above, it is possible to formulate the joint probability $P(C_j,x_i)$ as

$$\begin{aligned} P(C_j, x_i) &= P(x_i | C_j) P(C_j) & \text{(eq. 2–21)} \\ &= P(C_j | x_i) P(x_i) \ . & \text{(eq. 2–22)} \end{aligned}$$

which enables us to calculate

$$P(C_j | x_i) = \frac{P(x_i | C_j) P(C_j)}{P(x_i)} \ . \qquad \text{(eq. 2–23)}$$

Equation 2–23 is known as the Bayes' relation, and gives us a way of calculating the a posteriori conditional probability if this is not known directly. Generally, the rule used to decide which category a pattern belongs to is as follows: $x_i \varepsilon C_j$ *iff* $g_j(x_i) > g_k(x_i) : k = 1, 2, ..., J, k \neq j$, *where* $g_j(x_i)$ is the decision function for class C_j. Typically $g_j(x_i)$ is the probability $P(C_j|x_i)$ itself, although decision functions incorporating different elements of risk can developed by weighting some classes, so that $g_j(x_i) = w_j P(C_j|x_i)$. This might be required where it is preferable to make some mistakes over others, for instance in a weapons detection system a false positive (a false alarm) might be deemed less hazardous than a false negative (missing an incoming missile).

Assume a given set of p training patterns, $\chi = \{x_i : i = 1, ..., p\}$, and a corresponding set of classifications $L = \{L^j : j = 1, ..., p\}$. Let each L^j have an integer value in the range $\{1,m\}$, to correspond to m different classifications, $C_1, ..., C_m$. If any incoming pattern x_i is *identical* to one of the training patterns, then we can assign it to that corresponding category L_i. However, in most instances it is unlikely that there will be an identical training pattern to the new pattern, and if there was to be, then an infinitely long training phase would be required to ensure that every possible pattern would be accommodated. Instead, the pattern space is quantised into a finite number of cells, with all patterns within a cell to be assumed to belong to the same category, or some continuous-function representation of the discrete (training) distribution function is derived, perhaps by curve fitting. Both these techniques are actually interpolating or extrapolating from given patterns to possible ones. If the number of training patterns were infinite, this determination of the probability density function would be perfect, since it is not, the a priori probabilities of the classes can have a detrimental effect.

Given the predominance of Bayesian theory in pattern recognition, it is natural that neural networks have been proposed as ways of instantiating Bayesian discrimination. Baum and Wilczek (1988) suggest a way of interpreting output values of a back-propagation network as probabilities, and suggest a back-propagation algorithm that uses a log-likelihood measure instead of the usual LMS performance criterion. The claim is that this is a more natural interpretation of activation values, and will lead to a more efficient learning procedure. Anderson and Abrahams (1986) suggest a network in which the states of individual nodes are described by probability densities, and show how learning of pairwise joint probabilities of training data can occur.

Classifying data according to their a posteriori probabilities is an optimal procedure, in the sense that the performance obtained is as good as one can get with that information, However criticisms have been made of the Bayesian approach and its application to pattern recognition. For instance, equations 2-19 and 2-20 imply that

$$P(\sim C_i) = 1 - P(C_i), \qquad \text{(eq. 2-24)}$$

a relation not always true of beliefs (see Osherson and Smith 1981 for an adaptation of this problem to prototype theory). Other criticisms focus on this techniques inability to deal with incomplete or pooled evidence. A general criticism regarding Bayesian logic is that it does not reflect what goes in in the human case. Beliefs, rather than probabilities, seem to be the motivating factors in human decision making, and the consideration and combination of beliefs is not always appropriately dealt with by Bayesian logic.

Overlapping continuous joint probability functions can give rise to a simple threshold where the probability of one class exceeds that of the other. Discrimination in this case is reduced to comparing the given pattern with the threshold value. Generally these discriminant functions are either distance based (as in the Nearest Neighbour procedure discussed below, section 2.3.6.4, or in prototype procedures), or based on the formation of hyperplanes, that is, n-dimensional threshold functions (as in the Perceptron, or back-propagation algorithms). Below are discussed two hyperplane techniques, and a distance-based technique.

2.3.6.2 Fisher's discriminant

Fisher's discriminant (Fisher 1925) aims to find a discriminating (or decision) function directly from the training set of data. This is done by finding the normal to an appropriate $n-1$-dimensional hyperplane in the n-dimensional input space, dividing the space into members and non-members of some category. The equation of a plane is given by

$$\sum_{i=1}^{n} w_i \, x_i + b_0 = 0 \, , \qquad \text{(eq. 2-25)}$$

with the normalised normal given as

$$n_i = \frac{w_i}{\sqrt{\sum_{i=1}^{n} w_i^2}} \, . \qquad \text{(eq. 2-26)}$$

Fisher's method attempts to find such a plane so that the input space is divided appropriately by considering the projection of the training data points onto the normal of the plane. According to the prevailing ethic of this technique, it is considered desirable to maximise the distance between the mean points (or centre of mass) of the two classes, and yet minimise the spread of points within one class. The distance from the origin of a point $x^{(j)}$ projected onto the normal is

$$d^{(j)} = \sum_{i=1}^{n} n_i \, x_i^{(j)} \, . \qquad \text{(eq. 2-27)}$$

therefore the mean distance from the origin of points in a category A is

$$\mu_A = \frac{1}{N_A}\sum_{j \in A} d^{(j)} = \sum_{i=1}^{n} n_i \, m_{A,i} \; . \qquad \text{(eq. 2-28)}$$

where N_A is the size of the training set for A, and m_{Ai} is the mean point of the training set for class A. Maximising the distance between the means of two classes implies maximising $(\mu_A - \mu_B)^2$.

The spread of points in the class A along the normal is given by

$$\sigma_A^2 = \sum_{j \in A} \left(d^{(j)} - \mu_A\right)^2 . \qquad \text{(eq. 2-29)}$$

making it necessary to minimise the quantity $(\sigma_A^2 + \sigma_B^2)$. Thus Fisher's method seeks to maximise the quantity

$$\frac{(\mu_A - \mu_B)^2}{\sigma_A^2 + \sigma_B^2} \; . \qquad \text{(eq. 2-30)}$$

Determining the direction of the normal to the plane in this way will give varying results with different problems, depending on the particular distributions of sets in question.

2.3.6.3 Least Mean Squared error method

For some problems particular assumptions concerning the distribution of the categories paradigms might not be desirable, and instead an attempt may be made to minimise just the classification errors made of the training set, in the hope that the best separation of the training data will successfully separate the general case. The Least Mean Squared (LMS) error (see also section 1.4.4.1) method uses this technique. The LMS method can be applied to non-linearly separable problems (although this will limit its success), and as an adaptive technique performs an iterative adjustment of the hyper-plane (as in the Perceptron, section 1.4.2). The hyperplane is effectively 'pulled' into place by the erroneous training items (those on the wrong side of the plane), by a force proportional to the square of the distance of a point from the plane, hence points far from the plane have a large influence on its placement. Let a function F(x) be defined as

$$F(x) = \sum_{i=1}^{n} w_i \, x_i + b_0 \; . \qquad \text{(eq. 2-31)}$$

and d(x) by

$$d(x) = \frac{F(x)}{\sqrt{\sum_{i=1}^{n} w_i^2}} . \qquad \text{(eq. 2-32)}$$

giving |d(x)| as the distance of a point from the plane. Ideally, $d(x) > 0$ where $x \in A$, and $d(x) < 0$ where $x \in B$. The Widrow-Hoff procedure attempts to find the solution to $F(x) = 0$, difficult to obtain directly, but if it is possible to find F(x) and F′(x) for a given x (where $F'(x) = dF/dx$), then an arbitrary guess at x (call it $x^{(1)}$) for $F(x) = 0$ can be iteratively improved to converge on solution. This is similar to Newton's formula, where $F(x^{(1)}) = y^{(1)}$, and the curve at $(x^{(1)}, y^{(1)})$ is replaced by the tangent, giving an intersection on the abscissa at $x^{(2)}$. Newton's formula suggests making a correction to x such that

$$x^{(n+1)} = x^{(n)} - \frac{F(x^{(n)})}{F'(x^{(n)})} . \qquad \text{(eq. 2-33)}$$

This convergence is not guaranteed to work, and will fail if it comes across a point of inflection. We can adapt equation 2-33 to find the minimum instead of the the zero point. A maximum or minimum of F(x) will equal the zero of F′(x), giving the formula

$$x^{(n+1)} = x^{(n)} - \frac{F'(x^{(n)})}{F''(x^{(n)})} . \qquad \text{(eq. 2-34)}$$

Both the LMS method and the Perceptron convergence procedure are affected by distant (outlying) training patterns then, in this technique, as mentioned above, points far away from the plane can have a large affect on the hyper-plane placement. In the Perceptron method, only the validity of the position of a point with reference to the plane is considered, an incorrectly classified point having the same affect on the algorithm no matter how far it actually is from the plane (see equation 1-1). The next classification method to be considered differs from these two techniques in that only the nearest points to a test pattern are considered. Indeed, no separating plane is constructed at all. This is called the Nearest Neighbour method.

2.3.6.4 Nearest Neighbour methods

In some ways the Nearest Neighbour (NN) technique is appealing in its simplicity (see Fix and Hodges 1952). Given a set of N training patterns, and a classification for each of these patterns, a decision as to the classification of some new pattern is made on the basis of the given classifications of the *k* patterns in the training set *nearest* to the test pattern (the underlying principle being 'judge a person by the company they keep', Dasarathy and Sheela 1977). Generally, these techniques are called *kNN* techniques, and simply a NN technique where k=1.

Given a training set of p, n–dimensional patterns, $\chi = \{x^j = \{x_i^j : i = 1, \ldots, n\} : j = 1, \ldots, p\}$, let their classifications be known and given by $L = \{L^j : j = 1, \ldots, p\}$. Let each L^j have an integer value in the range $\{1, \ldots, m\}$, to correspond to m different classifications, $C_1, \ldots, C_m$. The simplest variation of the method is to classify any new stimulus according to the rule

$$X'\varepsilon C_k \textit{ where } k = L^j \text{ and } [d^{js} = Min\ (d^{is}) : X^i\chi\varepsilon \quad] , \qquad \text{(eq. 2–35)}$$

where d^{js} is the some distance measure between x^j and x^s (Dasarathy and Sheela *op. cit.*).

A significant proof of this class of classifier is given by Cover and Hart (1967), who put an upper bound on the error of this method as the number of paradigms p tends to infinity. For the case of a binary discrimination, where m=2, given a pattern x, and its nearest neighbour x', the probability of an erroneous classification of x is

$$e(x,x') = P(C_1|x')\ P(C_2|x) + P(C_2|x')\ P(C_1|x) , \qquad \text{(eq. 2–36)}$$

where $P(C_n, x)$ is the probability of a pattern x being in class C_n. Equation 2–36 is the sum of the probabilities of the two cases where the nearest neighbour is a member of the other class than is the pattern in question, i.e. An A being classified as a B, or vice versa. As the number of paradigms approaches infinity ($p \rightarrow \infty$) the distance between x and x' approaches zero, and so $P(C_1/x)$ and $P(C_1/x')$ tend towards equality, so

$$\lim_{p\rightarrow\infty} e(x,x') = e(x) = 2\ P(C_1|x)\ P(C_2|x) ,$$

and since $P(C_1/x) = 1 - P(C_2/x)$ in the binary classification class, it follows that

$$e(x) = 2\ P(C_n|x)\ [1 - P(C_n|x)] . \qquad \text{(eq. 2–37)}$$

The Bayesian error (see section 2.3.6.1) at x is

$$e_{Bayes}(x) = Min\ [P(C_1|x), P(C_2|x)] . \qquad \text{(eq. 2–38)}$$

Where $P(C_1/x) > P(C_2/x)$, $e_{Bayes}(x) = P(C_2/x)$, and similarly where $P(C_2/x) > P(C_1/x)$, $e_{Bayes}(x) = P(C_1/x)$, so, in either case, from equation 2–37,

$$e(x) = 2\ e_{Bayes}(x)[1 - e_{Bayes}(x)] . \qquad \text{(eq. 2–39)}$$

The mean Bayesian error, E_{Bayes}, is

$$E_{Bayes} = \int e_{Bayes}(x)P(x)dx \ . \qquad \text{(eq. 2–40)}$$

and the mean NN error is

$$E = \int e(x)P(x)dx \qquad \text{(eq. 2–41)}$$

$$= 2\ E_{Bayes}\ (1 - E_{Bayes}) - 2\sigma^2_{Bayes}\ . \qquad \text{(eq. 2–42)}$$

where σ^2_{Bayes} is the variance of E_{Bayes}. Hence given that the minimum error is the Bayesian, the bounds for E are

$$E_{Bayes} \leq E \leq 2E_{Bayes}(1 - E_{Bayes})\ . \qquad \text{(eq. 2–43)}$$

Cover and Hart (1967) show that for the general case, for any m, (but where $k = 1$),

$$E_{Bayes} \leq E \leq E_{Bayes}\left(2 - \frac{m}{m - 1}\ E_{Bayes}\right)\ . \qquad \text{(eq. 2–44)}$$

hence E is always less than twice E_{Bayes}. This result has been interpreted as meaning that half of the information, relevant to the classification of an item, in an infinite paradigm set is contained in the nearest neighbour. Cover (1969) extended this work to find the bounds on the error for the case where $p < \infty$, giving a result of

$$E_{Bayes} \leq E \leq 2E_{Bayes}(1 - E_{Bayes}) + \frac{c}{(p + 1)^2}\ . \qquad \text{(eq. 2–45)}$$

where c depends on the probability density functions of the classes. In general (where p is large) the larger k, the better the performance of the algorithm is (albeit at some computational expense).

A NN method can be instantiated in a neural network simply by having one hidden node allocated for every pattern in the training set, and adapting the weights such that a particular hidden node responds most strongly to its given input pattern. Intra–layer competition can be used to quench all other hidden layer activations, and the hidden node can be allowed to stimulate the appropriate output category node. Grossberg's 'contrast enhancement' applied to the hidden layer could be used to instantiate a *k*NN model. These models, operating largely in parallel, would be faster than a corresponding search through the training set, but are not economical in memory, acting merely as a look–up table.

Various adaptations of the *k*NN rule (equation 2–35) have been used in order to improve the accuracy of its discrimination. Dudani (1976) used a weighted *k*NN rule, where the influence of a neighbour was inversely proportional to the distance from the test pattern.

The problem to which NN techniques are most prone is that of computational complexity, i.e. the storage and retrieval of the paradigmatic set. Unlike the plane-finding techniques described above (i.e. the Perceptron, Fisher's discriminant, back-propagation and other neural network techniques) which *derive*, then use, a decision surface from the training set, all the NN methods require explicit storage of a number of labelled patterns. The complexity of the derived decision surface in a network model (and hence its computational load) will depend on the sizes of the input and output patterns, and the complexity of the mapping between them (i.e. the number and sizes of the hidden layers). Increasing the size of the training set may have consequences for the length of time required to learn the data, but increasing an already *representative* training set will not necessitate any extra learning expense beyond that of exposing the network to the new data (this is because the additional data will merely confirm the existing decision surface). The then application of the neural network to new data will not have been made more computationally expensive. In the NN case, one wishes to have as representative a dataset as possible while minimising the cost of this storage and retrieval. On one hand it is desirable to have as large a paradigm set as possible, in order to maximise the amount of information known about the distribution of the categories in question. On the other, determining the nearest neighbour of a test item becomes more computationally expensive as the size of the search space increases.

One approach to minimising this computational burden is to edit the set of labelled data in order to reduce its size while retaining its information content. This editing of the paradigm set prior to its use in classification has been shown to be able to improve the classification performance (as well as easing the computational burden) of NN methods. Wilson (1972) edited the paradigm set by rejecting any member of that set which was itself not correctly classified by a *k*NN rule, before using a simple NN rule to classify new patterns. This was shown to give a probabilistic error asymptotically approaching the Bayesian error (and better than a straight *k*NN rule). Hart (1968) using a method called Condensed Nearest Neighbour (CNN) attempted to find the minimal consistent paradigm subset (i.e. the smallest subset that itself correctly classifies the entire paradigm set), but this technique can lead to a large search problem in itself. Swonger (1972) has presented an iterative method of determining the minimal subset. An edited NN method can be compared to the exemplar representation posited by Rosch and her colleagues, in that a minimal number of training cases are sought as being representative of the domain space, sufficient to appropriately classify new data, i.e. to cover the ground, whilst being few enough to enable effective operation of the algorithm.

Editing techniques such as those mentioned above, however, in spite of addressing the problem of the computational burden can only partially alleviate it, since it is inherent in the nature of the method itself. Decision surface methods have the advantage that the geometry of the input-output mapping is directly represented, and the ability of the more recent network learning algorithms (such as Boltzmann learning and back-propagation) to learn more difficult (particularly non-linear) problems appears to be well supported for increasingly complex real-world problems. Given that the appropriate discrimination is possible in the particular n-dimensional space used as the input, it seems intuitively apparent that the output space will divide the input space in a smooth and non-chaotic manner. That is, increasingly small movements in the input space should correspond to increasingly similar outputs. The direct representation of this division thus seems to be the most efficient method of representing the input-output mapping. Rather than just the chosen exemplars, it is the consequential regions around them that are being stored.

2.3.7 Modelling a classification scheme.

For anyone interested in the classification of complex stimuli, it is necessary to know at what level a classification is being done (see sections 2.3.2 and 2.3.3.4). The recognition of animal sounds, for instance, is probably at the identification level, since there is little that an expert can tell us to assist in distinguishing a 'Baaaaaahhh' from a 'Mooooooo'. More complex tasks might require use of a core conception - however, experience in the field of knowledge elicitation tell us that being able to do something does not mean that we know how we do it. To consider a specific problem in pattern recognition - plankton. The task of plankton classification is to classify a specific item as being a member of one class (and possibly more than one, if overlapping classes are allowed) of plankton, or being a member of the non-plankton class. It is not obvious to a novice how the expert does this - indeed, some examples from a single class can differ (to the untrained eye) more than examples drawn from different, mutually exclusive, classes.

One approach, perhaps the most obvious, would be to ask a practitioner how such a classification is done. Again, we cannot assume that they will know this. If it is the case that they do not, two things might be done to cast light on the situation. Firstly, empirical studies of performance on various sets of the data could be done. The speed of classification, the accounts given as to ease of classifying each item (and perhaps some account of why), the perceived certainty of the resultant classification, and the pronounced problems (perceived obstacles) encountered on the more troublesome items (the slowest categorized) might indicate some aspects of the image that are pertinent to the task. Such data might be of use to determine 'good' examples of a particular category, perhaps 'bad' or borderline ones as well. These factors could be reflected in our model by increasing (or decreasing) the significance of those attributes the samples have. This is done in a feed-forward network by increasing the weights in the lines representing those particular attributes. In fact, a typical feed-forward net will do this when a particular sample is more frequent in the input field. The point is that statistical frequency is no indicator of what constitutes a good conjunction of attributes – typicality, from the point of view of a category, is not frequency. Perfect prototypes might be few and far between, but we should not let this fact stop us exploiting them if we do come across them. This may not give the whole story of course. Some problems with an account of such tasks is that much of what is said could be post-hoc rationalisation, ("... this is how I think I did it ..."). A second approach might be to study those items of categorisation that are under dispute. Following the various justifications of the opposing factions may enable one to become more aware of the saliences of a particular specimen.

As this type of classification is so different to the everyday domain it is worth considering whether our currently favoured model (a feature list model with weighted attributes) is appropriate. However once we admit complex conjunctions of disjunctive primitives to count as attributes in our scheme (so as not to fall foul of Osherson and Smith) we have imported our model with such discriminatory power that an arbitrary classification of an input field can be made. In other words, a three-layered network can classify an arbitrarily complex shape in an N-dimensional vector space. The problem therefore is to reduce the size of the input vector space so as to enable a network to generalise successfully from a limited teaching input. It is no good our network correctly classifying, say, only those thousand odd inputs that it has been taught. This amounts to no more than a look-up table. The human skill, which we wish to emulate, is the compression of relevant data into a 'concept', which can successfully be applied to novel inputs.

2.4 SUMMARY

It is suggested in this chapter that formal, mathematical, techniques for performing classifications on a data-set can be compared with models of human classification. In the human case many different types of act might be called classification, and it is clear that no one formal model can encompass all this wide range of behaviours. There are clearly behaviours that do equate with classifying an input on the basis of stored information, and the different consequences of the various formal approaches are of importance for both those studying human classification and those wishing to design effective classifying devices.

Chapter 3. Rationales.

3.1 INTRODUCTION

There have been several attempts to automate the classification of plankton over the past three decades. An examination of published literature in this field is useful, illustrating common problems that one might be expected to deal with, and showing the various strategies that have been used to overcome them.

This chapter also aims to justify the selection of a particular network algorithm and pre-processing technique, both with reference to experimentation carried out early in the project.

3.2 PREVIOUS WORK IN THE DOMAIN

Knowledge of the abundance and distribution of plankton is essential in understanding the role these organisms play as producers in their aquatic ecosystem, their role in the food cycle, and, the importance of which has been more recently recognised, their contribution to the global biogeochemical cycle (i.e. the global carbon cycle). The rapid behavioural response of these organisms to their chemical environmental make them prime indicators of change in oceanic conditions. Depending on the purpose of the data collection, information might be required at a number of different scales. The analysis of nutrient patches of plankton and their effect on other marine life requires data collected at a spatial resolution of metres. At the other extreme, understanding the role of plankton in the global carbon cycle requires broad-scale information from across the major oceans. Technological innovation in plankton analysis has been slow in coming, the most significant changes being in data collection, for instance those due to advances in video instrumentation and enabling *in situ* studies of these organisms. However, the problem of processing this information remains, no matter how it is collected. A recent (within the last 20 years) focus of attention has been on the automatic processing of such data, into which category this current work falls. It is instructive therefore to look at the nature of this task (following section), and to review some relevant contemporary research in this area (sections 3.2.2 and 3.2.3).

3.2.1 Traditional analytical techniques

"Methods of identifying and counting zooplankton in preserved samples have changed little since Johannes Mueller first towed a fine-mesh net through the ocean more than 100 years ago" (Jeffries *et al.* 1980, p. 303). Traditionally, analysis of sampled seawater is done by microscopic observation of a sample in a ruled counting chamber. This technique allows the enumeration of individual organisms within a particular sample, and so an estimation of the biomass of plankton within some area, and analysis on an individual basis, for the purposes of taxonomy and assessment of variability within a population. This 'by hand and eye' method of analysis, whilst being the most accurate at present, has a number of undesirable features. Firstly, the amount of time taken causes delay between sample collection and processing. This makes it impossible to make decisions to collect other relevant data at that time and site, and also slows down the dispersion of results to the interested community. Secondly, this technique is highly labour intensive, and thus costly. Cost is inevitably the limiting factor on the amount of data that can be analysed, and thus costly analysis typically means that less analysis can be done. Thirdly, it is often reliant on levels of expertise which are becoming rarer, and thus more difficult to obtain. This is partially a consequence of less people entering taxonomy as a career. Fourthly, it is arduous and tedious.

Identification of species is initially led by the publication of taxonomic accounts (e.g. Ehrenberg 1839, Jörgensen 1923). As a particular taxonomy gains wider acceptance standard identification keys may be published, (e.g. Larsen and Moestrup 1992). The study of these keys is the standard method by which taxonomists learn the identification of particular species. They typically consist of textual descriptions of visual features, along with photographs of selected individual specimens. For instance;

> *Dinophysis mitra* (Schütt) Abé vel Balech, 1967
>
> **Description:** Cell broad wedge-shaped 70-95 µm long, 58-70 µm wide (dorso-ventrally). Dorsal side convex, ventral side more or less straight in the sulcus region, becoming distinctly concave at the posterior end of the left sulcal

> list towards the antapical end. The epicone is visible as a small, slightly convex "cap' above the cingulum. Theca reticulated.
>
> **Taxonomic notes:** Schiller (1933) indicates that *D. mitra* is probably synonymous with *D. rapa* (Stein) Balech, 1967. The difference between the two species seems to be the degree of concavity of the ventral side. Hallegraeff and Lucas (1988) studied *D. rapa* by epifluorescence microscopy and sometimes found red chloroplast fluorescence.
>
> (Larsen and Moestrup 1992, p.7)

As can be seen from the above extract, a species description is loosely defined by shape descriptors, and differences in degree of some parameter often serve to distinguish two species. Thus the traditional approach to taxonomy is to build expertise through study of identification materials and actual specimens. Assessment of ability is by comparison with acknowledged experts. This process of learning the identification of species can take many years, although relatively inexpert taxonomists can perform more simple (but still useful tasks) such as enumeration of organisms.

Early work on systematising the classification of specimens used manual measurement of various dimensions of individual organisms agreed to be of a particular class. Microscopes with movable eye-piece hairlines are used for measurement in some cases (e.g. Matthews 1967), projections of microscopic images using a camera lucida apparatus in others (Yarranton 1967). Measures such as limb length, curvature and size of features such as teeth are taken, and various ratios between these calculated. To use such a laborious technique many hundreds of times over, to determine population distributions (for instance) in a seawater sample, would require an implausible amount of time and effort. Instead, the purpose of these methods are not to determine the species of a particular individual, but rather to provide a quantification of a visual classification scheme, once an individual has been classified (by the expert eye). This may provide evidence for the validity of that proposed taxonomy, and be used to further define it. This might typically be done in cases where a distinction is sought between two sympatric species, or where a new specification is proposed.

Automation of the above techniques would enable parameters to be taken at a very much faster rate than by hand, and thus it would be possible to accurately analyse vast numbers of individuals using morphometric techniques. Such analysis done as a matter of course would allow much more accurate analysis of populations in general.

3.2.2 Automation of morphometric analysis

The most amenable of tasks to automation is the most simple, that of counting the number of specimens in a sample. Electronic devices such as the Coulter Counter are now widely used in recording particle size of samples (see Sheldon and Parsons 1967), but are unable to distinguish between plankton of a relevant class and other particles in the sample. Further analysis of samples usually depends on some method of presenting an image to an analyser. The availability of cheap microprocessors has made digitisation, storage, retrieval and subsequent analysis of images possible (Campana 1987). Enumeration of specimens in an image can be done with a fair degree of reliability, and most attempts at developing such systems have analysed samples in terms of specimen size-frequency distributions also, an important indicator in population dynamics (see Mackas *et al.* 1981, Rolke and Lenz 1984, Brown *et al.* 1989 for examples). Slightly more sophisticated techniques can determine specimen area as well (Van Wambeke 1988).

The two main problems in classifying specimens in images of sampled seawater are the segmentation of the image, that is the separation of each specimen from the background and detrius in the image, and the classification of the segmented individual.

3.2.2.1 Segmentation of the image

This task might be considered to be preparation of the image prior to the use of a classification technique. Ishii *et al.* (1987) describe a system that isolates specimens in an image by compensating for the background, using a blank example of a photographed background slide to calculate deviations from this image. This relies on a set of images being taken in carefully controlled conditions. If this can be done the segmentation of foreground objects is possible automatically. Ishii *et al.* (op cit.) were able to rotate these segments to a common axis, and use these images for measurement. Operator control was found necessary for the preparation of suitably clean images, with close attention required to ensure that no specimens touched or overlapped in the image, these cases being beyond the ability of the automatic measuring device.

Jeffries *et al.* used published photographs, manually traced specimens and stained samples initially (1980), and latterly attempted to produce clean images from unstained samples (1984). In the earlier work hand-traced images were used as an input to the image processing software, and thus more reliable measurements were able to be taken at the expense of using a manual technique to clean up the images, as has been done in this currently reported work. Stained samples were able to be thresholded with manual control, and automatic counting performed thereafter. Viles and Sieracki (1992) have used staining, with fluorochrome, to enhance the contrast and thereby improve the accuracy of the automatic thresholding and so reduce measurement error, although such techniques do not necessarily reduce image noise due to clutter. In the latter work of Jeffries *et al.* (1984) manually controlled lighting was required to maximise the contrast between back- and fore-ground, to provide clear, measurable silhouette images directly from specimens. Once done, segmentation was automatically performed, although in the reported work the limitations of this technique in the case of partial occlusion are not discussed. Jeffries *et al.* (1984) point out that their system has an inherent problem with dealing with images at differing orientations. Rolke and Lenz (1984) use an automatic technique to maximise image contrast, which then enables automatic thresholding. Again, clumping of organisms was a problem, and manual separation of specimens was ultimately used.

3.2.2.2 Classifying the single specimen

The automation of the traditionally laborious measurement of parameters is clearly desirable. The development of a system that can provide those specified dimensions of a specimen that have been traditionally measured manually, is not yet at hand, although some progress has been made. The system developed by Ishii *et al.* (op cit) can take various measurements (horn length and body width *et cetera*) of an image of an isolated and rotated specimen (as described above). These measures were not as accurate as those made by hand, generally differing by up to 20% from manual measurements, with an average error of 10% (Ishii *et al.* 1987, compared to ± 3% found by Rolke and Lenz 1984). Much of this error is due to the rather-unpredictable automatic thresholding of an object from its background. Colour components of an image have also used by Ishii *et al.* (1987) to attempt to classify species, without reported success.

Jeffries *et al.* (1980) reports discrimination of various taxonomic groups was obtained at levels of around 90% over 101 images. The taxonomic groups were broad, for instance consisting of copepods, fish eggs, lavae and particulate debris, many discriminations of which can be made on size alone. No separation of closer groups, such as individual species, was attempted. Subsequent work (Jeffries *et al.* 1984) on more realistic images (less manually processed) was able to extract features such as length and area, distance between edge points, area moment invariants and Fourier descriptors of outline. These features were used to distinguish images with a reported success rate of 89% over a sample set of 500 individuals, again according to rough categories, not species.

Rolke and Lenz (1984) point out that the sample preparation time at present accounts for most of the recognition time of these systems. More recently, Jeffries' research has focused on the mechanics of presenting a clean, uncluttered image for subsequent image analysis (see Jeffries *et al.* 1984). This development would be generally useful in the domain, since it is in the nature of samples of particles in suspension that there is undesirable clutter. A robust mechanical method for the separation of individuals, and control over orientation and aspect, would enable a concentration of image processing techniques on the specimen in question. It is because of this prior problem that much work has been done using images cleaned in some manual way (for instance Jeffries *et al.* 1980).

3.2.3 Other automated techniques

Flow cytometry is widely used in the analysis of samples, and consists of the rapid measurement of particles in a stream of fluid. Typically, measurements are taken of individual cells' responses to light in the form of a focused laser, where particles are in the 1-150 μm size range. This technique is used to quantify and sort populations of plankton. The ability to deal with very large numbers of individuals (around one thousand per second) means that rigorous statistical analysis can be done. Simultaneous multiple measurement of properties can be done. Light scattering, emission and fluorescence can be used as a cell 'signature' which can then be used for the purposes of classification. Although primarily a laboratory technique, flow cytometry has been demonstrated in use on board ship in spite of the seemingly hostile conditions to accurate laser alignment (vibration and motion effects, see Olson *et al.* 1985). Discrimination of a species from clutter has been shown possible using light scatter (both the magnitude and polarisation properties of cells using both forward and right-angle scattered light) and autofluorescence emission and excitation (Olson *et al.* 1989), although this abundance of information has resulted only in enumeration of individuals and rough taxonomic classification, primarily by size and presence of pigments (Olson *et al.* 1989).

Fourier descriptors have been used in conjunction with automatic thresholding by Steidinger *et al.* (1990). No reference is made to the type of images used as data, but Fourier descriptors of each specimen outline are taken and used to separate two species. No success was reported in this technique, and Stedinger *et al.* conclude only that larger than obtained sample sizes were necessary to accurately classify a species by its Fourier descriptors, suggesting that around 300 sample specimens were needed for accurate discriminatory analysis.

3.2.4 Summary of previous work in the domain

Automation of traditional manual techniques has been a focus of research activity. Whilst it is possible to quickly and accurately count the number of particles in a sample, the classification of them individually remains troublesome. There appears to be two problems, firstly that of producing a clean uncluttered image to classify (also see section 3.4.2), and secondly the classification itself.

The former problem has often led to the use of manual intervention, such as an operator thresholding an image by eye (Ishii *et al.* 1987, Jeffries *et al.* 1984), or the prior manual manipulation of image capture conditions (Jeffries *et al.* 1980). Staining is also widely used, adding to the process time (Jeffries *et al.* 1980, Viles and Sieracki 1992). Where an automatic method of image capture and thresholding has been used manual intervention to prevent clumping was needed (Rolk and Lenz 1984). It can be concluded that no fully automatic technique for the separation of the organisms in an image has yet been reported, and it is because of this that many researchers have resorted to artificially clean images to work on (Jeffries 1980, Ishii 1987 and the currently reported work).

The problem of classifying the species has used several different types of data. Flow cytometric measures are increasingly being used for large-scale analysis of populations, but are not yet able to distinguish close species. Classification of a clean image has generally relied on morphometric measure such as length, width, area and various ratios thereof. This is undoubtedly due to these being historical the measures used in analysis of specimens. Other, more complex measures have been used such as Fourier descriptors of outline and area moment invariants, but these are seemingly as difficult to classify by as the more direct measures.

Given the nature of the problem, in that the classification task is difficult to codify explicitly, and yet example classifications are obtainable, it is surprising that neural networks have not been applied to this domain before. There is some research at Plymouth Marine Laboratories into the use of neural networks for the analysis of flow cytometric data, but no results have yet been published (Burkill, personal communication). This thesis reports on an attempt to classify clean images of plankton with reference to their spatial frequencies, and shows that neural networks are able to be trained to distinguish morphometrically similar species from these data. No work has been previously published in this area that uses such a technique.

3.3 NETWORK ALGORITHM SELECTION

There are a vast number of network models currently in use, each with their own distinguishing features and properties. It is unfeasible to try every published network architecture and learning rule in an attempt to find the one most suited to our domain. A classification can be made of networks according to the amount of supervision they receive in training, a broad distinction being made between those networks that receive some, and those that receive none (Simpson 1990). Networks of the former class are referred to as *supervised* networks, the latter classed as *unsupervised*. It was considered worthwhile to experiment with both these classes of network.

3.3.1 Unsupervised networks

Given an input pattern, an unsupervised network will produce a set of output activations representing the category of the pattern. Generally, an unsupervised network will class like inputs as like, and largely differing inputs at different classes. There may be a training period in which network connection weights are allowed to change until the network is performing satisfactorily, or until the network has been subject to all the possible input patterns. The competitive network (Rumelhart and Zipser 1985) is the classic unsupervised network and will serve to illustrate the principles of this class of device.

3.3.1.1 The competitive network

Competitive learning networks are unsupervised, that is there is no error-correcting teacher, and specific responses to stimuli are not taught. The idea of unsupervised learning in a network was applied by Rosenblatt to a simple perceptron (Rosenblatt 1957, see section 1.4.2.1 for a fuller description). Rosenblatt wished to allow the perceptron learn "spontaneously" from its input stream so as to provide a dichotomy in which members of one class were (in some way) more similar to each other than to members of the other class. His first attempt at a learning rule that might result in such behaviour was the *C rule*; weights on active input lines were increased if the input pattern fell within the '1' category, and decreased if within the '0' category. The hope was that a dichotomization would occur, with similarity of patterns within a class being in virtue of those active input lines that were common to the patterns of a class. However, in time the weights on some lines would grow so large that even one active input bit to that line would result in a sufficiently large input to the perceptron so as to classify such that pattern as a member of the '1' set. This would lead to the increase of all other active lines in that particular pattern, so eventually 'pulling' any pattern with activation on any of those bits into the '1' set as well. These patterns, in turn, would capture any patterns which overlapped with them, which would capture any patterns overlapping with them, *et cetera*. This results in all patterns that overlapped any other patterns in the set being classified as '1', which can amount to the complete input set, unless the input patterns are extremely sparse.

Rosenblatt amended the C rule by limiting the amount of growth permissible on a line, and by requiring any growth to be compensated for by decreasing the weights on inactive lines. This is known as the C' rule, and resulted in stable learning of pattern classifications, where patterns in one set were similar to each other in respect of the input lines that each activated (Rosenblatt 1959). Such a learning rule was incorporated into a model by Rumelhart and Zipser (1985) as the basis of a competitive learning network. This consisted of several sets of input lines feeding into an number of output nodes. A classification is performed by one node receiving the most input when a pattern is fed to all the sets of input lines. Competitive networks are characterised by:

i) mutually exclusive inhibitory clusters at the output layer, classifying the input pattern.

ii) adaptation of weights only when they are connected to a winning (classifying) node.

iii) a fixed amount of weight on connections feeding into a node (see section 2.3.1.2 for a fuller description of these features).

There are of course variations on these constraints that have been made, but they retain the idea of a competitive net.

Grossberg has proved that whilst such a model will learn stable classifications give sparse data inputs and sufficient competitive nodes, it cannot learn a stable classification of an arbitrary input stream. In Carpenter and Grossberg (1987) a set of four patterns are defined that are not able to be stably classified (i.e. that each pattern eventually fire a particular node upon presentation, and no other) by a competitive net. The unstable response of the net to one pattern is due to the change in weights (i.e. 'learning') done upon presentation of intervening patterns. The plasticity of a net which enables learning to occur in the first place also means that unlearning can occur, thus leading to an unstable net. This is the stability-plasticity dilemma. Learnt classifications must somehow be guarded against being washed away by learning some other rogue input pattern, yet the net must remain plastic enough to learn new relevant patterns. Carpenter and Grossberg suggest that Rumelhart and Zipser have avoided this problem by limiting the nets used in their examples to simple stable input environments. Kohonen (1984) suggests that learning in such a net be shut off at some appropriate moment. This requires an external supervisor to choose the moment. Grossberg's model attempts to provide an internal supervisor in the net to dictate when learning is to be suspended.

3.3.1.2 Experiments with an ART 1 model

'Adaptive Resonance Theory' (ART) is a term coined by Grossberg (1976b) to describe a class of neural networks which self-organize and self-stabilize recognition codes in response to an arbitrary input environment. Such networks may be thought of as an adaptation or enhancement of competitive learning networks. Indeed, it was the limitations of competitive networks that provided the impetus for their resolution in the ART networks.

Figure 9 shows the basic structure of the ART 1 network (from Carpenter and Grossberg 1987). In common with a competitive net, there is an input field (F1) feeding through a set of weights to a competitive layer (F2). The two major differences between this model and the typical competitive net (as described in Rumelhart and Zipser 1985) are firstly, the top-down set of weights to the input layer, feeding back from the competitive layer, and secondly the orienting subsystem, which is responsible for supervising the switch between the stable and plastic states of the system. It is the interaction of these two devices that enables the system to detect when learning would be 'unlearning', that is, when to develop a new representation for a pattern rather than recode an previously learnt one. It is this feature of the net that allows the stability-plasticity dilemma to be dealt with.

The pattern input to the system causes a pattern of activation to occur at F1 and is propagated through a set of weights to the competitive cluster at F2. The pattern that is active in F1 is not necessarily that which is input to the system. This is because the F1 layer interacts with any pattern which is the result of an active (winning) node at F2 propagating down through a set of weights (top-down LTM). This pattern is called the learned expectation, and is a template of that pattern which that particular winning node was trained on. Should a familiar (already learnt) pattern be presented to the system then the learned expectation will be a copy of that pattern. This means that there is a match between the input and the expected pattern, and so learning is allowed to occur.

Where the system deviates from the simple competitive network is when an unfamiliar pattern is presented. In a competitive net such a pattern would come to be associated with

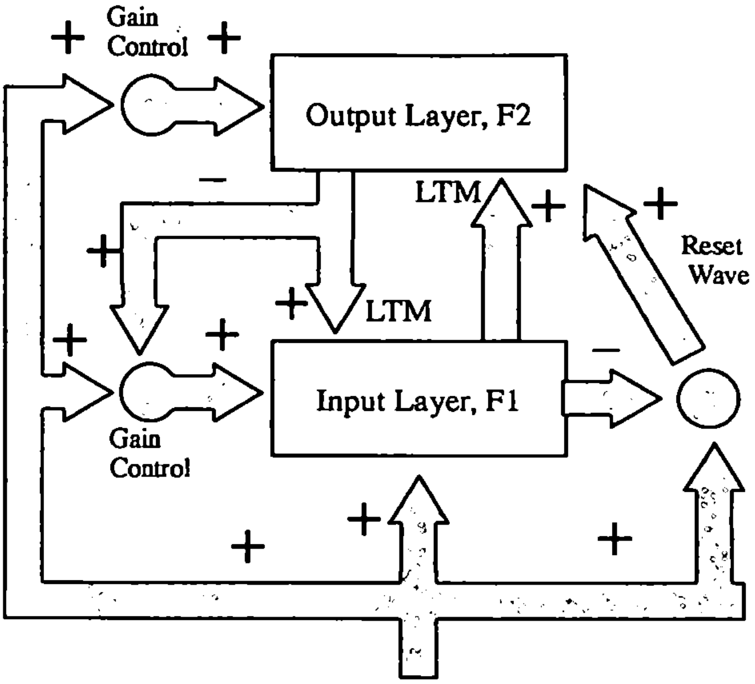


Figure 9: An Adaptive Resonance Network

that category which it was most similar to. This might lead to instability in the system, with a node learning first one pattern and then another and so on. In Grossberg's model such a case is recognised as such by the input pattern 'mismatching' with the learned expectation at the level of F1. If this mismatch exceeds some set parameter then the orienting subsystem selectively inhibits that node firing at F2, dictating that learning cannot take place on this node. This cycle continues until the learned expectation of some node is sufficiently close to the input pattern as to allow it to be recoded, or a node is found that has no learned expectation.

The top-down weights learn according to the associative decay rule for the weights between the competitive node and a non-active input bit, and to the template learning rule. This dictates that weights linking a winning node to an active input bit (at F1) converge to the value of the active bit. These rules together have the result of setting the set of weights running from a node, to the same values as the pattern that node was trained on. Hence propagating from an active (winning) node down the weights will produce a template of the pattern that the node is 'expecting'. It is this pattern that the pattern of activity at F1 is compared with. This comparison is a simple logical AND applied. The effect of this is to retain the activity of a bit if and only if both the input pattern and the learned template are active. Only where the two patterns overlap is activity retained. This means that a large pattern expected, compared to a smaller sub-pattern input to the system, will not cause any mismatch since every bit of the input sub-pattern is matched against an active bit of the expectation. This means that a coding of a pattern can be recoded as a sub-pattern.

One implication of this is that only the invariant features of a category are coded. This is referred to in Carpenter and Grossberg (1980) as critical feature patterning. This can be regarded as a good thing, in that transient irrelevant features will not appear in the coding of the category. But given a noisy signal it is as likely that features will be obscured as it is that additional features will be unwarrantedly added. In the event of a feature being absent from an input, that category with which it is associated will be recoded with that feature missing. If this was to happen several times over the whole pattern the coding would be drastically reduced. For example, if there was a one percent chance of a bit being erroneous in an input pattern of ten active bits, it would only take around sixty presentations (each with its own probability of error) to erode half the information from the coding. This is presuming that such erosion could be taken that far without causing a mismatch such that the system would decide that the input was not a member of its coded category. The fact is that given any amount of noise there are no invariant features of any pattern. This indicates that the net is best used at a higher level of input representation where noise does not occur. There may be no obvious invariant features of a photograph of a person - even a specific person - but there are such invariants of a description of a person - head, neck and torso perhaps.

When applying the ART network to the problem domain, the critical feature patterning led to failure to discriminate the species. Use of binary (or thresholded) input patterns, at a number of different resolutions, failed to let the ART network accurately classify the species. A critical parameter was the mis-match tolerance of the network. Set at zero each image was given a separate classification (no two images being identical). More liberal settings of the network tolerance led to more clumping together of images, but with no accuracy as regards species. Use of more compacting pre-processing techniques was thought advisable to reduce the amount of information the network had to deal with (as discussed below in section 3.4.1). Spatial frequency components were used as input to the network, being represented as histograms. Again the setting of the tolerance parameter was the primary cause for the number of selected categories. Observing this fact, so that the tolerance was set to whatever produced two classes (representing the case of that input set) did not give the desired classification. It is thought that most discrimination was being done according to the overall intensity of the image (and thus the height of the frequency histograms). Further normalisation of the histograms however did not result in correct classifications, patterns being grouped according to a few strong frequency responses.

The lack of success with the unsupervised ART model may be considered unsurprising if we consider what it is that we wish the network to do. The network was allowed to cluster the input patterns only with regard to their similarities, and these similarities consist only of the number of similar bits in the input pattern. We have regard to the content of the images, specifically the shape of the major element in the image. There are, however, many different ways in which one could consider a set of images, for instance orientation of the form, contrast, mean image intensity and so on. The pre-processing extinguishes many of these measures while making others more explicit. For instance, the spatial frequencies are *implicit* in the image, but made explicit by the Fourier transform. If an ART network is classifying bit patterns of frequency spectra then only first-order information will be used for the clustering. On reflection, it seems implausible that the network will extract the classes we desire without being explicitly assisted to, i.e. using our labelling to drive the network. Therefore at this stage it was considered desirable to investigate supervised networks that would aim to replicate given responses to a pattern, rather than hope an unsupervised network can develop the classificatory scheme we desire.

3.3.2 Supervised networks

Since we have access to the desired pattern classification, it seems that teaching a network to emulate our scheme is a more sensible approach than assuming that such a scheme is apparent in the data and will be developed by the network. A supervised network is trained with a sample of data, and given the correct response to that data. The effect of successful learning is for a network to replicate the given responses for any particular pattern.

Replication of outputs for a series of targets could be achieved by a look-up table. Rather than behave like this, it is hoped that a supervised network will come to develop pertinent representations of the data such that its behaviour on exposure to novel patterns is not arbitrary, but somehow an extrapolation or interpolation from the learnt data. In other words, the network develops some conceptual scheme that works for the training data, and by virtue of its general applicability works for novel data as well.

There are many types of supervised network, but the back-propagation network is probably the most well-known and widely used of these. Which supervised network we use for the experimentation is not crucial, since we are more interested in the task performed and the nature of the networks' solution than we are in the learning rule that was used to find this. The back-propagation algorithm has been shown to be of use in a large number of domains, and so was considered a suitable supervised network for experimenting with.

3.3.2.1 The back-propagation algorithm

Generally, the expression 'back-propagation' refers to a learning (in this context, weight adaptation) algorithm, and is applicable to many different network architectures. Previously published experimentation in a wide variety of domains had shown that the back-propagation algorithm is in principle able to learn many different input-output mappings. The inputs may be binary or analogue, as may the outputs. Since we are interested in classifying mutually exclusive classes a local output representation of the classes was used, giving output patterns of [0 1] and [1 0] in the two class case, a scheme that is readily extendible to more classes. Widespread use of back-propagation facilitates comparison of experiments in other domains with network performance in our domain. While there exist a plethora of techniques for improving upon the performance of this algorithm it was felt that experimenting with the most basic version (the so-called 'vanilla' flavour) was likely to make for more robust conclusions as to the general utility of this class of network. It was decided that the back-propagation algorithm represented the most tested learning rule, and made it suitable for use in these trials.

3.3.2.2 Feed-forward networks

A common use of the back-propagation algorithm is in feed-forward networks. The feed-forward back-propagation network is an extension of the simple perceptron model, and the learning algorithm is a generalised version of the delta rule, enabling non-linearly discriminable functions between inputs and outputs to be mapped (through the use of hidden units, see Chapter 2 for a full discussion of this). The networks used in these experiments have a simple forward only layer-to-layer connectivity, making analysis of network solutions easier than in networks with recurrent connections. Feed-forward networks also have the advantage of being faster than recurrent networks to settle when simulated in software, and in hardware implementation. If a certain network is desired to be used for some purpose permanently then it can be build as a hardware implementation. This, in conjunction with a suitably fast pre-processor means that very fast total processing times can be achieved, an advantage in this domain where very large numbers of samples may need to be identified.

Having decided upon a network architecture and desired outputs the question of input representation must be resolved. This is discussed in the next section.

3.4 SELECTION OF PRE-PROCESSING

3.4.1 The need for pre-processing

Taking a grey-level image as our basic information upon which a classification is to be performed, we may deduce that many features which are apparent to the human are not explicit in the image. It is the task of image processing to make explicit that which is implicit in an image. A grey-level image is not suitable for direct input to a simple neural network for several reasons, primarily the amount of data an image consists of and the opacity of the representation of form, by which is meant the difficulty there is in formally describing a shape in terms of image intensities. A single form can be responsible for a vast number of different images, due to the many environmental factors that can affect the image. Such problems require that an image be converted into a more manageable set of data before being presented to a network as an input. It is not our intention to build a simulation of a low-level visual system, so selection of a suitable pre-processing algorithm is permissible, regardless of biological considerations.

3.4.2 Pre-processing techniques

The previous work on plankton recognition has inevitably had to pre-process raw images in some way before using their various discriminatory techniques to analyse the image. A common approach is to initially simplify the image in some way, typically by converting a grey-level image into a binary (black and white) one, thus greatly reducing the amount of information contained in the image. The most simple approach to this is the least automatable, and consists of manually tracing perceived forms to produce a simpler representation of the specimen, retaining (subjectively) significant features at the expense of noise clutter, or unnecessary detail (Jeffries *et al.* 1980, and the current study, see section 4.4.3). A slightly more elaborate method is to perform this highlighting automatically, such as thresholding techniques (Van Wambeke 1988, Campana 1987 and Ishii *et al.* 1987), or to use a more controlled image capture process in an attempt to minimise the noise in the image from the start (i.e. Jeffries 1984, and the staining of cells by Viles and Sieracki 1992). It seems that a combination of clean image capture and intelligent thresholding will be the the likely technique for obtaining images suitable for further automatic analysis.

Once a suitably uncluttered high-contrast image is obtained there is the question of how to quantify it in some way that is amenable to some chosen discriminatory function. Morphometric measurement as described above is obviously inspired by the traditional manual methods of qualifying taxonomic decisions. One of the benefits of the measurement of explicit features in an image is that the results are meaningful to us, and can reside alongside historical data. Other measures, such as area moments and Fourier descriptors, are opaque to human expert pattern recognisers (as are cytometric data). This opacity to us does not mean, however, that such data might not be better suited for the purposes of discrimination.

Previous work on discriminating species by morphometric measurement (e.g. Yarranton 1967) suggested that limb angle was of primary importance in the classification of *Ceratia*. Initial attempts were made to derive these measurements directly from the images, but abandoned as being too fragile and slow in calculation. What was desired was a pre-processing technique that could be given an image and robustly produce some parameters that were of relevance. The development of hardware Fourier transform cards suggested the Fourier transform (FT) as a robust and speedy image processing operation. One benefit of the FT is that it gives a description of the whole image, thus making it unnecessary to isolate the specimen in question from the background in the image. This is useful when classifying clean images such as in the current study, but in noisy images this may be disadvantageous. The product of a Fourier transform can be further compacted by ignoring information relating to the orientation of spatial frequencies in the image. This was done by 'collapsing' the FT from a two-dimensional array to a one-dimensional histogram. The effect of this is to produce the same frequency histogram for an object irrespective of its orientation or position in the image. Since we are not interested in the orientation of the plankton cells (in fact positively *disinterested*, see Jeffries *et al.* 1984) this is a beneficial move. It was not known whether angular information in the image would be present in the transformed image. A series of trials were carried out to determine this.

3.4.3 Spatial frequency trials

Yarranton (1967) suggests that the angle of limbs to the body of a Ceratium, along with curvature, and rate of change of curvature of the limbs, serve as identifying features of various species. Hence it was decided that one of these factors, limb angle, would be examined as a possible candidate for representation of a species. This in particular was chosen because it seems to the novice observer the most practicable method of classifying the Ceratium species. Various techniques could be used for extracting such information from an image of an individual specimen, notably the smoothed local symmetries proposed in Brady and Asada (1984). Such a technique would make explicit the relative angles of the limbs, along with their rates of curvature. Trials into the effectiveness of these data as input to a network were made using computer generated images of angles.

The basic data for input to the network was analogue frequency histograms of a number of computer generated binary images of angles. These numbered 30, 10 of which represented an angle of 45 degrees at orientations of every 5 degrees from 0 to 45. Another set of 10 represented an angle of 90 degrees at the same 10 orientations, and the third set of 10 an angle of 135 degrees at the same orientations. The output for these data was to be [1 0 0], [0 1 0] or [0 0 1] indicating one of three angles depicted in the image. The training set was taken from the lowest 16 frequencies of the histograms of the images of the three classes of image, all at 0 degrees orientation. The data used in testing the network was the lowest 16 frequencies from the other 9 orientations of the three angles. This range of orientations was chosen to represent the maximum amount of image distortion due to the tessellation of the pixels (portraying a line at 50 degrees produces the same pixel set as for 5 degrees, only reflected and rotated. Neither of these transforms have a consequence on the final spatial frequency histogram). Fourier transforms were obtained for each of the 30 images, and from these the power spectra (spatial frequency components) calculated.

The network used in these experiments was a back-propagation model, as described in Rumelhart, Hinton and Williams (1986) (see section 1.4.4.2 for details of the network algorithm, and sections 4.2 and 4,.3 for details of the operation of the trials). A three-layered network was used, with input units, one layer of hidden units, and a layer of output units. In the experiments reported there were 15 or 16 input units, 2 to 4 hidden units, and 3 output units. The input layer was fully inter-connected with the hidden layer (i.e. each unit in the input layer was connected to each unit in the hidden layer), and the hidden layer similarly connected with the output layer. Once the network error has been reduced to a satisfactory level, new data is input to the network in the hope that the internal representation developed in the training stage will successfully generalise to new unseen data. The effects of four transformations on the data were investigated;

(i) No transformation of the data. The raw frequencies of the images of the training set were used as input to the network.

(ii) The energy of the frequencies were reduced by the amount of energy of the least energetic frequency. This amounts to eliminating the amount of signal common to each input unit.

(iii) The energy of the frequencies was 'normalised' so that the sum of energy for each individual input pattern was 1.

(iv) The 16 frequencies were transformed into 15 differences of frequencies by subtracting the value of each frequency from that of the neighbouring frequency. This produced a simple gradient of frequencies.

The results obtained showed that the raw untransformed data was unable to be learnt by the network. The root-mean-squared (RMS) error measure of learning initially would fall each epoch of training, but would begin to oscillate rapidly, and would in time become settled at a local minimum of a high RMS error. The transformed data (ii) had the effect of significantly accelerating the learning of the network, but did not overcome the problem of the oscillation of the learning curve. The transformation (iii) also had an accelerating effect on the learning curve initially, oscillation of the RMS error still occurred. Section 5.3 discusses these effects.

Transformation (iv), taking the 'frequency gradient' had the effect of accelerating the initial learning by an order of magnitude. This learning curve quickly settled to a very low RMS error, and did not oscillate but continued to fall, approaching zero. An early rationale for the success of this transform was that the features of the frequency histogram would tend to be enhanced by this process - a simple edge mask was being convolved with the histogram. The frequency gradients retain the information in the shape of the histogram without regard to the absolute values of the frequency responses. Since the absolute values are partly a product of the image intensities, it was felt that this step would produce information invariant across images of different contrasts and intensities. The results of using the frequency gradients were a great improvement on using the raw frequencies themselves. The learning curve descended faster initially, and oscillation was markedly reduced.

3.4.3.1 Results

Further to the network trained on the data described in (iv), the performance of the network on the unseen data was observed. The classification of the angles at various orientations got progressively worse as the orientation moved further from 0 degrees (the training set); $r = 0.88$, $df=26$, $p < 0.01$. When the images are rotated from 0 degrees the square tessellation of the pixels results in a more deformed image. It was considered that this was effecting generalisation performance, since the unseen images were quantitatively different from the training images (i.e. differing in more than just rotation).

A further experiment was done training the network on all the 30 data sets transformed as in (iv). Once again a stable and low RMS error was achieved. The network was then tested on further generated frequency histograms of new (rather than just 45, 90 or 135 degrees) angles at random orientations, both parameters selected out of 360 degree increments. degree. 71 % of the newly generated images were classified as belonging to that category which their angle was nearest to (counted as a correct classification). The cases in which this did not occur were of angles near to 0 and 180 degrees. These were classified in no consistent manner. It is hypothesised that this is due to the 0 and 180 degree cases being straight lines in effect, and not representing an angle at all.

3.4.4 Using Fourier information

The success of these trials demonstrates visual angle classification by a back-propagation network using only Fourier power spectrum preprocessed images. Thus it seems that some angular information is retained in the spatial frequency distribution. One inference from this result combined with previous plankton classification experiments (such as Yarranton 1967) is that the Fourier Transform preserves some information that may be suitable for classifying these images, while giving a massive reduction of data representing the images. A second reason for consideration of this method of pre-processing is that whilst any orthogonal set of functions can be used for the decomposition of signals, the Fourier series is the most simple mathematically, and would seem a natural choice when working with images of natural objects where change in the image is generally continuous. A third reason is that of engineering pragmatism. Whilst the Fourier transform is computationally expensive, and thus slow to perform in software, hardware instantiations have been developed that can work in real-time on a video image. The production of a plug-in board that can do real time FTs means that a pre-processor can be directly fed into a network, which if both are instantiated in hardware will result in a very fast overall processor hopefully able to perform rapid classifications. This is obviously desirable in environmental applications where quantity of data is large.

3.5 SUMMARY

This chapter provides an overview of previously published literature in the field of plankton classification, which suggests that there has been little success in developing a system capable of classifying morphometrically similar species. The issue of network architecture and learning algorithm is discussed, with an analysis of some prior experimentation with competitive learning. Finally the issue of pre-processing is addressed, which as the literature suggests, is a key component of the problem. Experimentation with spatial frequency data is described, leading to the adoption of Fourier analysis as the primary pre-processing. The next chapter describes the experimental methodology in more detail.

Chapter 4. The Plankton Data Set.

4.1 INTRODUCTION

In this chapter the particular problem domain tackled is addressed. The main task attempted was the classification of images of individual plankton into their respective species. This chapter details the obtaining, assessment, and pre-processing of the data set.

4.2 THE PROBLEM DOMAIN - PLANKTON

4.2.1 Ceratium plankton

The task attempted was to develop a network architecture capable of learning to discriminate between populations of *Ceratia* phytoplankton. This particular genus was selected as a suitable candidate for experimentation with because of the general morphological similarity of its species. Much of the existing work on automatic classification techniques have focussed on size of organism as a distinguishing characteristic (e.g. Brooks and Dodson 1965, Cushing and Nicholson 1966 and Parsons 1969), or some other morphological measures ranging from the simple, such as length, width, perimeter and so on, to the more complex such as area moments and ratio of widths (Jeffries *et al.* 1980 & 1984).

Classification of *Ceratia* was considered to be problematic due to the very close morphology exhibited between the species, and the relatively wide morphological variation exhibited by individuals within species. Different species appear even to intergrade. Generally it can be said that the classification of these species by the human expert is due to their experience of the *shape* of the species, and not due to learnt rules concerning some crude parameter such as size, number of limbs, or absence or presence of some explicit feature. Another characteristic of *Ceratia* is that specimens can generally be considered as two-dimensional creatures, consisting of an exoskeleton only a few cells thick. This renders the need for a complete three-dimensional model unnecessary - a great advantage since this is one of the more complicated aspects of visual processing, and can require the solution of a considerable range of problems. *Ceratia* observed on a microscope slide are found lying flat, that is, with their largest profile to the observer. Given their flatness, there are of course two orientations that an individual specimen might adopt. Being translucent due to their extreme thinness, there are no features to be obscured depending on which way up an individual is. Since current (human) practice in identifying *Ceratia* does not involve consideration of planar orientation (which way up they are) it is considered as being irrelevant to the classification problem.

Ceratium species appear to be distinguished by reference to the curvature and orientation of their limbs, although attempts at knowledge elicitation in this domain typically result in inconsistent and contradictory accounts of how a classification is performed. It is generally considered that such accounts are liable to be a result of post-hoc rationalisations of behaviour rather than on a truly introspective ability (Feigenbaum 1977). The extent of the information obtained from the experts is their actual classifications of the data, along with some self-assessment of their confidence in that classification (see sections 4.3.2 and 4.3.3 below).

The sources and pre-processing of the images used in the classification trials with the experts is more fully described in sections 4.4.1 and 4.4.3 respectively. However a brief account here will enable a clearer picture of the expert assessment of the data (as described in section 4.3.2) to be formed.

Examples of *Ceratia* were provided by Plymouth Marine Laboratory (PML). The primary source of data was a collection of line-drawings made by Yarranton in the 1960's. As described by Yarranton (1965) specimen individuals were taken from the Continuous Plankton Recorder (CPR) survey, and hand line-drawings obtained from these by projection using a camera lucida apparatus. The second source of data was in the form of actual slide-mounted specimen plankton, again drawn from samples gathered by the CPR. Images were obtained from these by the use of a camera attached to a microscope, and images from the camera were fed to digital frame-grabber. Homogeneity of these two sources of data was achieved by creating by hand, for each source images, a simple uncluttered outline image. These images were also used in experiments with human identification.

4.3 ASSESSING THE DATA

4.3.1 Ground truth in classifying data

Critical to any problem solving is knowing when a problem is solved. A problem solving system is to be judged by the correctness, or efficacy of its solution, whether it be explicit output or behaviour. A mathematical problem-solver might well judge its own solution by mathematical means - an answer to an equation is very often easier to assess than to come up with in the first place. A sophisticated forward planning robot might be judged behaviourally - did it get around the obstacle and to point B? A classifier should be judged by the classifications it makes - assuming that is that one can come up with some measure of what constitutes a classification. To illustrate this point, consider a sonar classifier, listening to sonar returns and (or so the manufacturers claim) being alert to possible missile signatures in the data. Such a device sits there waiting to recognise such an event, and then signals as much to the operator. How do we measure the performance of such a device - by the number of correct warnings (true positives), the number of correct no-warnings (true negatives), or by the lack of false alarms (false positives) or failures to identify a missile (false negatives)? Of course different situations might require different value weighting of events (in this case it would seem to be better to give a false alert than ignore a missile, although how many false alerts are to be tolerated?) but it may even be difficult to identity what counts as an event. How does one count events of an on-line continuous system? It has been suggested that if a typical missile signal is 1.0 - 1.5 seconds long, a minute of continuous monitoring consists of 40 to 60 events, thus a system giving no response over a missile-free hour is correctly responding to 3000 odd events (and missing three missiles in that time gives an error rate of one-tenth of a percent, a performance matched by that of a house brick). This suggests, among other anomalies, that training a system to detect a shorter signal pattern results in an automatic improvement of performance. (This has actually been put forward as a measure of system performance on DARPA data, see for example Solinsky and Nash 1991).

A system responding continuously to an image stream would raise the same issues. However, we can envisage our plankton classifier as a discrete system responding to a number of separate images, considering each image as a separate specimen, so that a fixed number of images can be presented, and a score recorded. Two issues related to this idea are that of the classification of non-plankton events, and a probabilistic measure of network performance (see section 4.4.5.4). A more immediate problem is that of knowing the answers to the problems that are given to the networks. It is true to say that a large majority of the classification problems considered in the neural network field are what might be called simply defined, in the sense that to humans there are unambiguous solutions to each problem, for example, alphanumeric character recognition, face recognition and speech processing, in which cases the human has instant access to the required answer (being an expert classifier already in that domain). Another class of problem is that in which the solution is known through some other channel than the problem signal, e.g. by generating a sonar return the class of which is known by artifice (taking a return off a known or placed object) or by other inspection (e.g. Gorman and Sejnowski 1988). Alternatively a signal may be synthesized itself, according to some model of the environment, in which case the correct classification is explicitly defined by that which is synthesized. In these cases it is easy to measure system response against the actual solution, or *ground truth*.

It is not that case that all data can be so simply classified. Some data can only be classified by an expert, not being in the range of the average naive human capacity. Much medical diagnostic work is in this area, signals being classified by some expert in the domain. Sometimes this expert assessment can be judged by reference to some other data (for example what is known about the patient), but this is not so in the case in point. There is no readily available ground truth for the classification of specimen plankton, other than the opinion of the experts themselves. Experts themselves are judged by their agreement with other recognised experts. This presents an immediate problem for experimenting with plankton recognition, since the 'solutions' to the problems are not necessarily determined, indeed, there are unresolved problems in this field, and they are the focus of much argument. What was wanted for training the networks was an unambiguous, and reasonably accurate set of labelled examples, a ground truth or benchmark for the data collected.

4.3.2 Obtaining expert judgement

Nine individuals employed by the Sir Alister Hardy Foundation for Ocean Science (based at Plymouth Marine Laboratory) were used to classify the plankton data. The data presented to these experts were the processed binary outline drawings as described below in section 4.4.3. Initially a collection of 110 images had been obtained, and these were thought to consist of roughly 50% *Ceratium longipes* and 50% *C. arcticum*. These images were presented to the experts in order to benchmark the data. In order to assess the efficacy of the experts each was presented with a double set of images, that is 220 images to be classified, so that the consistency of each experts' judgement (on identical images) could be measured. A few examples of these images are shown in Figure 10.

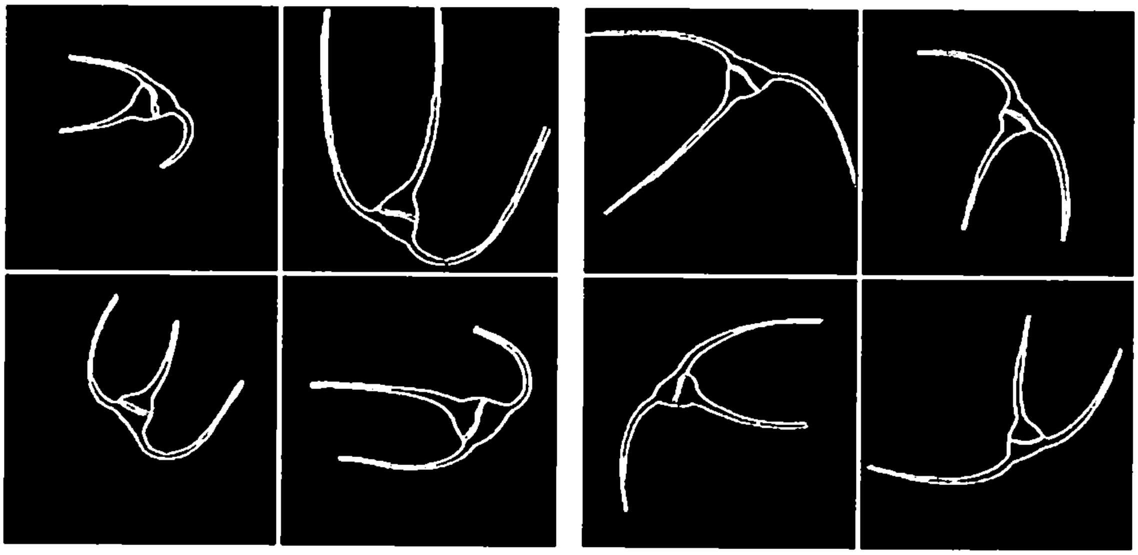


Figure 9a

Figure 9b

Figure 10: Binary images of *Ceratium* plankton; a) *C. longipes*, b) *C. arcticum*.

Each expert was given one set of the 220 images, each set randomised to confound order effects, and asked to classify each image. A key was given so that the more common *Ceratium* species could be recorded with one (initial) letter, and confidence ratings could be recorded between species if the experts so wished. The subjects' instruction sheet for the task read as follows:

Ceratium Classification Task

> This trial is part of an investigation into the classification of Plankton, notably *Ceratium* species. With these instructions should be nineteen sheets of paper, each with a dozen specimens on. We ask you to categorize, or classify, each specimen as a particular species. Given that most of the examples will be *Ceratium arcticum*, *C. horridum*, *C. triposi* or *C. longipes*, the initial letter 'A' (or H, T or L) will suffice for identifying a specimen.Using these abbreviations, please indicate which species you think each picture is of, putting your choice in the relevant box, i.e. for the left-hand uppermost picture, use the left-hand uppermost box, and so on.
>
> Along with each classification, we are looking for a 'confidence rating', indicating how sure you are as to your classification. A specimen which is very clearly *C. arcticum* therefore, perhaps a very typical one, might have a confidence rating of 100 percent, and could be classified as 'A 100'. If this is the case, a simple 'A' will suffice. A specimen which might be *C. longipes* but is more likely to be *C. horridum* could be classified as '40 L / 60 H', for instance. If you cannot decide which of two categories an individual falls into then a rating of 50 L / 50 T , (for *C. longipes* or *C. arcticum*) is appropriate, otherwise the highest rating will be taken as your final classification. There is an example sheet to make this clearer. Note that the ratings on this sheet bear no deliberate relation to the pictures. Thank you for your co-operation!

A copy of the instruction sheet and a specimen set of trial papers are included in Appendix III. The instruction sheet and trial papers were given out and returned between one and two weeks later. No time limit was given, the classifications were done unsupervised, and the experts were able to backtrack and reassess their own work if they so desired. On completion of the task it was found that the experts reported taking between 10 to 40 minutes on the task.

4.3.3 Assessing expert judgement

Once a set of expert judgements had been obtained regarding the data, it was necessary to find some valid consensus of opinion to give a ground truth regarding the classification of each image. Had the each of the nine experts given the same answers for the data then those classifications could have straight-forwardly been taken and used as a ground truth for the data. It was not expected that such unanimous agreement would be found, and indeed it was not. In fact, out of the 110 images classified, only 6 were classified as one species on every classification (i.e. all nine experts classifying the image as the same species on both occurrences, with no less than 100% confidence). Counting less confident decisions (i.e. including those images where each expert was at least 60% certain that it was a particular species), 13 images were unanimously classified - roughly 12% of the total set.

An alternative way to use the experts' classifications is to take those images that are classified to at least some set level of agreement, say a majority, and to use only those images in the database. Such was the level of disagreement in the classifications that only 75 out of the 110 images were able to attract a majority decision. This would have resulted in a much smaller database than had initially been available. The main criticism to be made of this approach is that it regards all the experts decisions as being equally valid. The results presented below appear to be contrary to this assumption.

Since the nine experts used in the trials had widely varying degrees of experience of classifying *Ceratia*, it was envisaged that this would result in a wide range of expertise. An initial working hypothesis was that the more inconsistent an individual is, the worse a classifier they are. The rationale behind this hypothesis was that an expert classifier will rely more on the information in an image to make a judgement, whereas a less competent individual will be less efficient in their usage of this information, and their decisions will be more prone to ill-informed sources of information such as guessing, and the influence of previous images. It might be the case that an incompetent "expert" could formulate an arbitrary rule guided strictly by some artefact of the image (such as "all left-facing images are species 1 and all right-facing images are species 2"), but given the similarity of the images this was thought unlikely. The experts were initially awarded a measure of their individual consistency, calculated as how often they gave the same judgement to both instances of an image. This was the rationale for giving each expert a double set of images. Of the 110 image pairs, the number of contradictions (i.e. labelling the two occurrences of an image in different ways) were recorded and found to be between 3 and 33 (see Table I).

INTERNAL (SELF) INCONSISTENCY (Errors per 110 pairs)									
EXPERT	A	B	C	D	E	F	G	H	I
ERRORS	3	6	7	17	18	19	26	33	33

Table I. Self-inconsistency of subjects on *Ceratium* classification trials.

Given that the two occurrences of an image would be seen between minutes and half an hour apart, it was surprising that such high degrees of inconsistency on the part of the subjects were found. This result also supported the conjecture that the experts had widely varying degrees of expertise. Using this measure of consistency it was found that three of the nine experts in particular were highly self-consistent, and therefore presumed to be the most competent classifiers (sorted by their consistency, and hence labelled in Table I as A, B and C).

EXTERNAL CONSISTENCY (rounded to %)									
EXPERT	A	B	C	D	E	F	G	H	I
A	99	95	95	78	74	53	43	65	53
B	95	97	95	78	75	53	44	66	57
C	95	95	97	76	74	53	43	66	57
D	78	78	76	92	78	63	53	67	56
E	74	75	74	78	92	61	57	70	54
F	53	53	53	63	61	91	58	63	50
G	43	44	43	53	57	58	88	58	52
H	65	66	66	67	70	63	58	85	58
I	53	57	55	56	54	50	52	58	85

Table II. Consistency between subjects on *Ceratium* classification trials.

Further support for adducing the credibility of these experts A, B and C was obtained by using cluster analysis. The working hypothesis was that the more competent experts' solutions would be very similar, while the less competent experts' solutions would be more spread. This is because while one can be right in only one way (the correct classification), one can be wrong in many ways (any incorrect classification). Hence two experts sharing a high level of expertise will produce similar lists of classifications, while two less able experts will produce more divergent results. Agreement between subjects is shown in Table II. The technique to find the best experts is therefore to find the most agreed individuals. This was done using a cluster analysis technique (from Jarvis and Patrick 1973), and results indicated that the three of the nine experts formed a close cluster in response space. Their high level of mutual agreement can be seen from Table II, subjects A, B and C having agreed on 95% or more of the classifications (bounded by a thick line). No other grouping of subjects produced such a high level of agreement.

The improvement gained by considering the classifications of just these three experts is shown by the fact that 98 of the images elicited a unanimous 100% confident response (compared to 13 images for all nine experts), rising to 108 images when including decisions with over 66% agreement (compared to 72 for all nine experts), and all 110 images attracted a majority decision amongst the three most consistent experts (compared to 98). Figure 11 shows for the best one, three, six and all nine experts the number of images eliciting a certain minimum level of agreement. Consider accepting all those images that elicit at least two-thirds majority agreement. For the most self-consistent subject, and considering the best three as a group, there are a high proportion of images with at least this level of agreement. For groups consisting of the best six, and all nine subjects, there are a much smaller proportion of images that attract this level of agreement. Generally, for the most consistent subject, there is much self-agreement illustrated. For the best three subjects (the line marked 3), this drops off a little as one increases the degree of agreement required to consider an item. For six and all nine subjects there is a marked decline in items with higher levels of consensus.

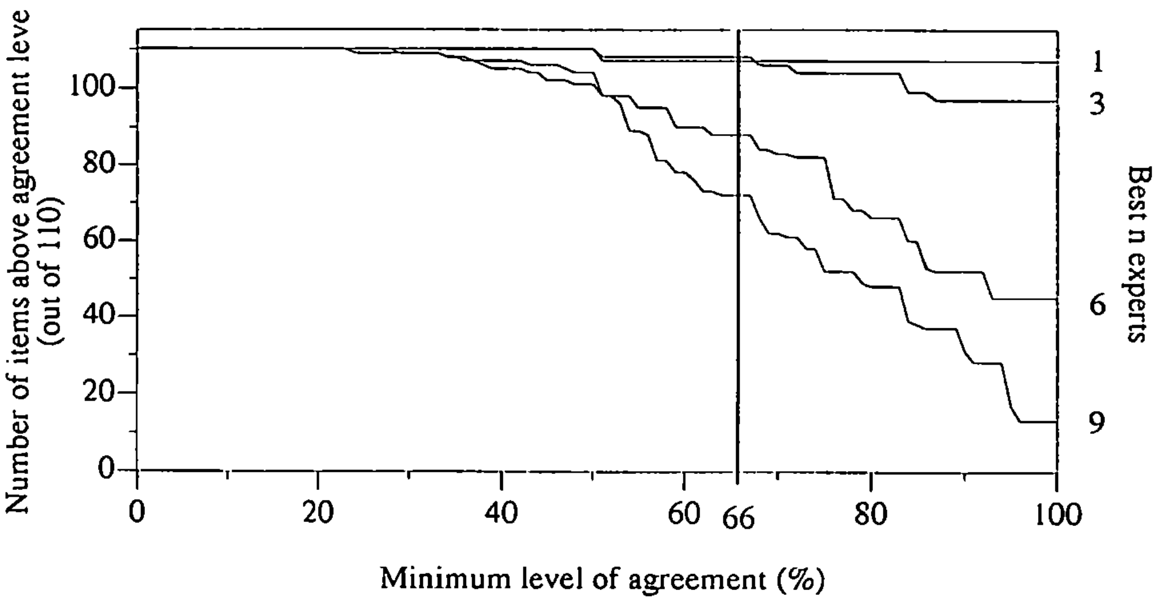


Figure 11: Numbers of images eliciting a minimum consensus.

Having decided upon and elicited responses from the three most consistent experts, their decisions were used to form a database of images that had at least a two-thirds (> 66%) majority classification. This database was then trimmed so that equal numbers of *Ceratium arcticum* and *C. longipes* were represented in it. This resulted in a final set of 98 images, 49 of each species. This was used as the source of data for all training sets and test sets.

4.4 PRE-PROCESSING THE DATA

The sources of the images and initial pre-processing are briefly described in section 4.2.1, but a review of these areas will address various issues not mentioned above.

4.4.1 Source of the images

Both the two sources of *Ceratia* data were provided by Plymouth Marine Laboratory (PML). The data came in two forms, actual plankton specimens slide mounted, and hand-drawn outlines of (different) specimens.

The primary source of data was a collection of line-drawings made by Yarranton in the 1960's, and used by him for the measurement of various morphological parameters exhibited by specimens of a number of species (Yarranton 1965). The procedure for obtaining the line-drawings from actual specimen individuals was as follows. Specimens taken from the Continuous Plankton Recorder (CPR) survey, from the North Sea and the Northern Atlantic Ocean, were mounted in polyvinyl lactophenol with blue ink as a stain. Images of magnification × 500 were obtained by projection onto a bench using a camera lucida apparatus, and the outlines and saddle of the plankton traced in pencil by hand. Thus an accurate and clean rendition of the planktons' outline was obtained, and used for the purpose of making various measurements of shape. The original drawings themselves were used as the primary source of plankton examples for the neural network experimentation.

The second source of data was in the form of actual slide-mounted specimen plankton, again drawn from samples gathered by the CPR. Samples consisting of numerous phytoplankton individuals were examined by experts, and a small number of individual specimens removed and placed on a microscope slide. The individuals selected were chosen on the basis of genera, namely *Ceratium*, and quality, i.e. only whole and unbroken specimens were desired. A small number of such slides were prepared. Images were obtained from these by the use of a charge-couple device (CCD) camera mounted on an eye-piece of a binocular microscope. The camera used was a Panasonic WD-CD 50 with an array size of 768 × 575 and an aspect ratio of 1:1. Images from the camera were fed to digital frame-grabber, a Primographics Limited 8 bit frame store, again with an array size of 768 × 575. Once a specimen image had been captured by the frame-grabber it was then stored on a hard disk drive, and was thus available for future retrieval and manipulation.

4.4.2 Problems with source data

The amount of data available is a very important factor in experimenting with neural networks. In training, networks are able to come to solutions to problems through utilising the statistics of the data, and it is important to have a data-set sufficiently representative of the whole possible data population. Given that even a network with a small number of nodes may have a large number of weights, the data available for training the network must contain sufficient information to set a possibly large number of parameters. It has been suggested that the number of input patterns to a learning network should be of an order of magnitude greater than the number of free parameters to be set. Although this ratio was not reached in the trials described below, it was considered desirable to have as large a data set as possible available for training the networks. However, this consideration acts as a constraint on the size of data set available for testing the performance of a network once trained. This is because a measure of a network's performance on *untrained* data was needed to assess the network's performance in general (i.e. on the world, or actual, population).

Given the limited size of each of the two separate sources of data (images and line-drawings) it was considered desirable to produce one uniform set of data. Doing this also allowed a resolution of a number of problems associated with the two data sets. Firstly, the Yarranton line-drawings were too fine to be easily and clearly picked up by a digital camera, and the drawings also had a lot of clutter on them in the form of lines, arcs, numerals and various other notations used in the original work on the drawings. Secondly, the grey-level images produced from the microscope slides were also cluttered with artifacts such as bubbles, seaweed and other organic matter. Whilst it is easy for a human to pick out the central object in the frame and to ignore the noise, there is currently no robust technique for performing this skilled low-level visual task reliably, and attempting a solution to this problem would be beyond the scope of this project. It is a common and legitimate practise in artificial intelligence research to attempt to solve one part of an overall problem rather than the whole problem, and in this case that was done by assuming that a network could have access to clean uncluttered data. The production of these data is beyond automatic techniques, and is currently achieved by human means. However, other research teams are tackling this problem in a number of ways. Jefferies *et al.* (1984) have been working on the production of clean plankton images from sample sea-water, and a wide variety of work on shape extraction and image cleaning goes on in many domains. It is hoped that the resolution of these problems will enable a more extensive automation of the process of marine plankton identification, from sea-water to statistics. The scope of this project, however, is to classify clean uncluttered data, and the production of these data requires human involvement.

The total pre-processing of the data consists of all transforms of the data prior to input to a neural network. In this example there are two stages to this. The first stage is the hand-production of a uniform set of data. This resolves the problem of having two disparate data sets, and also provides the remaining system clean uncluttered data to work on. This stage of the processing is described below in section 4.4.3. The second stage of processing is that involving only automatic techniques, and requiring no human interference, and is described in the various sections in chapter 5, and from henceforth will be regarded as the main pre-processing stage, and references to the pre-processing will be taken to mean this stage of processing.

4.4.3 Obtaining homogeneous source data

Homogeneity of the data was achieved by creating for each source images, a simple uncluttered outline image. In the case of the Yarranton drawings this was done by simply tracing the plankton outline onto an acetate sheet by hand. Only the outline and where indicated, the saddle, of each drawing was traced. A medium width felt-tipped pen was used giving a line width of between 1 and 2 millimetres. This gave a much clearer image than the original pencil drawings for the CCD camera to pick up. The off-slide images were processed in the same way by first printing out a hard copy of each image onto paper using a laser printer, and again by tracing an outline onto a sheet of acetate. Thus a single set of some 110 outline images were obtained, and these were digitally captured using the same

CCD camera attached to a tripod, and focussed on the images from a distance of roughly one half metre. Ambient room lighting gave sufficient illumination for the camera to clearly pick up the lines, and to further simplify the images hand-crafted thresholding was performed reducing the number of grey-levels from 256 (grey-scale) to 2 (black and white). These binary images were then clipped (reframed) with the object of interest completely contained within the frame where possible, giving a final image size of 256 × 256 pixels. From this point on these images were used as the source images for all simulation experiments, and were also used in the human classification trials described above (see section 4.3.2). It is worth noting the fact that all processing from this stage on is fully automatic and requires no human interference or assistance. Thus the system can be regarded as attempting the same problem as the experts. By testing the experts on the homogeneous binary data, it is shown firstly that the task is possible given the binary outline drawings alone, and secondly that the system is being tested against the same problem conditions as the experts, and hence can be assessed directly against them. By having no further hand-crafted stage of processing (i.e. requiring human skill and ability) it can be deduced that the production of the binary images alone is the only stage of the problem remaining to be formalised as an automatic technique, and that the system is genuinely performing as an unassisted automatic technique for the classification of these (binary) images.

4.4.4 Pre-processing the source images

The source images alone are not suitable for direct input to a neural network for several reasons; Firstly, the sheer size of an image. A typical image might consist of half a megabyte of data. A network that could accommodate such a huge input would present great difficulties in its simulation, primarily that of slowness in execution and adaptation. For a network to converge on a stable state might take many thousands of learning epochs, and each of these would take a considerable amount of time to complete on a typical (serial) computer. Secondly, a single body can project a multitude of different images depending on lighting, position in the frame, magnification of the body and orientation. A change in any of these factors will result in a different image, that is, a different set of pixel intensities. None of these factors however should result in a different classification. For a network to pick up the invariants of such a complex image (i.e. the shape of the body) would be an immense task, and would amount to a low-level visual system. Such problems require that an image be converted into a more manageable set of data before being presented to a network as input. This is the pre-processing stage of the task and consists of finding a suitable representation of an image for use in a network.

4.4.4.1 Fourier analysis

Fourier analysis of the image was decided upon as a suitable pre-processing technique, largely because of the fact that a change of position of a body in an image should not result in a change in the Fourier transform of the image, i.e. it is positionally invariant. Secondly, changes of scale of an object in an image result in a different Fourier transform, but one in which it is possible to compare the shape representation with that of the original image. Thirdly, Fourier transforms represent orientation of an image explicitly, so this information can be removed leaving us with an orientationally invariant transform, i.e. rotation of an object in an image does not affect our final transform. The Fourier Transform (FT) is common in image processing for the above reasons, and various software packages are available to perform it. The equation for the two-dimensional transform in the discrete case, for an image size N×M (with pixels $x_{i,j}$, $i = 0, 1, \ldots, N-1, j = 0, 1, \ldots, M-1$) evaluated at angular frequency values

$$\omega_n = n\left(\frac{2\pi}{N}\right) \quad \text{and} \quad \omega_m = m\left(\frac{2\pi}{M}\right) \qquad \text{(eq. 4-1)}$$

is given by

$$X_{n,m} = 1/NM \sum_{i=0}^{N-1} \sum_{k=0}^{M-1} x_{i,k} \exp(-j\,(\omega_n\, i + \omega_m\, k))\,. \qquad \text{(eq. 4-2)}$$

The FT is very computationally expensive to perform on an image, requiring something of the order of M*M*N*N calculations for an M*N size image. Derivations of the FT are available that are considerable less computationally expensive, called Fast Fourier Transforms (FFTs). They have the constraint that an image must be 2**N squared, but this is easy to achieve by clipping digital images. An FT of an M*N image produces another M*N image, generally with complex pixels in $X_{n,m}$. The pixels are no longer representative of light intensity in the spatial domain, but represent information in the frequency and phase domains. Such a transform is complete, and can be returned to the original image by an

Inverse FT (IFT). The values of elements of the FT are complex numbers, representing both frequency and phase information. Since we are only interested in the frequency domain, we can discard the phase information. This is done by taking the power of the value of the pixel,

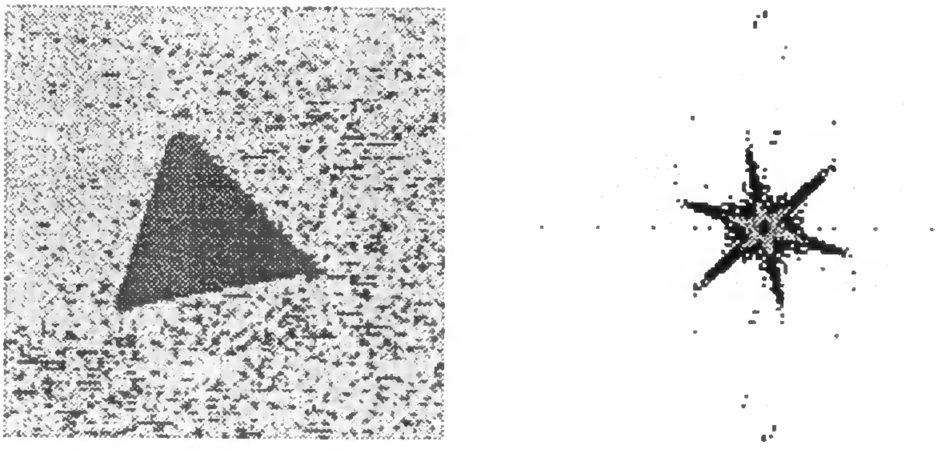


Figure 12a **Figure 12b**

Figure 12: a) A grey-level image. b) Power spectrum of a).

which is simply the square root of the sum of the squares of the real and imaginary parts of the complex number. This results in the power spectrum, which contains the frequency information intact (see Figure 12b, along with the original image in Figure 12a). This power spectrum is shown with the four quadrants reversed (i.e. moved through the centre) so that data from the corners of the power spectrum appears in the centre.

8	7	6	5	4	5	6	7
7	6	5	4	3	4	5	6
6	5	4	3	2	3	4	5
5	4	3	2	1	2	3	4
4	3	2	1	0	1	2	3
5	4	3	2	1	2	3	4
6	5	4	3	2	3	4	5
7	6	5	4	3	4	5	6

Table III. Frequency and orientation information in a power spectrum.

This power spectrum is shown with the four quadrants reversed (i.e. moved through the centre) so that data from the corners of the power spectrum appears in the centre. Each pixel in the power spectrum represents a particular frequency of a sinusoid in the image, at a particular orientation, depending on which pixel it is. The mapping of frequency and orientation to pixel position in the FT is shown in Table III. The 0 in the centre of the power spectrum is that component of the image which accounts for a zero frequency, a fixed value. This is in fact the average value of the pixels in the image, sometimes referred to as the 'DC response'. Around this are the frequency 'bins', representing higher frequencies and their orientations. Due to the nature of the algorithm that produces the FT there are more orientations for a higher frequency to be resolved into than a lower one. For example, a strong frequency of 1 across an image at 45 degrees, will be resolved into the 0 bin and the 90 bin, whereas if the frequency is 2, it will show up in the 45 degree bin. Similarly a frequency of three can be resolved into bins of 0, 30, 60 degrees and so on.

4.4.4.2 Processing frequency information

This is where the orientational information can be removed leaving only frequency information. This is done by summing the various bins of each individual frequency, to produce a total measure of a frequency in an image. This will produce a list of frequency components, and can be represented as a frequency histogram (Figure 13b).

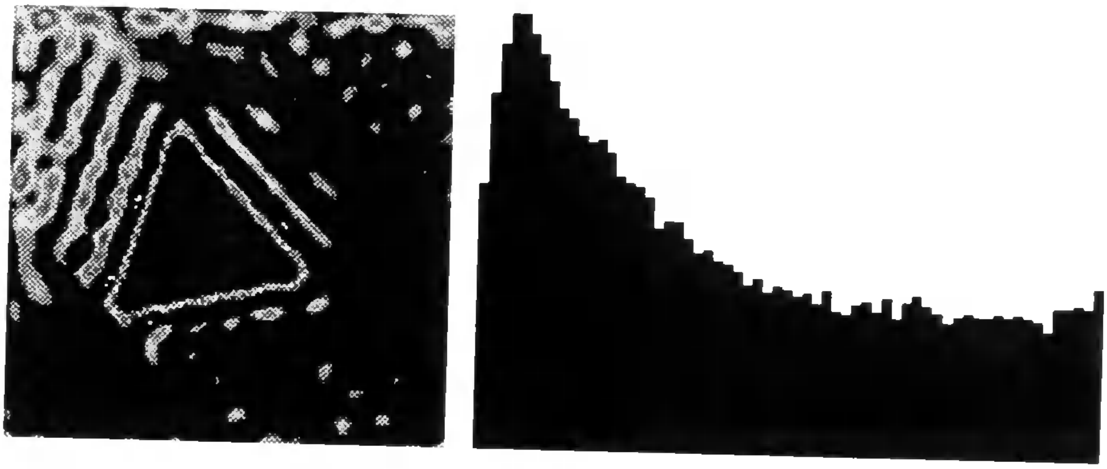


Figure 13a **Figure 13b**

Figure 13: a) Low frequency image of Figure 12a. b) Frequency histogram of Figure 12a.

Discarding the orientational information has the benefit of greatly reducing the amount of data. This is a great advantage for input to a neural network. Such a reduction in size of the network not only reduces operation time in the case of a simulation (taking less time to perform a learning epoch for instance), but also enables the network to learn faster (i.e. in less epochs). There is a further way in which the amount of data input to the network can be reduced. Since we are primarily interested in the shape of objects in our images (in this case), we need only concern ourselves with the lower frequencies in the image. Higher frequencies will be result of small-scale changes in the image, perhaps due to background texture or fine detail of the specimen. As such, the higher frequencies can be ignored. The effect of this can be tested by performing an inverse transform on an FT which has had the high-frequencies removed, to recreate an image with only low frequencies (see Figure 13a). The broad detail of the image remains, and shapes above a certain size are unaffected. The practical justification for this is that if an expert can classify a plankton image with the high frequencies removed, then those frequencies do not play a part in that classification. Images of *Ceratia* processed in this way remained identifiable by the experts, and so this processing step was considered unlikely to remove any information necessary for categorisation from the image, and was also considered a legitimate processing step, not introducing any additional information extraneous to the image.

4.5 SUMMARY

Given the problem domain, it might seem a simple matter to obtain the relevant input data and expert judgement thereof. Analysis of the expert judgements of the images showed the problematic nature of these organisms. Obtaining a benchmark was achieved by considering the most similar and consistent performers. Obtaining the data, however, can really be considered as obtaining a set of data suitable for input to some device (in this case a neural network). The source of the problem itself, i.e. the actual specimens, and thereafter the images of them, is not suitable for input to a neural network. Some pre-processing has to be performed, and the constraints on that are those computational expense and efficacy. It has to be feasible and useful. Fourier transforms were considered a suitable pre-processing technique for the images due to their inherent invariances and rate of data compression.

Chapter 5. Methodologies.

5.1 INTRODUCTION

In this chapter the main methodologies used in the empirical work are introduced and explained. The primary tool of the investigation was the use of feed-forward network structures, using the backward error propagation algorithm for the adaptation of the weight parameters. The operation of this algorithm is described in section 5.2.1.

The actual operation of a complete learning process consists first of the selection of training and test sets from the main body of data, in which considerations of sample distribution and sample size have to be addressed. Secondly, the training of the network given a particular training set (according to the network equations detailed in section 1.4.4.2) is performed, and thirdly the network maturity is determined so as to know when the training is complete, or at least to be halted. These stages of the operation of a network are detailed in section 5.3, "Training the network".

The final section of this chapter is entitled "Testing the network" (section 5.4), and covers the main techniques used in the assessment of network performance on a task. These techniques include measurement of the correlation between network performance and training set size, extrapolation from available training set sizes to possible performance given a larger set of data, and the introduction of a performance measure based on probability theory to enable a more accurate assessment of a network to be possible.

5.2 BACK-PROPAGATION NETWORKS

5.2.1 The back-propagation algorithm

Using the terminology introduced in previous chapters (see section 1.4.4.2), the standard back-propagation algorithm (often referred to as the *vanilla* back-propagation algorithm) is described applied to a network of n layers of nodes, (that is, $n-1$ hidden layers), where each layer is *fully* connected to the layer immediately above it. This means that every node in a layer is connected to every node in the layer above it (see Figure 6). There are variations on this topology but this structure will be sufficient for the purposes of explanation. The dynamical equations of the back-propagation algorithm are given below.

Let a network have p input nodes, defining an input vector $X = \{ x_1, x_2, x_3 \ldots x_p \}$, where each term in the vector is one given input to the input layer, such that the input layer consists of an ordered list of such activations. Let the input to any particular node j be a given x_j for nodes at the input layer, and for hidden nodes (i.e. not at the input layer) let

$$input_j = \sum_{i=1}^{p} w_{ij}\, x_i + \theta \;. \qquad \text{(eq. 5-1)}$$

where w_{ij} is the weight on the connection between node i and node j, and x_i is the activation of the node i.

Although various non-linear transfer functions have been used (e.g. the hyperbolic tan function tanh($\underline{x}$), and sigmoid threshold functions in the range of [-1, +1], see Stornetta and Huberman 1987), the most commonly used is the sigmoid (or "squashing") function

$$sig\ (input) = \frac{1}{(1 + e^{-x})} \;. \qquad \text{(eq. 5-2)}$$

which is conveniently differentiable, and which has an output range of [0, +1] from an input range of [-∞, +∞] (see Figure 5), giving the activation of a node as

$$x_j = sig\left(\sum_{i=1}^{p} w_{ij}\, x_i \right) \;. \qquad \text{(eq. 5-3)}$$

where x_i is evaluated as the activation of the node i, and w_{ij} is the strength of the connection between nodes i and j.

In the training stage of operation, pairs of associated input and output vectors are used to train the network to respond to particular input vectors with particular output vectors. Activation is propagated through the network from the given input activations, through the various weights and connections, to the output layer. The weights are then adjusted to minimise the discrepancy between the actual and desired output. This is done by minimising some cost function, typically the summed squared error on the outputs (but see section 5.2.1.1 for alternative cost functions), formulated as

$$E_p = \frac{1}{2} \sum_{j} (t_j - x_j)^2 \;. \qquad \text{(eq. 5-4)}$$

where E_p is the error over pattern p, where given a pattern p propagated through the network, t_j is the desired (target) output on node j and x_j is the actual output (activation) of node j. Hence the global error over some set of m patterns is given by $E = \sum E_p$. The network error is reduced by updating the weights according to the derivation of the cost function with respect to the weights, dE_p / dw, (hence the 1/2 term in equation 5-4 for easy differentiation). The weights are updated according to the errors on the nodes they lead to, the amount of activation on the node leading to the weight, and some learning rate η. For every layer, this weight change is calculated as

$$\Delta w_{ij} = \eta \delta_j x_i \ . \qquad \text{(eq. 5-5)}$$

where δ_j is the error term associated with node j. For the output nodes,

$$\delta_j = x_j (1 - x_j)(t_j - x_j) \qquad \text{(eq. 5-6)}$$

where t_j is the desired output activation value on node j (i.e. the jth term of the desired output vector). It is however, the calculation of the error term for nodes *not at the output layer* which enables the delta rule to be generalised to multi-layer perceptrons. For such a node, (i.e. a hidden node), the term $(t_j - x_j)$ in equation 5-6 is replaced by the sum of the error terms in the layer above, filtered through their respective weights, giving

$$\delta_j = y_j(1 - x_j) \sum_k \delta_k w_{jk} \ . \qquad \text{(eq. 5-7)}$$

5.2.1.1 Variations on vanilla

An additional term can be added to the weight change of equation 5-5, and that is *momentum* (see Rumelhart *et al.* 1986b). This has the effect of including the last weight change into the equation, so that sharp and rapid changes in direction are minimised, while allowing changes to become increasingly larger as the weights progress in one particular direction. The effect of this is to prevent the gradient descent in error-space oscillating rapidly over some local minimum. It also has the effect of keeping the weight changes going on "plateaus" of shallow gradient. Attempts have been made to find an optimum momentum term by relating it to the learning rate (see Allred and Kelly 1990). This additional term gives the change in a weight as

$$\Delta w_{ij}(t) = \eta \delta_j x_i + \beta \Delta w_{ij}(t-1) \ . \qquad \text{(eq. 5-8)}$$

where β is the variable determining the amount of momentum to use (usually $0 \leq \beta \leq 1$).

As discussed in section 1.4.4.2, this procedure is not guaranteed to converge on a good solution, that is, a global minimum in error space. In practice some coaxing of the network, or data was required to obtain satisfactory performances. These practises are presented as results in chapter 6.

Various techniques to determine a suitable learning rate have been investigated, often resulting in changing it 'on the fly', eg. Allred and Kelly (1990). Cater (1988) uses large learning rates of up to 30, again adjusted 'on the fly'. Jacobs (1988) proposes a number of algorithms to adjust both learning rate and momentum, adjusting each network parameter independently. Other cost functions than the summed squared error have been used, notably by Hanson and Burr (1988), who use a cost function of

$$E_p = \frac{1}{r} \sum_j (| t_j - x_j |)^r \; . \qquad \text{(eq. 5-9)}$$

and show that various values of *r* may be desirable for different tasks. Various transfer functions for the nodes have been experimented with also, for example

$$f\,(input) = -\frac{1}{2} + \frac{1}{(1 + e^{-input})} \; . \qquad \text{(eq. 5-10)}$$

which gives a range of [-1/2, +1/2] (Storneta and Huberman 1987). In general, these practices were not used in the current experimentation, since it was considered desirable to keep the algorithm as simple as possible so as to be able to interpret the results as being due as far as possible to the problem domain rather than some adaptation of the back-propagation algorithm.

5.3 TRAINING THE NETWORK

5.3.1 Selecting training and test sets

In order to test any aspect or aspects of a network, whether it be architecture, transfer function or training algorithm, in some given problem domain, a set of data from that domain is required. The approached generally used is to train a network on some portion of the data, and then test the trained network. In some cases the network may only be required to learn the training data, and a measure of performance over that data is all that is needed. An example of this is the existence proof provided by a multi-layer network learning non-linearly discriminable pattern categories, such as the Exclusive-OR function.

Demonstration of network competence over just the four patterns in the training set provides the proof (see section 6.2.1). It is often expected that a network will be able to learn the various (and sometimes even arbitrary) mappings given in the training stage (with enough hidden layers, nodes, epochs et cetera): this is often regarded as trivial *in theory*, and indeed there exist many statements of this universal ability (see for example Lippmann 1987, Arai 1989).

In many cases however an ability to generalise from the taught pattern classifications is desired, and so the network must be tested on data that are not used in the training stage. This is commonly referred to as *novel*, *unseen* or *test* data. The general hypothesis is that the network's performance on the test data is indicative of performance on the problem in general, i.e. that the test data are representative of the world population of this class of data. This implies two things, firstly that the test data are representative of the *variety* of the total population, i.e. that the distribution of the test data in the state space are representative of the distribution of the population in the state space, and secondly that the state space is sufficiently sampled by the test data. One way to ensure these conditions is to deliberately select a set of data so that they are as representative as possible. For example, selecting certain of a wide range of possible inputs is often done to insure inclusion of particular conditions in the set of training data. For instance, one would not expect a network to discriminate between five different types of input without having experienced at least one example of each in the training set. If one is trying to train a network to diagnose various engine malfunctions from engine vibration say, it is only common sense to ensure that the training data includes each of these. Another approach might be to do some statistical analysis on the patterns, by selecting some features of the patterns, and hence determine the properties of the population. This would be very time consuming, and this is one of the reasons for using an automatic adaptive system such as a network. The most simple approach is to select randomly as much data as is affordable, in the hope that randomly selected data will exhibit the desired property of mirroring the world population in terms of distribution. Obviously in assessing the performance of a network on a world population it is desirable to have as extensive a test as possible. There are limitations on this however, time and availability of test data the most prominent. The test data itself must be classified correctly in order to assess a network, and in domains where this classification requires expert assessment, or is somehow problematic, the quantity of available labelled data might be limited.

In the experiments using plankton data (see Chapters 6 and 7) only a limited set of data was available for use, and the acquisition and labelling of these data is described fully in Chapter 4. A working hypothesis is that this database itself has an adequate sampling of the whole population, and a sufficiently realistic distribution. This hypothesis is largely based on the fact that the specimen individuals were either drawn at random from a sample collected by the Continuous Plankton Recorder, or were selected in order to provide a wide variety of forms. It is hoped that by using as many individuals as possible the distribution will tend to be similar to that of the world population. When performing a trial with a network both the training and test data were taken from this one source. The database in the main plankton experiments consisted of an equal number of two species of plankton (suitably pre-processed). From this a number of training patterns were randomly selected, ensuring that an equal number of each of the two species was drawn. This consisted the training set. The remainder of the patterns were used in testing the system, i.e. were the test set. Utilising all the patterns not being used for training as the test set ensured that the two characteristics considered desirable of a test set (as described above) were being maximised: an as large as possible test set being utilised is likely to result in the best possible distribution of the test data in the state space, and provide the best cover, or sampling of the state space. By randomly drawing the training set (and thus randomly drawing the test set), and performing a number of experiments, it is hoped that any chance "bad draws", perhaps having an unrealistic distribution, will be outweighed by better draws in other trials.

5.3.2 Training the network

For training a network the procedure was as follows: A network structure would be set up with the appropriate architecture, consisting of a designated number of layers of nodes, and a designated number of nodes in each layer. The size of the input layer is determined by the input data, the size of the output layer is determined by the desired output patterns, and the size of any hidden layers is a matter of some choice. Then the desired node to node connections would be made. Each of these would each be initialised with a small randomly generated value in the interval of ± 0.1. The network would then be trained on the training set, with the weights being updated, according to the training algorithm, after every presentation of an input pattern. When the network reached some level of maturity the training was considered complete, and the then current state of the network taken as the final resultant state.

5.3.3 Network maturity; error, weights and epochs

Training a network to solve some problem can be a lengthy business. Large training sets, large networks and slow iterative training algorithms such as back-propagation can all lead to trials taking a significant period of simulation time. Added to this is the fact that the vast majority of network experimentation is done on serial machines, where the advantage of the innate parallelism of networks is lost. In order to keep machine time down to a realistic level, it is necessary to know when to halt a training sequence so that one does not continue training a network indefinitely. Three criteria were investigated for determining when to finish training a network: the error of the network; the change in the network's weights; and the number of training epochs performed.

5.3.3.1 Error of a network

The most obvious criteria of network maturity is whether it is trained to perform the required task or not. So if a network was required to produce an output of (say) 1.0 on some node given some input, training would be complete when that output was produced for that input. There are three problems with this idea: Firstly, in the typical back-propagation network, due to the nature of the transfer function, the nodes are unable to produce an output of *exactly* 1 (or 0), since this would require an infinitely positive (or negative) signal to be input to the node, and hence infinite weights. Secondly, *even if* an output of (say) 0.9 or 0.1 was required, the iterative learning process would only approach the desired values *asymptotically*, and so would never actually reach them; thirdly, if the training set is only a subset of a large (or unlimited) total population of valid inputs (as is the case where a network is required to perform any natural real-world task) then it is unfeasible to test the network on all of these patterns. The first and second problems can be resolved by specifying an acceptable level of inaccuracy for the desired outputs. This is done most simply by training a network until the error over some portion of the data (usually the training set) is less than a set amount, and accepting this inaccuracy in performance. An output of greater than 0.9 (or less than 0.1), say, might be considered sufficiently close to the desired figure of 1.0 (or 0.0), and so be taken as an acceptable (correct) output. This technique has previously been extended to the training phase by only correcting the network on those nodes with an error of more than some specified amount. A variation of this technique was used in the current experimentation. Consider the case above where an output of 0.9 (0.1) might be considered close enough to count as 1.0 (0.0). In a case like this the output itself can be processed (say with a threshold filter) so that an *actual* output of 1.0 (or 0.0) is achieved. The technique used in the experiments described consisted of filtering the output of the network so that it matches some valid target output. This was done by reference to a list of all legitimate outputs (all possible classifications), and by considering the output to be that classification to which it was nearest (in Euclidean vector distance – see section 5.4.5.1 for a fuller description of this technique). Thus the network was "forced" to make a decision regardless of the actual values of the outputs. An error measure based on these forced outputs was calculated over the training set giving "percent correct classifications" measure for the network. This figure, along with the actual Root Mean Square (RMS) error of the output nodes, was used to determine when to halt the training. An RMS error of less than 1% *and* a percent correct classification figure of 100%, both measured over the training set, was considered sufficient to halt the training sequence.

The third problem, that of choosing a set of data over which to calculate the network performance, can be resolved by simply making a pragmatic choice. The type of data being used will determine its availability. The considerations in choosing a set of data upon which to test the network *whilst training it* are the same as for choosing a test set for a trained network (as discussed in section 5.3.1). Theoretically, testing the network over the whole available database is the easiest way to see if it has reached the required level of performance..In practice however it is desirable to be able to determine whether a network that is failing is doing so because it is unable to generalise from the taught data, or because it is unable to learn anything at all. It is common practise to test a network on the training data during the training phase, and to investigate its properties of generalisation once the training data is known to have been successfully learnt.

5.3.3.2 Change in network weights

Another approach to determining network maturity is to regard a network as mature when it has settled into a stable state, or in other words is not learning any more. Given that learning in a network is achieved through weight change, it would seem that observing the weight dynamics would allow the maturity of a network to assessed. However, following the equations specified in section 5.2.1, it can be seen that whilst the total amount of adaptation in the weights of a network does depend on the total external error of a network on the training set (in that a zero error would lead to no further adaptation of the weights in the vanilla back-propagation algorithm), this relationship is very complex, depending on hidden activations, existing weights and possibly momentum terms. It can be said that *in general* the dynamics of the weight vector give little indication of the network's external behaviour.

This is illustrated in Figure 13, Figure 13a and 13b showing the amount of change in the weights of a network against an RMS error curve, for a 4 bit and 5 bit encoding problem respectively, and Figure 13c and 13d showing the same for the more complex problem of plankton classification. The biggest problem in using such a measure of network maturity is in detecting a truly 'stable' state. Initially in training randomly structured networks, weight changes are often very small, and the presence of wild fluctuations often leads to intermediate conditions of low weight change, neither of which states will be desired as a stopping condition for training. It would seem that if one is intending to use the weight changes as a reflection of the error of a network, then it is as well to use the error directly as an index of network performance.

5.3.3.3 Number of training epochs

A third approach to determining network maturity is to train a network for a certain number of presentations of the training set — a certain number of epochs. The advantage of this technique is that it is computationally simple — a record of the number of training epochs performed is all that is required. This measure is independent of size of network,

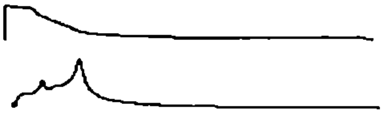


Figure 13a

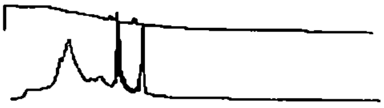


Figure 13b

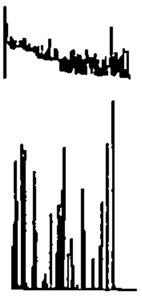


Figure 13c

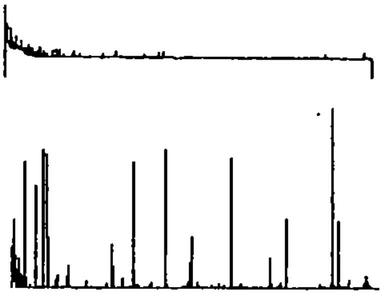


Figure 13d

Figure 14: Each figure consists of an RMS curve above, and the weight dynamics below. a) and b) are for a binary encoding problem, (encoding a serial representation of a 2^n bit number on n bits), size 4 and 5 bits respectively. Figures c and d are for a plankton classification problem. c) has halted with an error of less than 1%, d) has continued for 1000 epochs.

complexity of network or number of training patterns. The disadvantage of this technique is that it is blind to performance of the network. A network may have failed to learn any input–output associations at all after a considerable number of epochs, indeed, the network may be trapped in a local minimum and be unable to learn any further no matter how many epochs of training are performed. Thus number of training epochs is no guarantee of reliable performance.

However, if a network does become stuck in some local minimum then it is desirable to halt the training of the network and perhaps try again with some different initial starting conditions. For this reason, it is profitable to keep a record of number of training epochs so that some limit can be put on the amount of training a network is to receive. In the current set of experiments, a combination of error measures (RMS and percent correct, as described in section 5.3.3.1) and limitation on training time was used. Training was halted when a network satisfied both error measures, *or* had been trained for some set number of epochs (typically 1000). Both those networks that had reached the specified level of performance and those that had had their training terminated were used in measuring the ability of a system to learn a set of data.

5.4 TESTING THE NETWORK

5.4.1 Multiple experimentation

The purpose of the experiments conducted was primarily to investigate the potential of a neural network system to classify unseen or novel plankton forms. A number of different factors such as network structure, network size, pre-processing of the data and various details of the training algorithm were to be investigated. A simple approach might be to train a network once on each different parametric variation, and then measure the network's performances and so determine which technique works the best. Unfortunately this technique would not guarantee to give the best network for the task. There are a number of factors in this type of experimentation that preclude the possibility of accurately assessing a network's performance by one trial alone. These include network factors such as the random initial state of the weights in the system, the random selection of training and test sets as described in section 3.2.1, possibly random presentation sequences of data, possible inclusion of noise in the system and so on. The effect of these random elements in training conditions on the performance of some network under those conditions can be minimised in the assessment of network behaviour by performing a large number of trials. It is hoped that this technique will allow a more true measure of performance to be obtained, and that random fluctuations will be evened out over a large number of trials. Each trial would involve a different selection of the random variables such as initial weights, selection of training data, presentation sequence and so on, and a number of trials would be performed.

5.4.2 Correlation

As is discussed in section 5.3.1, it is desirable to have as large a data set as possible for both training and testing the network. Given a limited source of suitable data it is immediately apparent that these two aims are at odds with one another, and that some type of compromise has to be reached. A large training set is desirable only for the purposes of improving the networks resultant behaviour given the problem. If this problem is to classify just a limited set of patterns and no others then the whole set of possible inputs can be included in the training set, and the network exhaustively trained until a satisfactory performance level is reached. If, however, the class of patterns to be classified is a *world* population, and has many, possibly limitless members, then only a subset of these can be used for training purposes (as is the case in the classification of speech, sonar returns, marine organisms, vehicles and so on. These are *real world* problems). The size of subset required to train a network will depend on the distribution of the classes of the population in state space. It *is* possible to have a simple case where a vast population might consist of two discrete classes, well separated in state space — discriminating a fog-horn from a sneeze might require training on a very small subset of the total population of sneezy/fog-horn sounds (possibly only two members) — but in more complex real world problems it is likely that a larger than minimal subset be required to enable a network to sufficiently encode the required discriminations. The amount of training data to be used therefore is to be determined empirically, and by performing multiple trials with different amounts of training data it is possible to determine a suitable amount of training data for a specific type of problem on a specific type of network, and to see if the amount of training data is having an effect upon the performance of the network. This is done by looking for evidence of a correlation between the size of training set and the network's ability.

The test then of a network's ability is its performance on the test data. It is desirable for the test data to be as representative of the world population as possible, so that a result upon the test data is indicative of a (theoretical) result on the world data. As discussed in section 5.3.1 the simplest way to obtain reasonably representative data is to use as much as possible drawn at random from the population. Since in each trial the training data had been selected at random, the remaining data had also been selected randomly from the source. Using all the remaining data (i.e. that *not* in the training set) was the obvious approach to maximising the test set (and so was practised). In order to assess the response of the system to an increasing size of training set, it is desirable to look for a *trend*, in this case a trend for the error as measured over the test data to decrease as the size of training set increases (an improvement in performance with an increase in amount of training data). Thus a correlation is to be looked for between these two variables. The correlative measure used was the *product-moment correlation*, also known as *Pearson's rho* (usually ρ is used to represent the population value, i.e. that one wishes to know, and r used to represent the sample value). The formula below expresses this.

Product-moment correlation;

$$r_{XY} = \frac{\sum xy}{nS_X S_Y} \qquad \text{(eq. 5-11)}$$

where: $x = (X - \overline{X})$, $y = (Y - \overline{Y})$, $\sum xy$ is the sum of the products of the paired deviation scores, n = the number of *pairs* of scores, $x = (X - \overline{X})$, $y = (Y - \overline{Y})$, $\sum xy$ is the sum of the products of the paired deviation scores, n is the number of *pairs* of scores, and S_X, S_Y are the standard deviations of the two distributions.

An example of such a correlation would be between the height of a person and their headroom below some particular beam. For any given experiment, a pair of scores for each of the numerous trials would be recorded, one score being the size of training set, the other being the error over the unseen data. A value for the correlation would be calculated as in equation 5-11.

The significance of the correlation is dependent on the size of the set of pairs of scores. Since the correlation measure is in effect a measure of how deviant the data is from the regression line, at least three pairs of scores are required to calculate it (any two points always lie on some straight line). The measure of *degrees of freedom* (*df*) is therefore n–2. To calculate the significance of a certain correlation over a certain number of pairs of scores a level of probability table is generally used (for instance Fisher and Yates, 1963). These give the critical values of r at various levels of probability for a range of degrees of freedom. What level of significance is to be used as the benchmark (i.e. when to accept or reject the hypothesis of a trend) is the choice of the researcher. In the experiments described here, a level of 5% probability (0.05) was usually taken as the level of significance required to indicate a trend, except where other values are shown. That would mean that there is only a 5% chance that the results are due to chance, and that it is 95% likely that the trend is actually due to a genuine association between the variables (all other assumptions being true). For a particular experiment, the value of r (let us say 0.423) over n pairs of scores (say, 47) would be looked at against the table of critical probabilities. In this case it would be found that 0.423 exceeded the critical value of 0.3721 (for a two-tailed test) for a probability of 1%, and so the result would be deemed significant and recorded thus;

This coefficient takes a value between -1.0 and +1.0. A positive score indicates a positive correlation, that is, there is a tendency for high (or low) values of one variable to be associated with high (or low) values of the other variable. A negative score indicates the opposite. Roughly speaking, the degree of association between two variables can be regarded as the square of the correlation. A correlation of +1.0, or -1.0, implies that there is a perfect correlation between the two score, and that given one score the other can be (perfectly) predicted. In such a case a plot of the two variables would fall upon a straight line.

Experiment 4; $r = -0.423$, $df = 45$, $p < 0.01$ (*two-tailed test*)

It is also possible to calculate the confidence intervals associated with any measure of r given the number of variable pairings. This interval indicates the range that world population correlation might fall within, for example, using the above figures;

$$r = -0.423,\ n = 47,\ C(-0.463 < \rho < -0.381) = 0.95$$

This is read as stating that the value of the correlation between the two variable used is 95% likely to fall between the values given. Of course, correlations between other variables can be looked for, for instance, the error over the training set against the size of the training set might be used to see if the network was having more difficulty in learning as the amount of training data increased.

5.4.3 Extrapolation

Another advantage of performing a large number of trials is that the different sizes of training set can be used in order to attempt to assess the ability of a network given a greater availability of training data than is actually the case. In the plankton experiments described below, there were only a limited number of labelled samples available. Given that a portion of these were to be used in testing the network, it was considered that a test on the few available samples (where a relatively large training set was used) might not be fully indicative of the ability of a network (or any other system) in this domain. A test on a much larger data set was considered desirable for experimental reasons, but would require a much diminished training set (possibly leading to a poorly trained net) or the acquisition of more source data (not an immediately practicable proposition). In order to test the potential of a system, an extrapolation was made from actual performance on the available data to possible performance given more training data. It would be expected that in a domain of this complexity, a trivially small training set would result in a poor performance of the network when tested on novel data. Given one example of each of two classes of input, it was expected that a network would be unable to generalise from such data to novel inputs, the training set being insufficiently representative of the population. It was also expected that this performance would improve with an increase in the size of the training set, as it became more representative. These hypotheses, along with the reasons cited in above (section

5.4.1), led to each test of a network on a problem consisting of a number of trials (usually 100), with a randomly selected size of training set for each trial.

The correlation between the size of training set, and performance score, was calculated as in equation 5–11, (as described in section 5.4.2). Where a significant trend was indicated, it was possible to attempt to extrapolate from the given data to hypothetical cases with a larger training set. The aim of this technique was to determine whether or not the performance of the network might be expected to improve above and beyond that already achieved with the currently available training set, given a larger training set.

5.4.4 Regression analysis

Recall that the correlation coefficient can be regarded as giving a measure of fit to a straight line. The value of Pearson's rho gives one an indication of the predictive possibilities of the data, a non–zero correlation implying that knowledge of the value of one variable gives us some knowledge of the other, but does not enable one to make the prediction. In order to do this, a *best fit* line is calculated for the data. For most purposes, a best fit is regarded as that which minimises the *squares* of the deviations from the line, thus the method is called a *least squares* method. For predicting the value of variable *y* given *x*, a regression line of *Y* on *X* is calculated. The equation for this line is

$$\textit{Regression of Y on X;} \qquad Y' = \left(\frac{S_y}{S_x}\right) X - \left(r\frac{S_Y}{S_X}\right) \bar{X} + \bar{Y} \qquad \text{(eq. 5–12)}$$

where: Y' is the predicted score in Y, S_X and S_Y are the two standard deviations, $\bar{X}$ and $\bar{Y}$ are the two means, and r is the correlation coefficient between X and Y.

For any set of two variables there is also another regression line of *X on Y*. This line would be used for predicting the value of variable *x* given *y*. Which regression line is used depends on the direction of the prediction required. However, as mentioned in section 5.4.2, the use of such regression lines implies a presupposition of an underlying linearity to the correlation measured. Attempting to fit a line to data that does not follow such a trend is of little use. In the example of extrapolating from the performance of a network with one size of training set to that with a greater training set, it is apparent that any trend that might exist cannot be linear. This is because there is are upper and lower bounds to the performance. 100 percent and 0 percent are the limits of its ability. It is meaningless to state any performance outside these bounds. Hence any improving performance of a network can only approach and reach 100%. (One would have reservations about claiming that level of performance in a real world situation — in effect claiming infallibility — although it would be possible where there were not limitless inputs.) Hence any trend of improvement in performance must be non-linear, approaching (perhaps asymptotically) its limit. A possible model for such behaviour is a logarithmic curve, tending towards, but never reaching the limit. By converting the performance measure (or error) to a logarithmic scale, it is possible to apply regression analysis to the data, in order to predict the likely size of training set required for a given performance, or the likely performance given a training set of some size.

5.4.5 Probability measures of performance

A significant proportion of papers in the field of Neural Networks, whether presented at conferences or published in the journals, report the results of experimentation with particular data. Experiments into the effects of different training algorithms, architectures and data sets are often reported with reference to the 'success' (or not, as the case may be) of the network in achieving a particular goal. Occasionally single results are reported, and the network is shown, for one particular instance, to have achieved some particular goal state. More often a number of results are reported, and a value for network performance derived from either 1) averaging the results of a number of separate trials, or 2) the number of correct responses of a particular network to a particular data set. A simple report of some percentage 'score' to represent the performance of a network on some task is inadequate, since it does not necessarily represent the true ability of a network. This being the case, a more significant measure of network performance is needed, which would give a truer assessment of a network in a domain, and thus be more suited for making comparisons between systems.

5.4.5.1 Generalisation

Typically a network is initialised with a given architecture, and a randomly selected set of weights. A training regime is then applied, using the training data (typically consisting of a set of input-output pairs), and the network (one hopes) adapts in some way to accommodate this data so that when given some input from the training set, a suitable output is generated. Very often a measure of the success of a training regime is used as part of the learning algorithm itself, and training can be stopped when the network is giving satisfactory performance (as determined by this measure, see section 5.3.3.1). A network can then be tested on inputs that are not part of the training set (the test set). If the particular input–output mapping 'learnt' by the network in response to the training data results in appropriate responses to the new, novel data, the network is said to be able to generalise. This ability can be given some measure that indicates the success of the network at generalisation.

One commonly used measure is the *Root Mean Squared* (RMS) distance between the actual outputs of a network given some input, and the target outputs, that is what the network is desired to give in response to that input. While this measure does give an accurate account of the difference between the actual and desired response of the network, it does not necessarily give a good picture of the network's behaviour. Consider a network that is required to classify (categorise) some data. In such a case the output of the network might represent various particular categories, say by having one node per distinct category, or some distributed representation of the category membership. So for a particular set of inputs we obtain the RMS measure of fit between the actual categories and those generated by the network. Say for instance that the network is required to distinguish between those patterns that are to be classified as [0 1] on the output layer, and those that are class [1 0]. If we have an RMS error of 0.2, what does that tell us? We cannot from that alone distinguish between a network that is correctly classifying 80% of the data, and incorrectly classifying the remaining 20%, and a network that is giving outputs that are within 0.2 of the desired output (hence giving the two classes as [0.2 0.8] and [0.8 0.2]). The latter case looks very much like a network that is moving to the right solution, and correctly separating the inputs. In such a case we might wish to train the network a while longer, or we might settle for the network as it is and recognise the actual outputs as indicating the required categories. This can be done by some post–processing of the output layer, as mentioned in section 5.3.3.1. The activation functions used in many networks actually prohibit the generation of an exact [0 1] output, and so, often outputs within some degree of 'closeness' to the required targets are treated as correct. A more rigourous formulation of this idea is to adopt a *Nearest*

Neighbour (NN) approach to the outputs. This implies that an output pattern is treated as that target pattern to which is nearest, by some measure. A simple Euclidean metric then, gives us

$$d_{j_k} = \sqrt{\sum_{i=0}^{n}(x_i^j - x_i^k)^2} \ . \qquad \text{(eq. 5-13)}$$

as the distance between patterns j and k, the actual and the desired output. A particular output is then taken to stand for that target to which it is closest (i.e. in the previous example [0.2 0.8] would be taken to mean [0 1], (as opposed to meaning [1 0]), as would [0.4 0.5]). By interpreting the output in this way we force the network to imply one of the possible categories. Comparing these new *implied* outputs with the desired outputs tells us how many patterns the network is classifying correctly. Then a 'score' of 0.2 tells us that 20% of our inputs are not invoking the correct response. This technique has frequently been adopted in reporting network performance (e.g. Gorman & Sejnowski, 1988; Sietsma & Dow, 1991), and is used in the current experimentation where indicated.

5.4.5.2 Quality of generalisation

Lendaris (1990) points out the need for some "quality of generalisation" metric to enable comparisons to be made between different network performances. A simplified account of his metric will suffice to show that it gives an inadequate measure of how well a network actually performs. Let t be the size of the training set, and g be the size of the test set (the 'generalise set'). Similarly, let t_c and g_c be the number of *correct* responses to the training and test sets respectively. Lendaris initially proposes the expression

$$\frac{g_c}{g} \ \textit{generalisation via} \ \frac{t_c}{t} \ \textit{training performance on} \ \frac{t}{t+g} \textit{exposure} \qquad \text{(eq. 5-14)}$$

to represent the performance of a network. There is a problem with this, in that there is no indication of how two such expressions might be compared (e.g. how does 0.8 generalisation via 0.8 training performance on 0.8 exposure compare with 0.8 generalisation via 0.9 training performance on 0.2 exposure? Which is better, and why?). A more qualitative metric is proposed as

Generalisation Ratio; $$GR = \frac{g_c \ /g}{t_c \ /t} \qquad \text{(eq. 3-1)}$$

(which might be written as $g\% \ / \ t\%$), a ratio of the test performance to the training performance. This enables a direct comparison to be made between different systems. To quote the example given in Lendaris (op. cit.), a network that has 80% generalisation via 80% training performance gives $GR = 1.0$, compared to a network that has 80% generalisation via 100% training performance, $GR = 0.8$, thus "yielding numbers that correspond with the intuition that (the former network) *did better* than (the latter network)" (Lendaris 1990, p.711, my italics).

In what way is the first network performing better than the second? Granted, there is some degradation in the latter case between the training and test performance, and it might be suggested that in the former case a better internal representation/mapping/feature set is found and fully exploited on the test set. However, such a representation or mapping might have been found in the latter case, and in addition to that some further fine tuning been done. Had the two networks obtained their performances on the same sets of data (training and test), then the former network would have actually classified *less* items correctly than had the latter.

A second and final objection to the Generalisation Ratio as a performance metric is this. An untrained network will, on average, perform as well on the test data as the training set (having been exposed to neither). To return to the two-way classifier mentioned above, (with [0,1] and [1,0] as outputs), such a network would give an RMS error of around 0.5 on data *before any training*, whether it be (proposed) training or test data. (The NN error would be about the same). Attempting to use the above metric gives us $GR \approx 1.0$, supposedly a better performance than a network giving 90% generalisation via 100% training performance.

5.4.5.3 Performance probabilities

It seems that $g_c \ / \ g$ (or $g\%$) alone gives us a better measure of generalisation than it does in comparison with the performance of the network on the training set, and indeed it is often this figure that is reported in the literature. However, we need to take account of the size of the test set when considering these data. Occasionally one sees a one-off performance of '100%' reported over one trial. How should this figure compare with a performance of (say) 90% over 50 trials? Unfavourably, one should hope. What is needed is a measure of how *significant* a reported performance is. If we find that some performance can be explained as a chance occurrence, then we would not wish to impute any skill to that system. A coin, for instance, might get a 50% success rate on a binary classification task. The more *unlikely* the systems performance (in terms of success or failure) is over a set of data, then the more probable it is that such an outcome is due to some effect other than chance occurrence. The probability of a system's response gives us a measure of the efficacy of that system.

Such a measure is to be found in binomial theory. A series of trials of a network, i.e. testing the network on N different inputs, can be treated as a series of *Bernoulli* trials. These are trials in which the probabilities of various outcomes are unchanged from one trial to another (such as tossing a coin). In our example, the outcomes are success (a correct response) or failure (an incorrect response). A *binomial* experiment consists of a fixed number of Bernoulli trials, independent of each other, where there are two possible outcomes. We can treat the test set of data as the basis of such an experiment. Now the probability for a particular error rate can be calculated. The binomial distribution theorem says that if an experiment consists of n trials, each with a probability p of success (and hence probability $q = 1-p$ of failure), then the probability that the experiment results in exactly x successes (and hence $n-x$ failures) is

$$b(x;n.p) = \binom{n}{p} p^x q^{n-x} \quad \text{where} \quad \binom{n}{p} = \frac{n!}{r!(n-r)!} \ . \qquad \text{(eq. 5-15)}$$

In our example of a two-way classifier, $p = 0.5$ and $q = 0.5$, since the network has provisionally a 50% chance of giving the correct response to a given input pattern. If the network has learnt anything, we might reasonably expect performance to exceed 50%. To be convinced of the ability of a system to discriminate, we might set a level of *un*probability that we deem to be significant. That is to say that if we observe performance beyond this level of probability, we will regard it as an artifact of learning rather than chance. It must be borne in mind that rather than being interested in the probability of a particular result (say 70% performance) occurring, we are interested in the probability of *some result that unlikely occurring*. For instance, say we toss a coin 100 times. The chances of, say, *exactly* 54 heads coming up is less than 5%. Rather than take this probability (and hence declare the coin to be biased due to so unlikely a result occurring) we should look at the chance of anything (at least) that unlikely occurring. This will tell us the chance of getting such an extreme score. To calculate this figure we need to sum the probabilities of 54 heads, 55 heads, and so on to 100 heads. In general, the equation for this cumulative probability is

$$\textit{Cumulative probability;} \qquad \sum_{x=r}^{n} b(x; n, p) \quad \textit{where} \quad x > \frac{n}{2}\,. \qquad \text{(eq. 5-16)}$$

(In the case of $x < n/2$, $n-x$ should be used instead of x.) This formula can get quite complicated to calculate for high values of x, but as we shall see this figure is of more significance to us at lower values of x. Extensive tables exist for both the binomial distribution (equation 5-15), and the cumulative probabilities (equation 5-16) (Romig 1953, Harvard Comp. Lab. 1955).

5.4.5.4 A proposed performance metric

The calculation of the cumulative probability of a system performance enables us to judge the worth of the reported performance of g_c correct responses to g items, with the a priori probability of a correct response p. It is not of importance whether this represents a training or a test set of data. It is suggested that this probability be used to judge the worth of an obtained performance of $g\%$, to give a probability performance metric P of

$$\textit{Probability performance metric;} \quad P \;=\; \frac{g_c}{g}\left(1 - \sum_{x=g_c}^{g} b(x; g, p)\right) \cdot \text{(eq. 5-17)}$$

P can be calculated for a network performance on any data, whether training or test data. Figure 15 illustrates the effect of the second term in equation 6 (the cumulative probability term) on the usually reported $g\%$, for the binary classifier case. Curves are shown for values of P when $g\%$ is approximately 60, 70, 80, 90 and 100%. It can be seen that the greatest effect is had on $g\%$ when the number of trials is low, and/or when the reported performance is low. The proposed metric has very little effect on the score when the either $g\%$ or the number of trials is high. This might encourage one to run a greater number of trials, especially in those cases where the results are modest. By incorporating the effect of the size of trial and the *a priori* probability of success, the probability performance metric enables a quantitative measure to to calculated, which can be used to directly compare the performances of different systems, even when obtained on quite different trials.

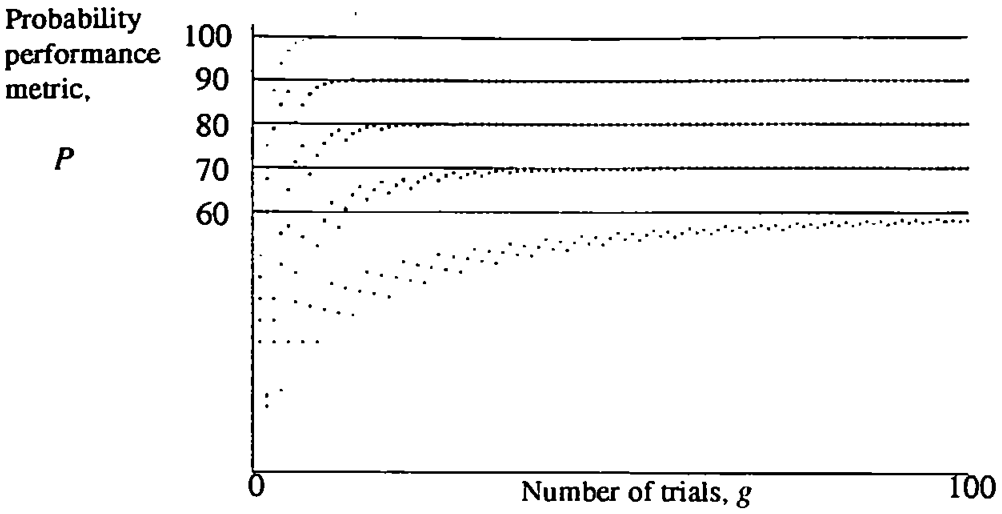


Figure 15: The probability performance metric vs. number of trials

5.5 SUMMARY

This chapter has sought to introduce the particular techniques and methodologies used in the experimentation described below. Of particular relevance is the back-propagation algorithm discussed in section 5.2.1. This powerful algorithm is introduced and discussed, along with some variants. Various experimental considerations regarding the selection of data for training the network are discussed in section 5.3.1, and operational decisions in training the network are discussed in the following part of section 5.3.

Analysis of trained networks is discussed in section 5.4, where the four heads of multiple experimentation, correlation, extrapolation and regression are considered, leading to section 5.4.5 where it is argued that there is a need for the results of neural network experiments to be represented in such a manner that comparisons can be made between experiments on different data sets and tasks. Current practice does not encourage such comparisons, and previous attempts at formulating a metric have failed to provide a measure that meaningfully represents the ability of a system in some domain. An expression for such an evaluation, based on *a priori* probability, is proposed.

Chapter 6. Initial Network Tests.

6.1 INTRODUCTION

This chapter introduces the initial back-propagation simulation experiments, carried out with a view to verifying that the actual implementation in use performs as it should. First a simple non-linear mapping problem is attempted, a classic problem demonstrating that characteristic of a multi-layered perceptron that enables it to solve problems that the single-layer perceptron cannot. Some relatively more complex problems are then attempted to investigate the network's performance. Finally, some further experiments are performed to investigate the effects of altering some parameters of the data to be learnt, in order to investigate what types of pre-processing might be beneficial to a network attempting to learn a classification.

6.2 IMPLEMENTING BACK-PROPAGATION

6.2.1 Verifying the simulation

The first task to be carried out in the series of trials reported here was to implement a back-propagation network in software, and to then verify that the implementation was working correctly. The equations for the algorithm for the network simulation were largely taken from Rumelhart *et al.* (1986b). The code was written in POP-11, a high level language in the Poplog environment (© University of Sussex), commonly used in Artificial Intelligence development applications (see Barrett *et al.* 1985). The complete code for the simulations is shown in Appendix II.

Use of the back-propagation simulation was through a basic user interface, to enable use of the code by a variety of users and for a variety of applications. A simple example will suffice to illustrate this process, and at the same time allow presentation of a simple test of the algorithm, namely the Exclusive-Or problem (variously referred to as the EOR, EXOR or sometimes XOR problem).

6.2.1.1 The Exclusive-Or function

INPUT A	INPUT B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

Table IV. The Exclusive-Or function.

The EOR function takes two binary inputs and produces one binary output. The truth table for this problem is shown in Table IV. As can be seen, EOR is true if and only if just one input is on (i.e. not both). One characteristic of this function is that it is non-linearly separable, and hence cannot be solved just by a summed consideration of the inputs. In network terms, this means that a single-layered network is unable to solve this problem, and that a hidden layer is required.

6.2.1.2 Simulating a network

The following sequence illustrates creating a network and using it to solve the EOR problem. Comments are in italics and are not interpreted by the machine.

```
load lib/backprop.p;          Loads the appropriate program file

net_set_up([2 2 1]);          Sets up a network structure with 2 input
                              nodes, 2 hidden nodes, and 1 output
                              node.

connect_all(1,2);             Connects every node in layer 1 (input) to
                              every node in layer 2 (hidden).

connect_all(2,3);             Connects every node in layer 2 (hidden) to
                              every node in layer 3 (output).

newarray([1 4]) -> input;     Set up array to store the training patterns...
newarray([1 4]) -> target;    and similarly the training outputs (classes)

[0 0] -> input(1); [0] -> target(1);Specify the particular problem
[0 1] -> input(2); [1] -> target(2);in terms of the input-output
[1 0] -> input(3); [1] -> target(3);pairings.
[1 1] -> input(4); [0] -> target(4);
```

```
0.5 -> learn;               Set the learning rate (η),
0.9 -> momentum;            and the momentum (β) as in equation 5–8.
newarray([0 500]) -> error;    Build array for recording errors as
                               trials continue.
RMS() -> error(0);          First recorded error (prior to training).
for z from 1 to 500 do      For 500 epochs (500 presentations
                            of the training set),
for y from 1 to 4 do        for every pattern in the training set,
input(y) -> Input;          present input to input layer,
target(y) -> Target;        present desired (teaching) output.
propagate();                Allow the activations to propagate
                            forward throughout the network,
correct();                  and back–propagate the errors, correcting
                            the weights.
RMS() -> error(z);          Store the latest network error.
endfor;                 Repeat cycle for next pattern in epoch.
endfor;                 Repeat cycle for next epoch.
```

This piece of code produces a running tally of the network's RMS error on the EOR problem in the array *error*. This can be graphically illustrated, as is done in Figure 16.

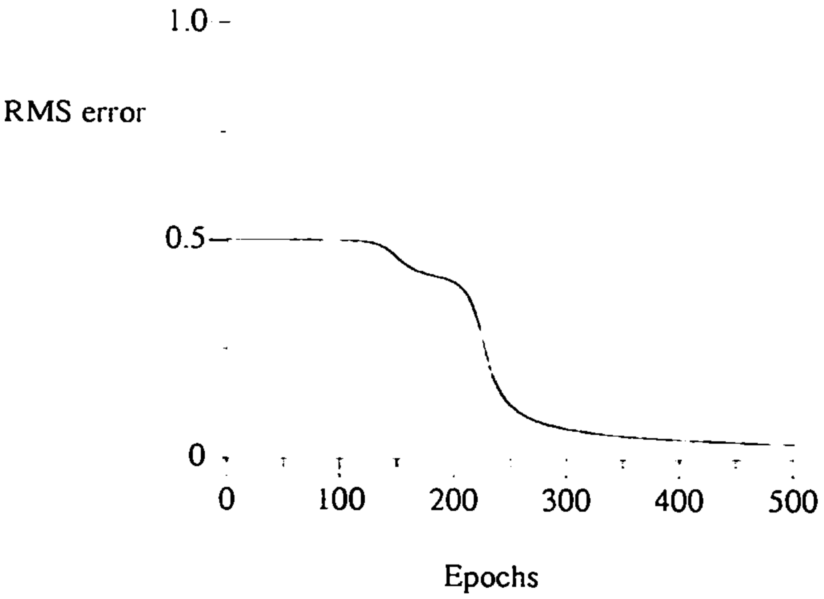


Figure 16: RMS error in a network on the EOR problem.

Figure 16, derived from a real simulation illustrates well a typical, if well-behaved, learning curve. It can be seen that initially the RMS error is around 0.5. This is because the weights in the network are initialised to small values (say $-0.1 \le w_{ij} \le +0.1$), effectively prohibiting the propagation of activation through the network. As a consequence of this, the activation received by the output node is very small, producing an output near to 0.5 (as according to equation 5-8) regardless of the input values. Since the two target values are 0 and 1, the error of an output of 0.5 in both cases is 0.5. This error does not change much over the first few training epochs. Once change is under way, however, it is rapid, until the error rate settles down as it approaches zero. The speed of learning is comparable with that quoted in Rumelhart *et al.* (1986b), see pp. 331 and 333. Figure 17 plots the amount of weight change in the network at each epoch. Comparison of Figure 16 with Figure 17

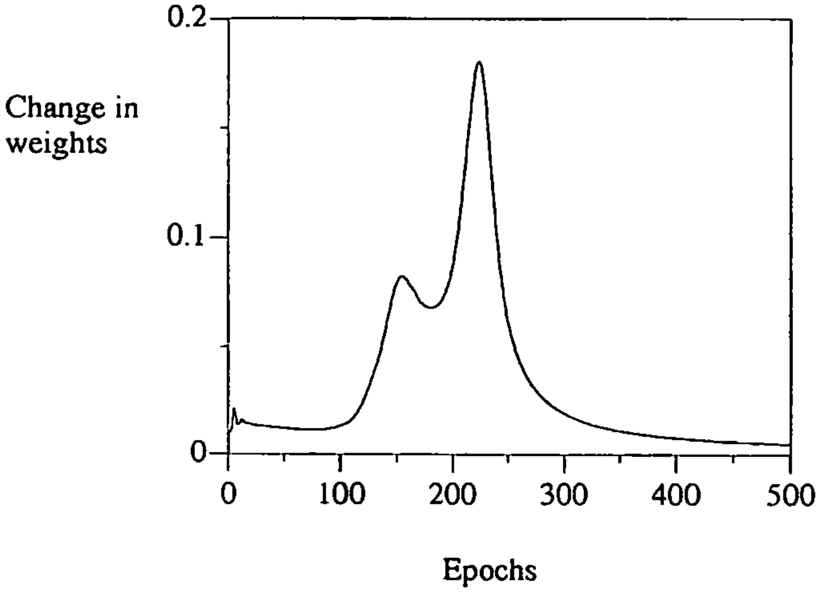


Figure 17: Weight change in a network on the EOR problem.

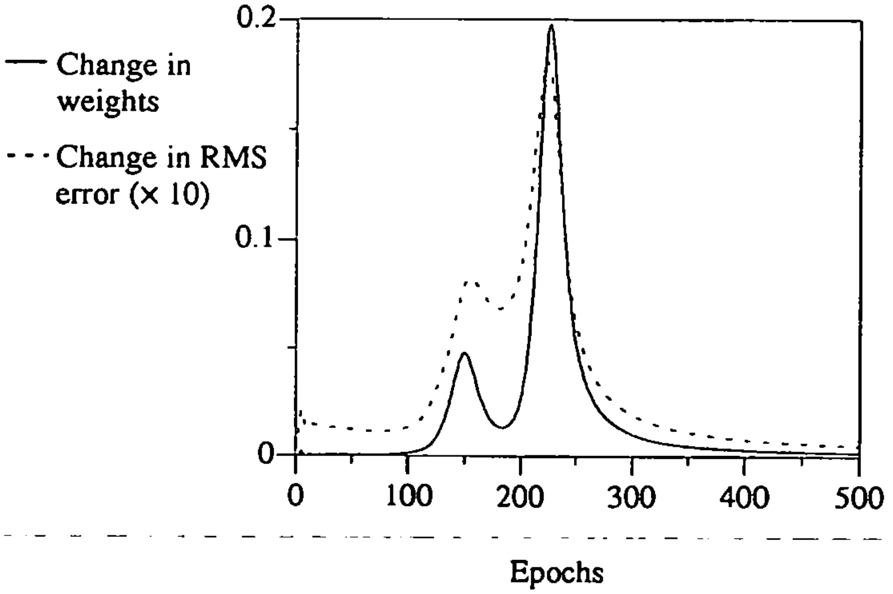


Figure 18: Weight and RMS error change in a network on the EOR problem.

shows that as might be expected, the largest changes in the error of the network occur at the time of the largest changes to the networks weights. This can be made more explicit, as is done in Figure 18, where the change in weights of the network (as plotted before) is shown along with the change in error, calculated as error(t-1) - error(t+1), a rough gradient function. This graph illustrates the very high correlation between the amount of adaptive activity and the reduction in the error measure.

6.2.1.3 Data compression

The task of data compression was used as a more complex test of the back-propagation simulation. This problem takes an input of 2^n binary bits, feeding through a bottle-neck of n hidden units to an output of 2^n units, the output pattern being identical with the input pattern (see Figure 19). The inputs consist of one binary bit being on, being a place representation of a number from 1 to 2^n say. The required output is exactly this input. For the hidden units to represent these 2^n possible inputs on only n nodes some concise representation must be found (for instance, a binary representation would suffice).

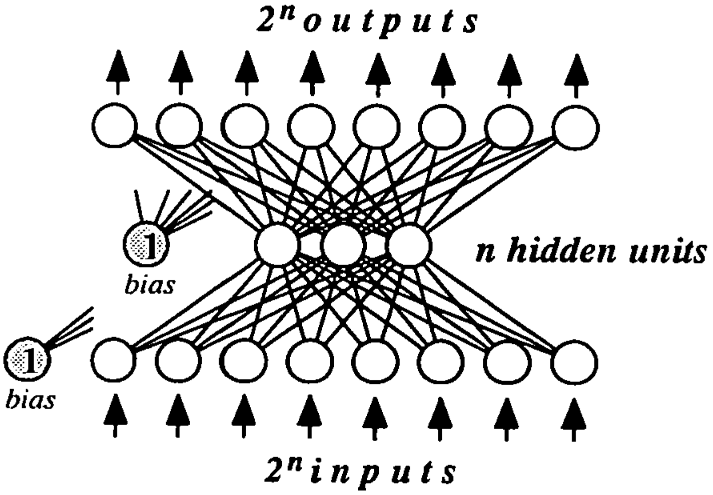


Figure 19: A feed-forward data compression network, with $n = 3$.

The algorithm was tested on trials with $n = 1, 2 \dots 5$, a progressively more difficult problem. The results are shown in Figure 20. In every case the algorithm learns the problem fairly rapidly, to a low level of error. One observation is that the larger n is, the more rapid is the initial descent from an error of 0.5. This is because of the increased size of training data, such that one pass through the data (one epoch) gives a greater number of opportunities for the weights to adapt to the problem. We can conclude from the (not exhaustive) results presented above, that the back-propagation simulation behaves appropriately.

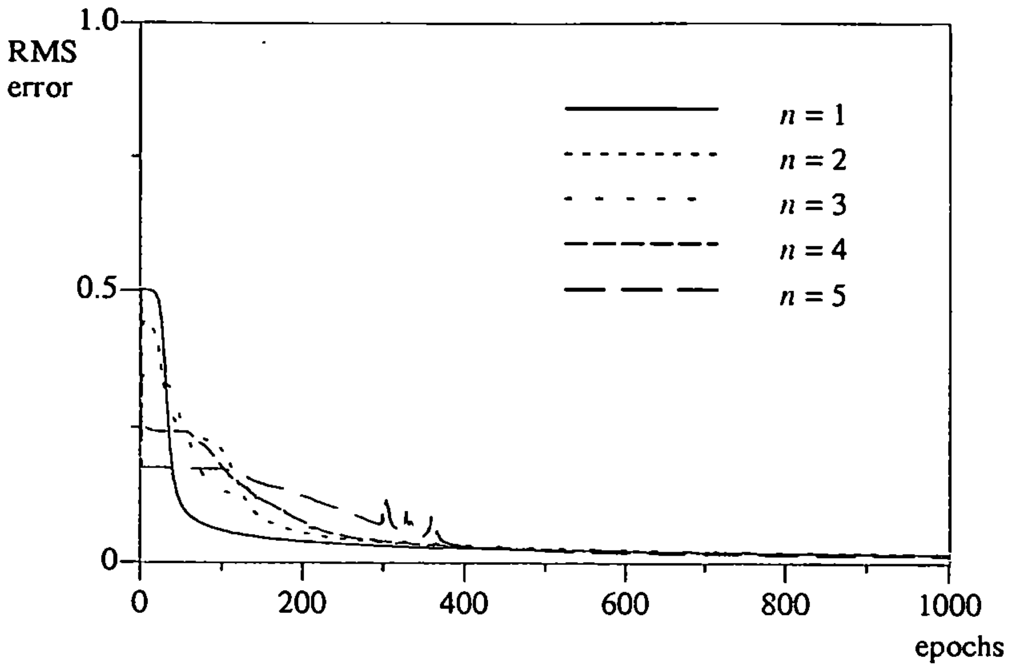


Figure 20: Learning data-compression, for n = 1, 2, ... 5.

6.3 PARAMETER EXPERIMENTS

While it is possible for a back-propagation network to learn to classify training data correctly, this does not necessarily happen in every case (see for example Rumelhart *et al.* 1986b). This might be because a particular random initialisation of the network weights allowed the network to fall into a critical local minima (*ibid.* p.331). This will quickly become apparent if multiple learning trials are carried out with differing initialisation conditions. The problem itself might be unsolvable given a particular network architecture, although the use of multi-layered networks minimises the likelihood of this. Yet another reason for a failure to learn might be that that a larger training set is needed for the network to develop the appropriate internal representations. Increasing the size of the training set however has an effect on the network's learning performance. Similarly, attempts to pre-process the data in certain ways affects the speed of learning of the network. Thus it was felt that an examination of the effects of varying certain parameters of a network's training set would be useful in determining what type of data set a network is best suited to.

Initially four parameters were selected to investigate. These were;

i) The number of patterns in the training set.

ii) The size of the patterns in the training set.

iii) The size of each item in each pattern.

iv) The noise/signal ratio of the data.

The task in these experiments was for the network to learn to classify random input vectors in accordance with a teaching input. The input data consisted of *n* vectors of length *l*, each to be given an arbitrary binary classification (i.e. a 0 or a 1 as output), for example, a vector where l=5 might be [0 0 1 0 1], requiring an output of [1]. The network in the following experiments was of [10 3 1] construction (10 input nodes, 3 hidden nodes and an output node) except where stated otherwise. The learning rate (η) was initially set to 0.5, and the momentum(β) set to 0.9. In general, 8 input vectors (n=8) were used. Each vector consisted of 10 random numbers (l=10) in the range of 0–10, and these were then normalised to sum to unity. For each experiment the RMS error of the network was taken after each complete presentation of the training set (one epoch).

6.3.1 Training set size experiment

The network was tested though training sets of *n* vectors, where $n = 2\times i$, i = 1, 2, ..., 10. 'Speed' of learning was observed to increase slightly as the number of patterns increased from 2 to 4 and 6 and 8. 'Speed' is in scare quotes for the reason that as the number of patterns increases each epoch consists of more learning sequences. The error measure here takes no account of that, and so the true speed of learning is not apparent. What is apparent is that the network has increasing difficulty learning the training set as it gets larger. At 12 patterns and above, the error measure starts to oscillate, although it appears to come out of this oscillation after some time. When 20 patterns were presented for learning the network showed a 'phase change', by which is meant that the error 'flipped' to a level and stayed there for some time. Again this was a temporary state, and learning eventually took place (Figure 21a).

Reducing η to 0.2 from 0.5 had the effect of enabling smooth learning with larger training sets than previously (Figure 21b). Oscillation and 'phase changes' occurred, but both of these effects were less prominent than before, and seemed to occur on larger training sets

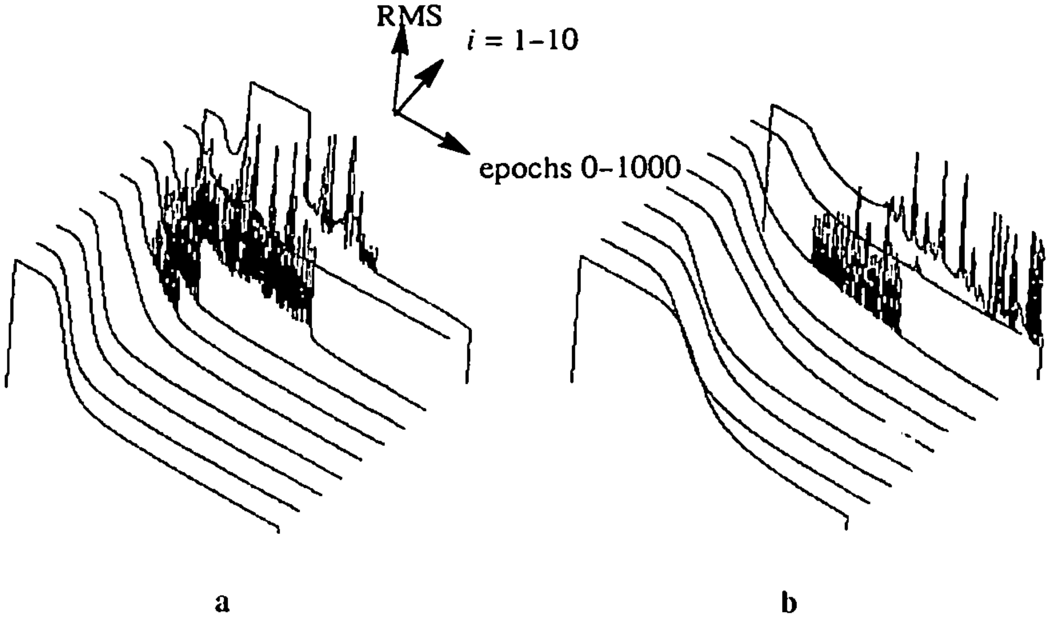


Figure 21: Trials with increasing size of training set: a) $\eta = 0.5$, b) $\eta = 0.2$.

than before. In these experiments eight patterns seemed to produce the optimum learning curve, and so this number was used as training set size in the following experiments.

In practical applications increasing the amount of data in the training set is not going to increase the required load on the network as much as is the case here. Each additional pattern learnt in this series of experiments is a randomly constructed vector, bearing no correlation with prior vectors designated the same classification. In real-world problems, it is likely that additional training data will bear a relation to other training data, being in a particular class not by arbitrary designation, but in virtue of its particular characteristics.

6.3.2 Training pattern size experiment

In these experiments the number of input nodes in the network (I) was increased from 1 to 10. This had the effect of determining the size of the input vectors presented. The network had trouble learning the shortest vectors, and less trouble the longer they became (see Figure 22). The reason for this perhaps surprising result is that the less input nodes there are, the less dimensions there are for placing a dividing hyperplane. The higher the dimensionality of the input space the more freedom there is for the hidden nodes to develop effective classifying hyperplanes. Again, in real-world problems there is likely to be more

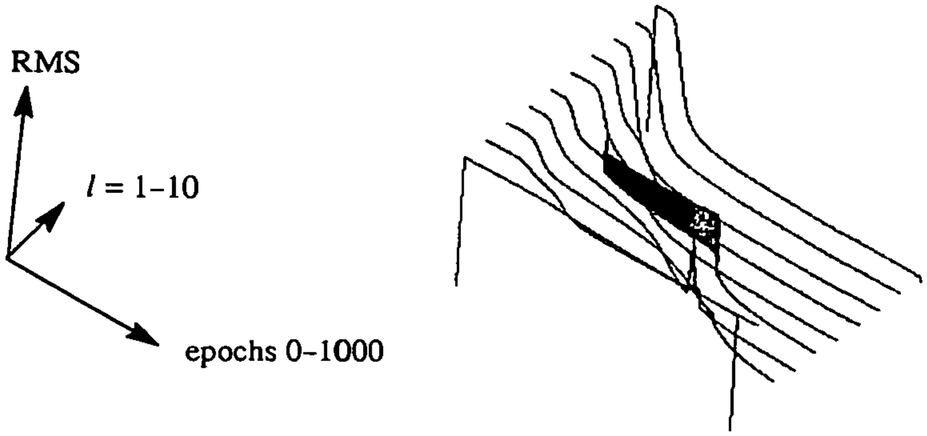


Figure 22: Trials with increasing length of input patterns.

correlation between input patterns of the same class than is the case with these randomly constructed inputs, and thus effective division of the input space is likely to be easier.

6.3.3 Input size experiment

In these experiments the numbers in the input vectors were multiplied by some factor f to see the effect of size of input signal on learning. The scaling factors were 10^f, where f = 1.0, 1.1, 1.2, 1.3, ..., 2.0. With η = 0.5 the network had no trouble learning the patterns until f=1.5. In this, and subsequent trials the network appeared to get stuck in a local minima, and not to make any performance gain at all (Figure 23a). With η = 0.2 this did not occur until $f \geq$ 1.8 was used to scale the inputs (Figure 23b). With η = 0.05 no problems were experienced with the given scalings (Figure 23c). The effect of a large signal being input to a network is to 'saturate' the nodes it is feeding to. A node receiving a large absolute amount of activation (positive or negative) from an input is activated at one of the flatter (and hence less responsive) ends of its activation function. Thus any change in the weighting of that connection will have a smaller effect on that nodes subsequent activation than if the node was being activated near zero, the steepest part of the activation function. One solution to this is to scale the inputs down, (for example by normalisation) or by using smaller initial weightings in the network. Both these measures were taken in the *Ceratium* trials, all inputs being normalised prior to training, and initial weightings of < ± 0.1 being used.

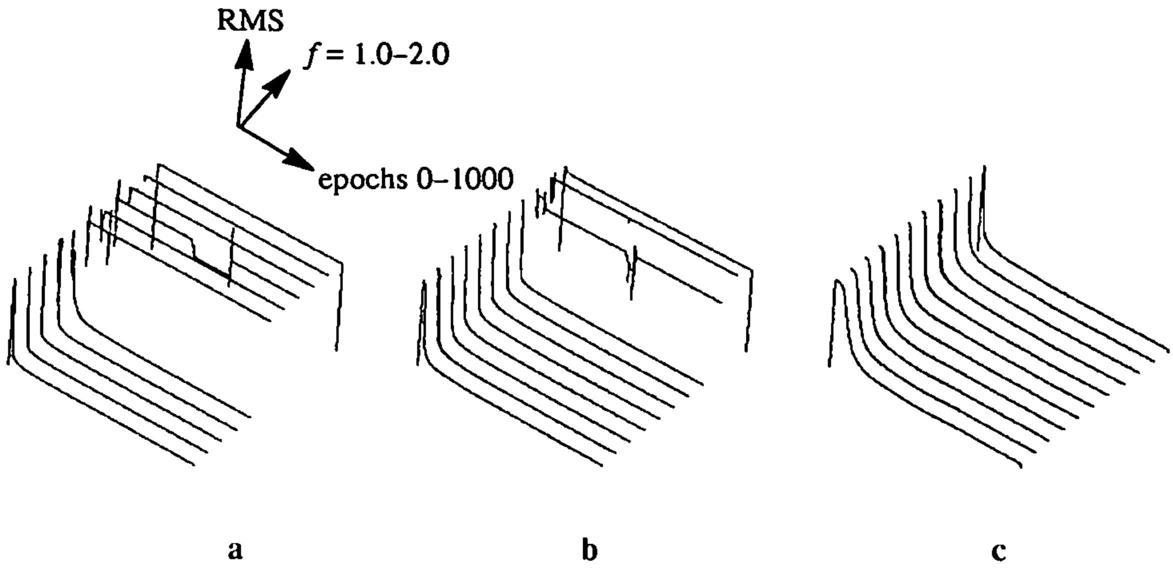


Figure 23: Trials with increasing gain of input patterns:
a) $\eta = 0.5$, b) $\eta = 0.2$, b) $\eta = 0.05$.

6.3.4 Signal/noise ratio experiment

In these experiments additions to the elements of the input vectors were made uniformly across each vector. This has the effect of increasing the noise but leaving the signal the same. Experiments were made with additions of 0, 0.1, 0.2, 0.4, 0.8, 1.0, 1.5 and 2.0 to the vector elements. Since the input vectors were normalized before use an addition of 0.1 would have the effect of (roughly) doubling the size of that input to the bottom layer of the network. Difficulty with learning was seen at noise levels of above 0.8 (Figure 24). Oscillation in the network error occurred although learning did eventually take place. With higher levels of noise (1.5 and 2.0) no significant learning took place within the 1000 epochs that the trial was run for, although it seems likely that given time these inputs too would have been learnt.

Most clearly these results indicate that selected pre-processing of the data input to a network can aid the learning of the data. The results in section 6.3.1 do not give any useful technique for improving the learning ability of a network. Size of training set is quite likely to be determined by the problem, and attempts to reduce this amount of data seem to be wrong-headed, as one may well be discarding important information. As is observed in section 6.3.1, in all likelihood the correlation between members of a class will be greater than that with the randomly constructed vectors, so increasing the amount of data will not

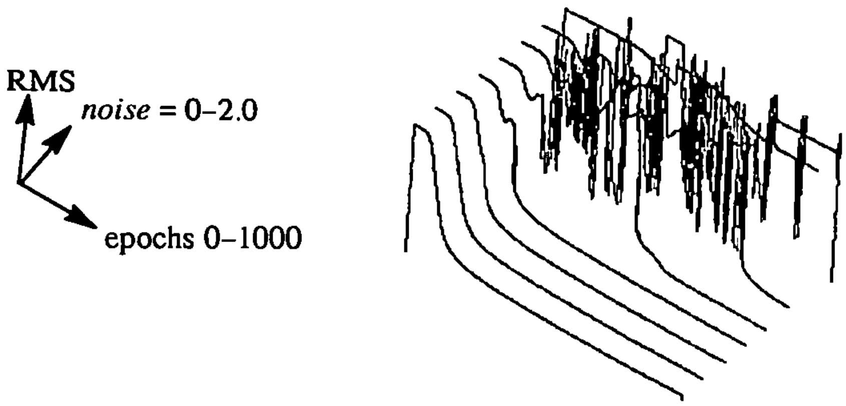


Figure 24: Trials with increasing addition of noise.

have such dramatic effects on the learning procedure. The hypothesis raised by the experiments in section 6.3.2 (regarding length of input patterns) is also unlikely to be true of real–world data, for the same reason.

The experiments reported in section 6.3.3 suggest that the absolute size of input activations is important in enabling a network to learn. Indeed in the following experiments normalisation was carried out on all data. Section 6.3.4 suggests that the signal/noise ratio is also an important characteristic of the input patterns. Increasing this ratio appears to prevent disruptive oscillation in the network's adaptation from occurring. It is often the case that this can simply be done with a set of data, and following experiments bear witness to the success of such a ploy.

The above results illustrate the significant effect that various data parameters can have on the behaviour of networks. The data used in the experiments above were randomly generated vectors, randomly assigned to categories. This does not produce classes typical of real data sets. Hence it was considered desirable to investigate other facets of input data to see their relationship with network behaviour. The final experiment of this chapter investigates the importance of within– and between–class correlation of patterns, by using patterns that although randomly generated, bear certain resemblances to other members of a class.

6.3.5 Within– and between– class correlation

The Pearson product moment correlation (see section 5.4.2) was taken as a suitable measure of correlation. Since for two vectors, an element plays the same role in each vector if it has the same position (i.e. is the same element of both vectors) the correlation between two vectors can be calculated. In our experiments we have n vectors, divided into two groups or classes. For each class an assessment of correlation between all the vectors in that class can be made by calculating the mean correlation of any two pairs in that class. This will give us a guide to how closely correlated the group is as a whole, and is referred to here as a 'within-class' correlation. We can also take a measure of how well two different classes correlate with each other, again by taking the mean correlation of all possible pairs between classes. This is called the 'between-class' correlation.

MEAN CORRELATION		
WITHIN CLASS A	WITHIN CLASS B	BETWEEN A AND B
0.037	0.021	0.020
0.021	0.071	0.035
0.221	0.006	0.010
0.095	0.165	0.019
0.337	0.254	0.083
0.373	0.315	0.044
0.290	0.448	0.135
0.599	0.586	0.040
0.604	0.648	0.054
0.815	0.842	0.438
1.000	1.000	0.152

Table V. Within- and between- class correlations.

For each experiment a training set of two classes was constructed. Each class consisted of a random vector, and other vectors within a certain tolerance of that vector. In this way it was possible to construct a class to a (roughly) given within-class correlation. Since these classes were based on random numbers between-class correlation was low. Table V gives the actual correlations recorded and the learning curves are illustrated in Figure 25.

It can be seen in Table V that the technique of vector construction has in general had the desired result of an increasing within-class correlation. This has the effect of increasing the speed of learning of the network. This suggests that both within- and between-class correlation, while not parameters to be explicitly manipulated, have an important consequence for a network's learning behaviour, and so can be used as a measure of suitability of a data set for network learning.

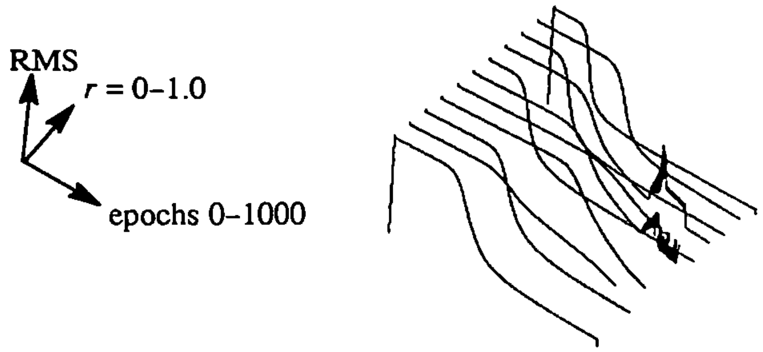


Figure 25: Trials with increasing within-class correlation.

6.4 SUMMARY

This chapter has equipped us with a working back-propagation network, and some ideas as to how to best prepare the problem data for the network. The particular code use for the network simulation has been shown to work for some standard problems, namely that of exclusive-or and data compression. Experimentation with input patterns to the networks has shown the importance of some parameters, primarily the signal strength and signal / noise ratio. As will be shown in the next chapter, some of these ideas are exploited in the plankton problem, with some success. The next task is to apply the network to a problem in the domain of interest, that is, *Ceratia* plankton, and to assess these method usefulness in this context.

Chapter 7. Ceratium Trials.

7.1 INTRODUCTION

This chapter concerns the experiments performed training a network to learn to classify the *Ceratia* species introduced in chapter 4. Initially trials using raw frequencies of the images were used, with mixed results. Latterly, an additional step of pre-processing is introduced that has the effect of significantly improving network learning performance. Further trials are performed investigating the various effects of different training conditions on the data.

7.2 *CERATIUM* LEARNING TRIALS

Having ascertained that the network simulations were working appropriately, the initial trials on the *Ceratium* data set were carried out. The obtaining and subsequent pre-processing of this data is fully described in chapter 4. All that need to be reiterated here is that the processed data set consisted of a set of normalised spatial frequencies for each individual specimen. Initial trials attempted to train a network explicitly with these data.

7.2.1 Raw frequency experiments

7.2.1.1 Raw frequency data

The classifications by the 'best' (see sections 4.3.2 and 4.3.3) three experts were taken as the 'ground truth' in classifying the images. A set of 49 *Ceratium longipes* and 49 *C. arcticum* were used for drawing both the test and the training data from. From the original set of 63 frequencies collected from the Fourier Transform (FT), a quick investigation showed that the form (or rough shape) of an individual specimen was distinguishable in an image reconstructed from only the lowest sixteen frequencies (see Figure 26b). Since the experts found sufficient information to classify the images in these low frequency reconstructions, the FT's were truncated to just sixteen frequencies, and these data used for both the training and test set.

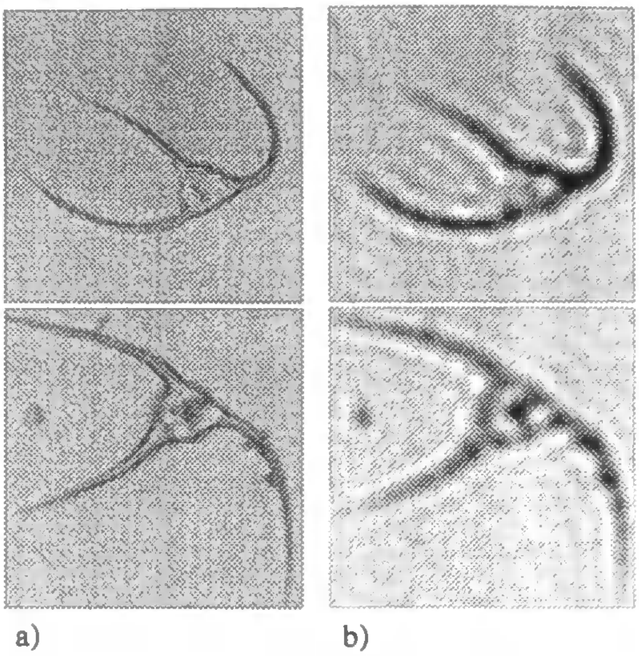


Figure 26: a) Grey-level images of (top) *C. longipes* and (bottom) *C. arcticum*, b) Reconstructed images of a) using lowest 16 frequencies.

The first *Ceratium* trials used networks of a [16 3 2] structure (16 inputs, 3 hidden units and 2 outputs, see Figure 27). The inputs were normalised vectors of the lowest sixteen frequencies of an image. The outputs were desired to be [0 1] for one species, and [1 0] for the other. The learning rate (η) was set to 0.5, the momentum (β) was set to 0.9.

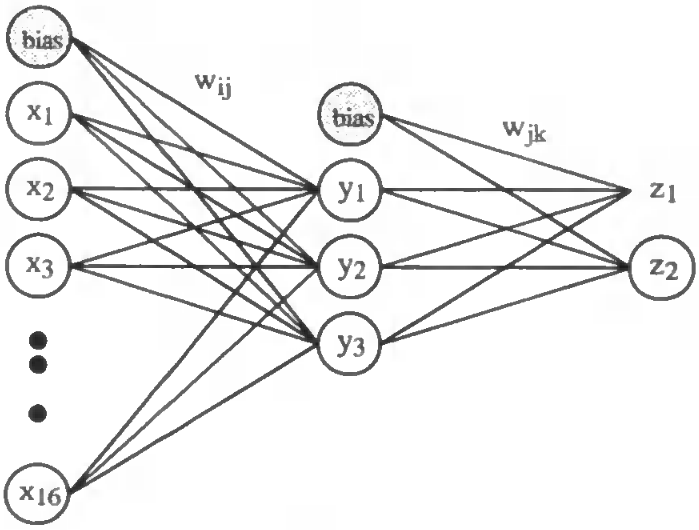


Figure 27: A feed-forward network with 16 input units, 3 hidden units and 2 output units.

7.2.1.2 Raw frequency results

A number of trials were performed, each with different initial network conditions (weights and biases) and different selections of data for training and test sets. The typical behaviour of the network in trying to learn to classify the raw frequency data is shown in Figure 28. As can be seen, the network falls prey to oscillations of the error measure. Although there are times in the learning cycle when the network is able to classify the training data reasonably well (see the extended period from about the four-hundredth to the nine-hundredth epoch in Figure 28) it is prone to jumping out of these periods and suddenly having a very poor performance measure indeed. The classification performance (in contrast to the RMS error and as described in section 5.4.5.1) measure is illustrated in Figure 29. Note that this error is typically less than the RMS error, since a pattern that is giving a significant RMS error may in fact be causing an output nearer to the target output than any other, and so be considered correctly classified using the nearest neighbour technique.

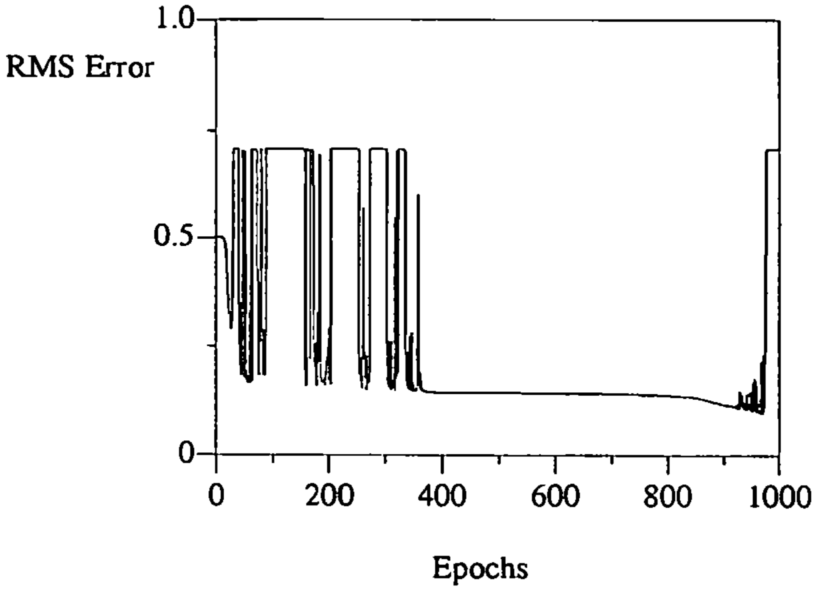


Figure 28: RMS error of a network on a *Ceratium* frequency trial.

Table VI shows the state of 10 trials after 1000 epochs learning each. Following the nomenclature of Chapter 3, t is the number of patterns that the network was trained on, selected randomly for each trial. t_c is the number of training patterns correctly classified (by the classification performance or Nearest Neighbour measure) by the network. g is the size of the remaining test set, hence t+g = 98 in these trials. t_c / t % is the percentage of the training data classified correctly, and g_c / g % the percentage of the test data classified

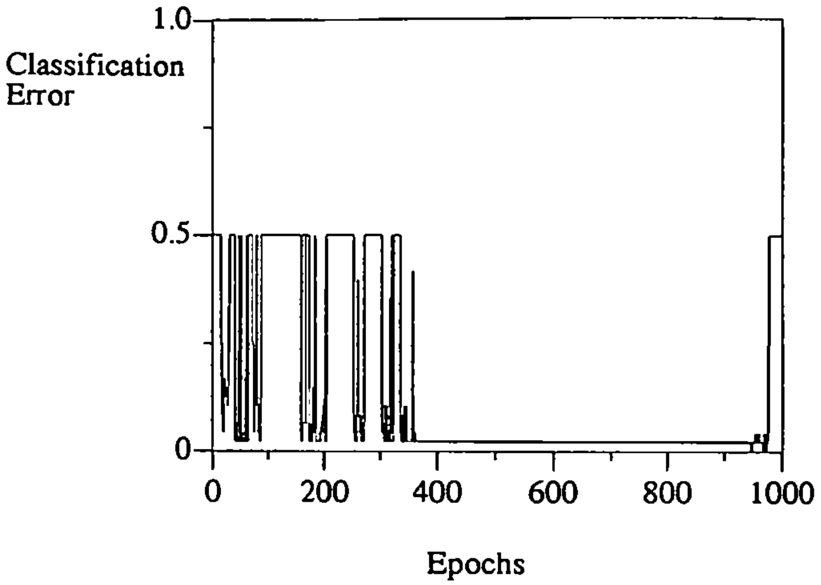


Figure 29: Classification error of a network on a *Ceratium* frequency trial.

correctly. $P(t_c\ /\ t)\%$ and $P(g_c\ /\ g)\%$ are the *probability performance* measures, given the size of the particular training and test sets respectively (see equation 5-17, section 5.4.5.4). For example, in row one, a performance measure of 50% is equivalent to a probability performance measure of 0%, since the likelihood is great (1 in fact) that such a performance could be by chance (say by tossing a coin). In row six, a classification performance of 100% is rated as a probability performance of only 88% due to the small

training set size (t)	learnt patterns (t_c)	trained score %	$P(t_c/t)$ %	test set size (g)	correct classn. (g_c)	test score %	$P(g_c/g)$ %
48	24	50	0	50	25	50	0
78	74	95	95	20	16	80	79
60	30	50	0	38	19	50	0
38	19	50	0	60	30	50	0
50	46	92	92	48	39	81	81
4	4	100	88	94	74	79	79
74	62	84	84	24	21	87	87
48	46	96	96	50	39	78	78
56	53	95	95	42	31	74	74
56	53	95	95	42	31	74	74
Mean Probability Performance			64.5				55.2

Table VI. Network performances on *Ceratium* frequency trials.

size of the data set under consideration (i.e. only 4 patterns). Table VI shows that over 10 trials shown, a mean probability performance of 74% was attained on the training data, and 64% on the test data (i.e. generalising to novel patterns).

The main problem found with this series of trials was the difficulty in knowing when to halt the learning at a good performance level before the network suddenly degraded. There certainly appears to be some inherent instability in training a network on these data. Various attempts to reduce this effect were tried, and eventually a procedure that resulted in more stable learning was found.

7.2.2 Frequency gradient experiments

7.2.2.1 Frequency gradient

The next set of learning trials included an addition stage of pre-processing of the data - namely, taking the frequency gradient of the images. A justification for this step is that the simple frequency gradient retains the 'shape' of the frequency histogram without reference to the absolute values of the frequency histogram. These absolute values of the power spectrum are related to the contrast in the pixel intensities in the image. Transforming an image with a dynamic range of [0 10] compared to an image with a dynamic range of [0 1] results in a frequency histogram of 10 times the magnitude. In other words, multiplying the image intensities by some amount multiplies the frequency histogram by that amount. Differences in image contrast typically arise from different lighting, and/or image capture conditions (i.e. focus), and are not relevant information in deciding what class an image belongs to, by virtue of not being properties of the object itself. By transforming the frequencies into the frequency gradients this source of noise is removed, and yet no shape information lost. The gradient is obtained by simply taking the local difference of neighbouring frequencies, so that a frequency vector $F = \{f_i : i = 1, \dots, n\}$ is mapped into a frequency gradient $G = \{g_i : i = 1, \dots, n-1\}$, where

$$g_i = f_i - f_{i+1} \qquad \text{(eq. 7-1)}$$

The resultant vectors were used to train a network of [15 3 2] architecture (n.b. one less input node, see Figure 30), with the same learning parameters as above.

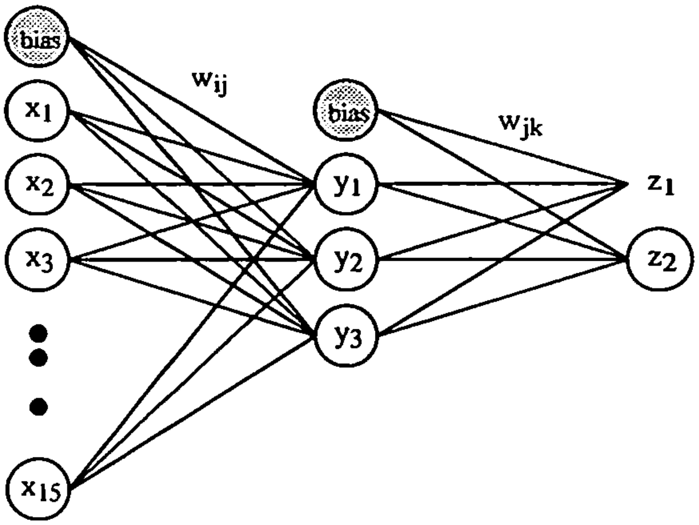


Figure 30: A feed-forward network with 15 input units, 3 hidden units and 2 output units.

7.2.2.2 Frequency gradient results

A typical learning curve is illustrated in Figure 31, with RMS and classification errors shown overlain. Since the learning is so rapid, a close-up view of the first 20 epochs of learning is also shown (Figure 31b). Figure 32 overlays the RMS errors curves obtained on the first 10 frequency gradient trials performed. As can be seen, in none of trials performed was there the violent oscillation observed in the raw frequency trials. Low RMS errors were achieved in every trial (under 1% error), and remained stable over the observed learning

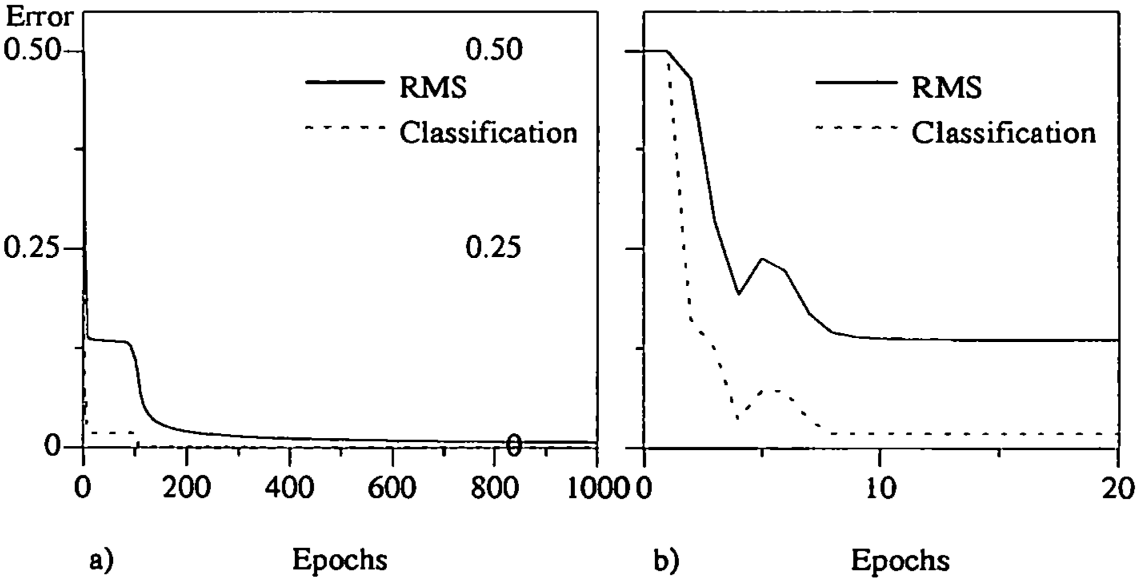


Figure 31: RMS and classification errors of a network on a *Ceratium* frequency gradient trial: a) 0-1000 epochs b) 0-20 epochs.

period. Extended learning trials of up to 20,000 epochs were done in some cases, the error measures remaining stable. The classification performance of the first 10 gradient trials is shown in Table VII.

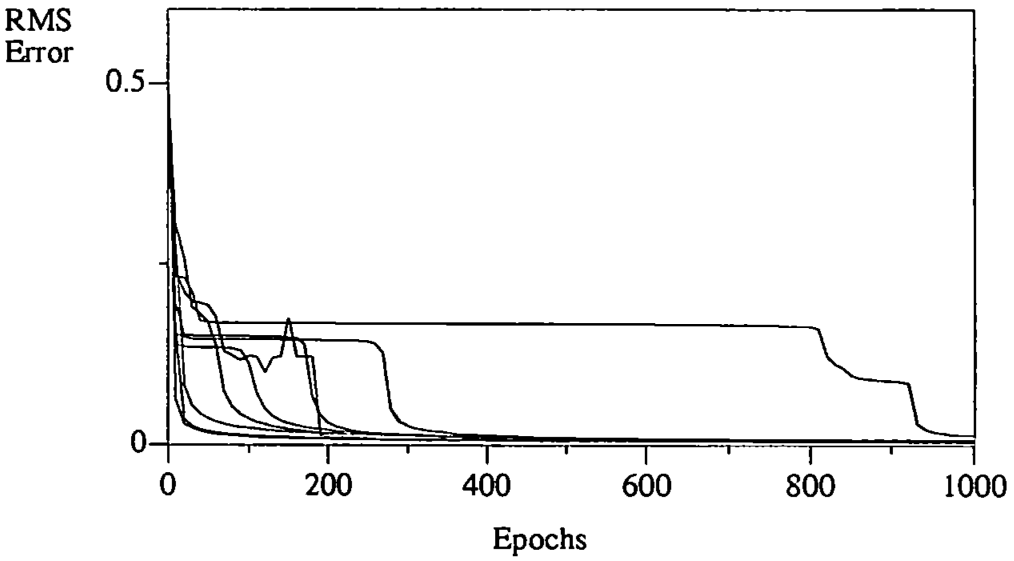


Figure 32: RMS error of 10 networks on a *Ceratium* frequency gradient trial.

training set size (t)	learnt patterns (t_c)	trained score t_c / t %	P(t_c/t) %	test set size (g)	correct classn. (g_c)	test score g_c / g %	P(g_c/g) %
38	38	100	100	60	44	73	73
44	44	100	100	54	45	83	83
28	28	100	100	70	54	77	77
72	72	100	100	26	21	81	81
56	56	100	100	42	34	81	81
16	16	100	100	82	57	70	70
68	68	100	100	30	26	87	87
48	48	100	100	50	41	82	82
42	42	100	100	56	46	82	82
52	52	100	100	46	38	83	83
Mean Probability Performance			100				79.9

Table VII. Network performances on *Ceratium* frequency gradient trials.

It seems that the step of taking the frequency gradients instead of the raw frequencies has led to a data set that learnt faster, and with more stability than before. Applying a t-test over a total of 40 trials, 20 in each condition, for the RMS error; $t= 5.17$, $df=38$, $p < .001$

(two-tailed test), and for the classification error; $t = 3.23$, $df=38$, $p < .01$ *(two-tailed test)*. These encouraging results led to the use of the frequency gradient data in all the following experiments.

7.3 NETWORK GENERALISATION

Once a technique for (almost) ensuring that a network can learn the required data is known, the focus of investigation becomes the ability of the network to do that task for which it is designed, namely, to generalise from the given training set to new and novel patterns. The ability of a network to be able to extrapolate from the training set critically depends on the quality of that data. 'Quality' is perhaps best described as the distribution of the training data - the sample population - in the n-dimensional input space, with regard to the characteristics of the classes (in this case species) themselves - the general population. The more complete and representative the sample population, the better the likely generalisation ability. Given the limited amount of data, multiple experiments were performed with different (often random) initialisation conditions - both experimental conditions such as size of training set, and network conditions such as weights and biases (see section 5.4.1). Section 5.4.2 discusses the technique of performing multiple trials under different sizes of training set in order to see the effect of this variable on both learning and generalisation.

7.3.1 Classification in the normal condition

The first trials of this nature were done on the frequency gradient data. 54 trials were run, each of 1000 epochs long, and the resultant error measures plotted against size of training set (Figure 33). The normal condition consists of on-line learning of the training set, the training set being repeated in sequence each epoch. No alterations were made to the data.

7.3.1.1 Normal results

It can be seen that in most of the trials a very low RMS error is achieved. The actual classification performance of the networks is rather better than this, as can be seen in Figure 34. It can be seen that good classification is obtained even with large numbers of training patterns. The generalisation ability of the networks is shown in Figure 35. Since

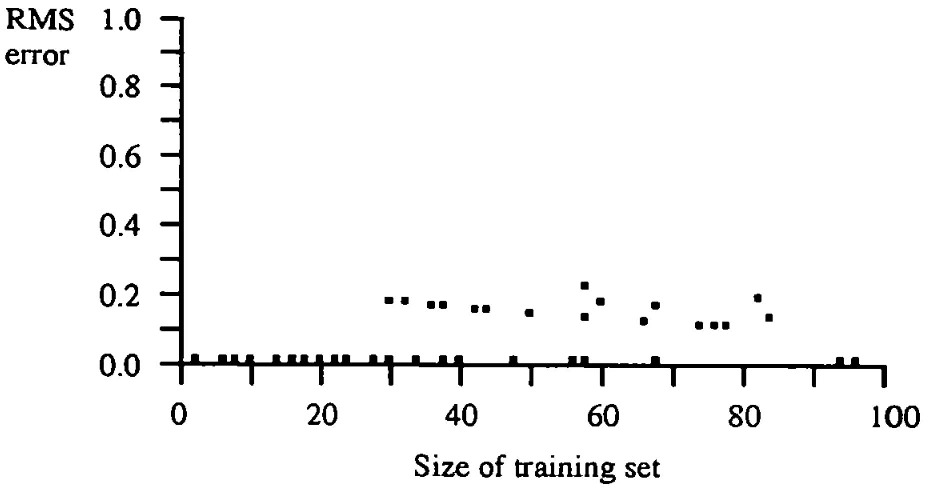


Figure 33: RMS error of 54 networks on *Ceratium* frequency gradient trials.

this figure shows the classification performance (rather than the RMS error) it is possible to calculate the probability performance of each network's state. Those performances that are statistically insignificant (at the 0.05 level of significance) are marked by a symbol. These are gathered around the 50% level of classification. Note that some performances at error

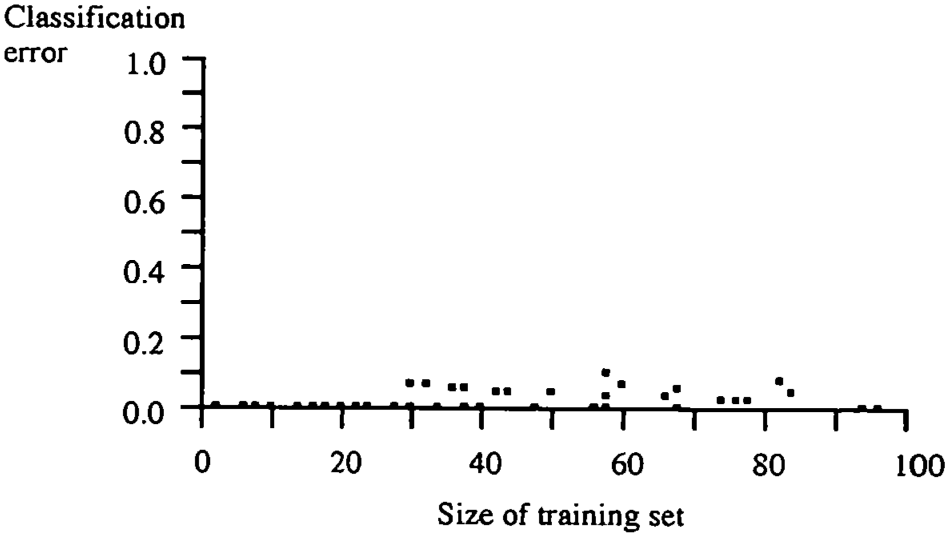


Figure 34: Classification error of 54 networks on *Ceratium* frequency gradient trials.

levels of *greater* than 50% (i.e. worse than chance) are statistically significant, that is unlikely to be due to chance. This remark is intended to show that the significance alone does not make for a good performance, rather, it should be considered when looking at *good* performances (especially over small test sets). The linear regression of y on x (see equation NO TAG, section 5.4.4) is also shown here, although the trend of improving generalisation with more training data is not significant in this case; $r = -0.095, df = 52, p > .05$ (*two-tailed test*). These results are summarised in the row 'Normal' of Table XI (below).

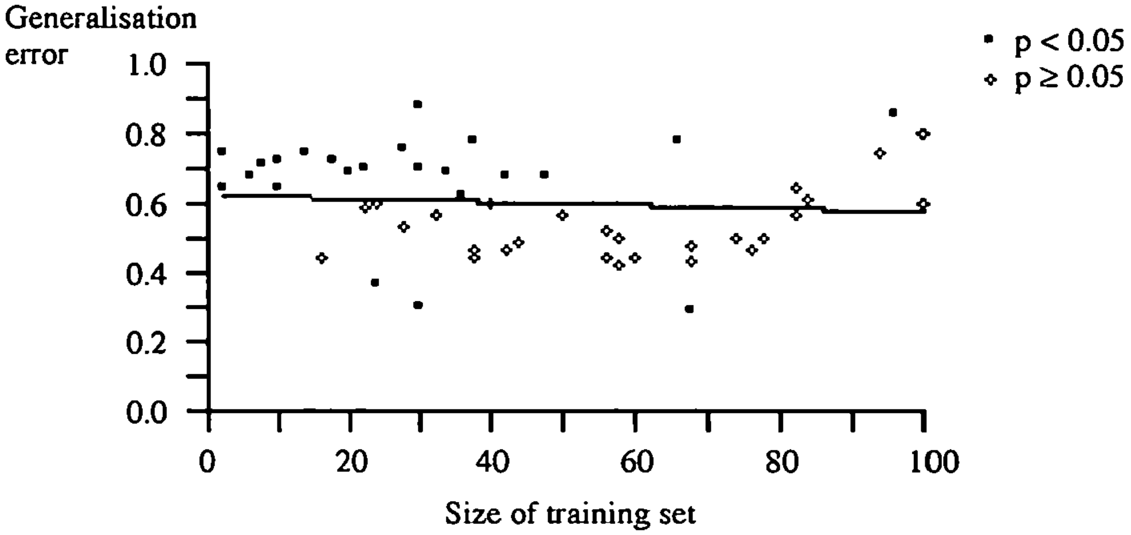


Figure 35: Generalisation error of 54 networks on *Ceratium* frequency gradient trials.

Trial Condition	Mean t	Mean t%	Mean g%	Correlation Significance
Normal	23	98.2%	39.8%	-----

Table VIII. Performance of networks on *Ceratium* frequency gradient trials.

7.3.2 Pattern noise experiments

7.3.2.1 Additive pattern noise

Some attempts were made to improve the generalisation performance of the networks. Two alterations to the training conditions were made. The first was to add noise to the input patterns. This has been suggested by a number of authors as a method of i) adding noise to the learning algorithm so as to aid the gradient descent (e.g. GyÖrgyi 1990, Krogh 1991), and ii) increasing the size of the training set (Peeling *et al.* 1986, Elman and Zipser 1988). These methods effectively train the network on a continuously changing approximation to the true error surface, and this can enable a network to escape from a local minima by perverting the surface so that that point is (temporarily) no longer a local minima. HolmstrÖm and Koistinen (1992) add random noise to their input patterns drawn from the Parzen-Rosenblatt estimate of training data density. In the current experimentation noise was added by some random amount to each element of the input pattern. The magnitude of the noise was random, but the amount of permissible noise (i.e. the bounds on the randomness) on each element of a vector was kept to within plus-or-minus one standard deviation of that element of the vector across the training set. Thus, the inputs reflecting differences in neighbouring spatial frequencies, if one portion of the pattern (say the difference between frequency bins 5 and 6, input element 5) was very variable within a species, then large amounts of (random) variance would be allowed as noise. This was to ensure that the noise conditions roughly reflected the natural variance in the patterns of one class, in that consistent elements of that class of pattern would remain consistent, while elements that had a large natural variance would be allowed to be perverted by additional noise – "eliminating idiosyncrasies without obscuring truly representative features" (Hampshire and Waibel 1990). The standard deviation was taken across the patterns of one category, and from only those patterns in the training set, so as to ensure that no information resident in the test set was used in the training cycle. This principle of isolation is important as it ensures that the testing of the network is truly done on novel data, that which the experimenter has had no access to. In a real-world application new patterns *would* be entirely novel, and no information about them would contaminate the training data.

7.3.2.2 Additive pattern noise results

The generalisation results for these trials are shown in Figure 36. Comparing Figure 35 with Figure 36, it can be seen immediately that the introduction of noise into the training cycle has a beneficial effect. Firstly, the generalisation of the network is much improved at all sizes of training set, and secondly there now is a significant trend for this generalisation to improve with an increase in the amount of training data: $r = -0.650$, $df = 52$, $p < .001$ (*two-tailed test*). Note again a few cases where the network performance is not statistically significant according to the probability performance – three cases with a large test set – say, more than 80 items – where the performance is close to 50%, and one case with a small test set – 8 items – and around 40% performance. It appears that the addition of noise to the input patterns has improved the generalisation by a significant amount: t=15.5, df=106, p<.001 (*two-tailed test*). Adding random noise to each presented pattern effectively increases the size of the training set. The constraints on the amount of noise added to each element of the

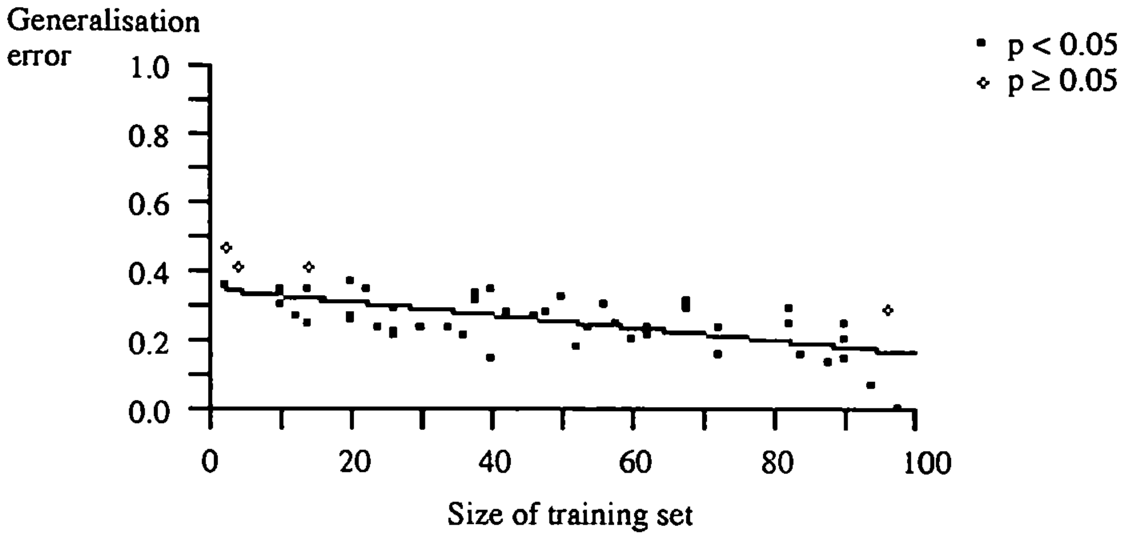


Figure 36: Generalisation error of 54 networks on *Ceratium* trials with added noise.

input pattern are intended to ensure that the resultant patterns are plausible members of that particular class. The distribution of the 'new' enlarged training set is, one hopes, reflective of the general population, and so constitutes valid training data.

Trial Condition	Mean t	Mean t%	Mean g%	Correlation Significance
Normal	23	98.2%	39.8%	------
Noise	23	99.0%	73.9%	< .001

Table IX. Performance of networks on *Ceratium* frequency gradient trials.

Another interesting result is that the network classification performance on the training set is actually better than in the no noise (normal) condition (see Table IX). This appears counter-intuitive – the networks now trained on noisy data are classifying the *original* data better even though they *have not actually been exposed to* that original (i.e. undistorted) data. This finding is contrary to that in LeCun (1989), which finds that added noise improves generalisation at the cost of network performance over the training set (cited in Lacouture and Marley 1991). The effect of improved performance on training data is similar to that found in Posner and Keele (1968), where subjects classified the original prototypes faster than the actual patterns they were trained on – distorted versions of the prototypes. The hypothesis there was that the subjects were extrapolating from the distorted patterns to learn the underlying prototype itself. A network exposed to a set of data, which by construction is distributed around a central tendency, is likely to learn that underlying tendency, though this does not in itself explain why *better* generalisation is achieved. This may be due to the introduction of many new 'valid' patterns to the training set by virtue of the constraints on the added noise.

7.3.3 Random pattern sequence presentation

7.3.3.1 Training set order

An additional stochastic element was introduced to the training by randomising the presentation order of the training data. The same data and network architecture as for the normal case were taken, and a series of trials performed. The effect of a random training set order is to again provide the network with an opportunity for escaping local minima. The whole training set was presented each epoch as in the normal condition.

7.3.3.2 Training set order results

The results are shown in Figure 37. Again, an improvement in generalisation over the normal case is seen: $t = 13.6$, $df = 106$, $p < .001$ (*two-tailed test*), and again a significant trend in generalisation improvement with more training patterns is now observed: $r = -0.621$, $df = 52$, $p < .001$ (*two-tailed test*).

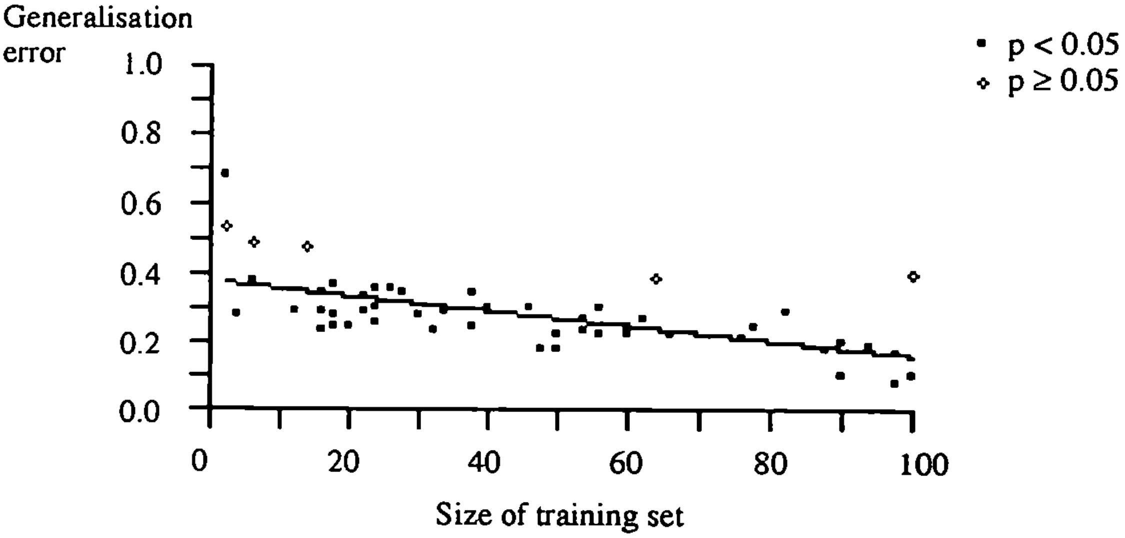


Figure 37: Generalisation error of 54 networks on *Ceratium* trials with random pattern presentation order.

Trial Condition	Mean t	Mean t%	Mean g%	Correlation Significance
Normal	23	98.2%	39.8%	------
Noise	23	99.0%	73.9%	< .001
Order	25	99.2%	71.9%	< .001

Table X. Performance of networks on *Ceratium* frequency gradient trials.

7.3.4 Combining pattern noise and sequence order

The obvious next step to take is to combine both the random presentation sequence with the pattern distortion to see if the beneficial effects of both remain.

7.3.4.1 Order and noise results

The result of this experiment is shown in Figure 38. It can be seen that again the generalisation is improved over the normal case: $t = 19.5$, $df=152$, $p < .001$ (*two-tailed test*). In fact there is a slight improvement in generalisation over either of the noise or order conditions alone, and there is a stronger trend to improving generalisation: $r = -0.670$, $df = 98$, $p < .001$ (*two-tailed test*). The results for these four sets of experiments are summarised in Table XI.

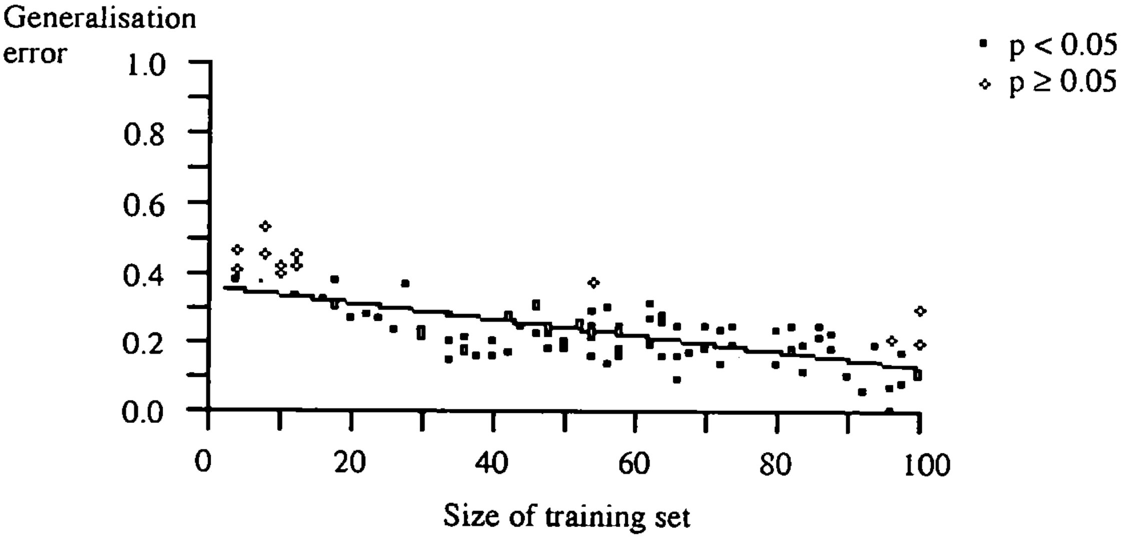


Figure 38: Generalisation error of 100 networks on *Ceratium* trials with random pattern presentation order and added noise.

Trial Condition	Mean t	Mean t%	Mean g%	Correlation Significance
Normal	23	98.2%	39.8%	-----
Noise	23	99.0%	73.9%	< .001
Order	25	99.2%	71.9%	< .001
Noise & Order	24	98.8%	76.9%	< .001

Table XI. Performance of networks on *Ceratium* frequency gradient trials.

7.3.5 Limiting receptive fields

7.3.5.1 Receptive fields

The change in a network's weights as it learns some task means that the amount of signal propagated from the various portions of the input to the various hidden nodes is progressively altered. Some connections will be strengthened, others diminished. This can be viewed as the allocation of the attention of the hidden nodes to the various input elements. In effect, the hidden nodes are developing their own receptive fields. In the next chapter, we shall look at some weighting strategies that have resulted from learning the *Ceratium* task. Prior to this, a generalisation experiment constraining the hidden nodes' possible receptive fields is presented. The initial network conditions were the same as for the experiments presented above (sections 7.3.1 to 7.3.4), with the exception that the network was *not* fully inter-connected between the first (input) layer and the second

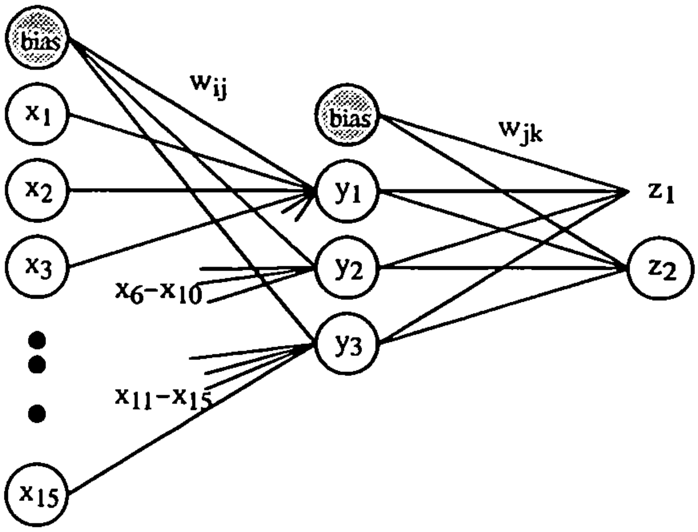


Figure 39: A feed-forward network with 15 input units, 3 hidden units and 2 output units.

(hidden) layer of units. Instead, each of the three hidden nodes was connected to five of the fifteen input nodes (in addition to the bias node). Node (2,1) was connected to the first five inputs (the lowest five frequency gradients), node (2,2) was connected to the next five inputs (numbered 6 to 10), and node (2,2) was connected to the last five inputs (11 to 15). Thus each hidden node received input from a distinct part of the input field. The training algorithm and parameters were the same as in the experiments reported above.

7.3.5.2 Receptive fields results

The results of this experiment are shown in Figure 40. Again, an improvement in generalisation over the normal case is seen: *t*= 40.36, *df*=152, *p* < .001 (*two-tailed test*). The trend in improving generalisation with an increase in training data is also observed: *r* = -0.458, *df*=98, *p* < .001 (*two-tailed test*). Table XII summarises the set of five experiments.

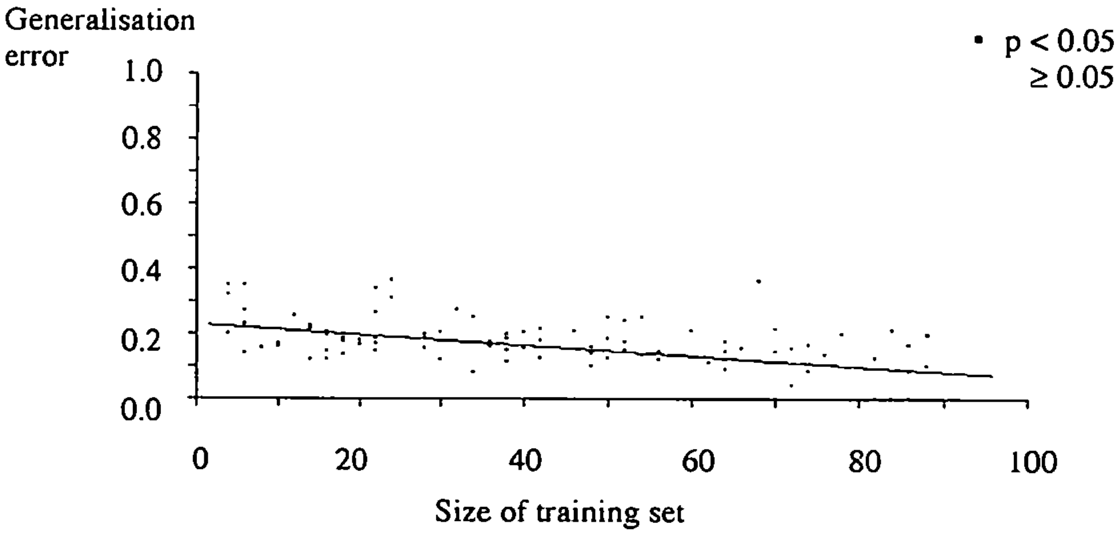


Figure 40: Generalisation error of 100 networks on *Ceratium* trials with limited receptive fields.

Trial Condition	Mean t	Mean t%	Mean g%	Correlation Significance
Normal	23	98.2%	39.8%	------
Noise	23	99.0%	73.9%	< .001
Order	25	99.2%	71.9%	< .001
Noise & Order	24	98.8%	76.9%	< .001
Receptive Fields	21	97.8%	82.4%	< .001

Table XII. Performance of networks on *Ceratium* frequency gradient trials.

7.4 SUMMARY

The task we had set ourselves - namely distinguishing between *Ceratium longipes* and *C. arcticum* - appears to have been achieved. A number of networks have been trained to perform such a discrimination, and some of them are performing very well indeed, (in some cases getting 20 out of 20 right, and another case getting 40 out of 41 right). This is all the more impressive given the behaviour of the human experts, none of who were entirely self-consistent. However, the trials performed so far are limited in the size of the problem, and the amount of data being used for testing the networks. Analysis of the working systems is needed to determine the extent to which this solution can be scaled up, and to investigate the actual workings of the networks in order to see if these can be made more apparent to us. This would enable a more efficient design of network to be developed.

Chapter 8. Network Analysis.

8.1 INTRODUCTION

Having shown that neural networks can learn an interesting classification task, inevitably the desire is to extend and improve upon the current devices. Designing bigger devices is not just a matter of building bigger networks. One needs to understand the way in which the devices work in order to be able to extend this ability. This chapter analyses the developed networks to accurately assess their performance, and to see if different architectures might be more suited to the task. The 'internal' workings of the networks are analysed in terms of the functions developed by the hidden units, and finally network performance is compared with an alternative classification technique.

8.2 SOLUTION ANALYSIS

8.2.1 Assessing generalisation

It has been shown that the *Ceratium* classification problem has been solved to some degree of success by back-propagation networks. How exactly can one assess this success, and more importantly assess the likelihood of scaling up the application to a massively increased amount of real-world data? One way of assessing attained performances is to compare them with those of the human experts. Treating the networks again raises the problem what to test the networks against. Treating the networks as supplementary humans and mapping their consistencies (as was done in section 4.3.2 with the human data) might be one way to check their performance, but there are two problems with this approach. Firstly, the networks are entirely (100%) self-consistent over fixed images (beating all humans) due to their determinate nature, and secondly, putting them in a consistency table (such as Table II of section 4.3.3) only indicates the level of agreement between a network and a number of humans. Since the network have been trained to agree with the 'best' human-classifiers, the level of agreement with the best three experts will be g%. Comparisons with the 'less expert' tells you less about the network's performance than about the human's ability. Another problem is that the networks are tested over limited amounts of the source data, and not the full set of images that the experts are. The only way around this problem of comparing networks tested on different, as well as different sizes of test set is to use the probability performance measure introduced in equation 5–17 of section 5.4.5.4. Figure 41 shows the distribution of this, along the straight classification performance, of 100 networks. The worst generalisation score was 64% (one network), and the best was 100% (five networks). One advantage of multiple experimentation is that one can of course select the best performing network for actual application. It is an impressive demonstration of the robustness of a neural network solution to a particular task that it can repeatedly discover solutions with high effectiveness. However, in an application one is most likely to settle for one given solution, and the fact there are many others generated is by the way. Multiple experimentation does not only give us a choice of networks to apply, but can provide statistical evidence of likely network behaviours.

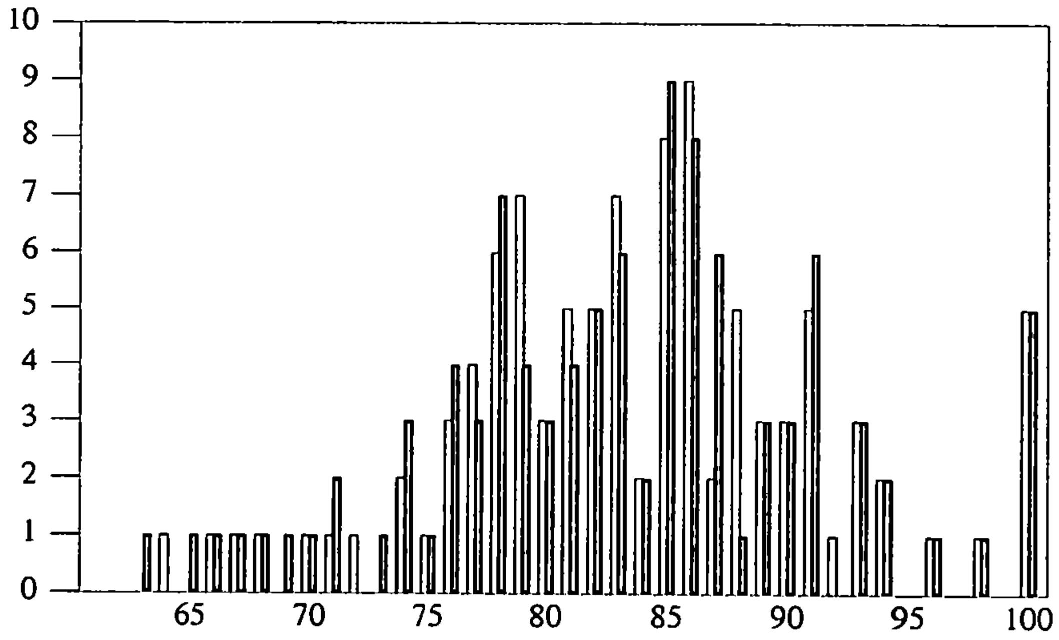


Figure 41: Performance error of 100 networks on *Ceratium* trials with random pattern presentation order and added noise, ▭ = classification performance, ▭ = probability performance.

Looking at the trends in generalisation, it can be seen that under some training conditions the generalisation improves as more data are included in the training set. As discussed in section 5.4.4, it is not possible to directly extrapolate from the linear regression line to predict generalisation performances with more training data. This is because the improvement must be non–linear, so as to avoid crossing the x–axis (which represents –

possibly unobtainable - 100% performance). An estimation of the possible improvement may be done by plotting the error on a logarithmic scale, as is done for the data of the trial condition 'noise and order' (see Figure 38) in Figure 42: $r = -0.431$, $df = 100$, $p < .001$ (*two-tailed test*). Unfortunately the spread of the data around the regression line is such that predictions of improvement in network performance must necessarily be hedged with caveats that poor behaviour can result from any size of data set. Predicting the *best* performance that might be attained with a certain size of training set is possible, Figure 42 suggesting that training sizes greater than those available in the current study might be required to obtain performances suitable for real-world application (say < 2% error).

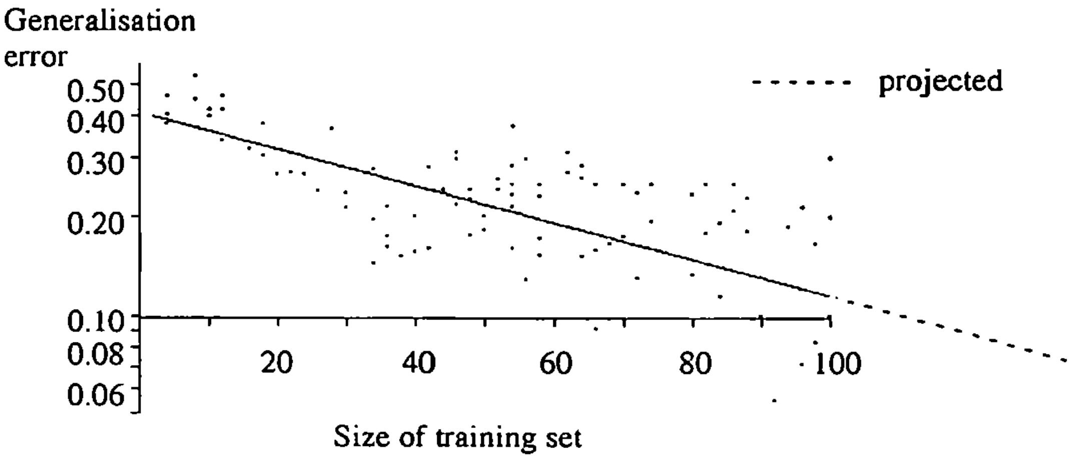


Figure 42: Generalisation error of 100 networks on *Ceratium* trials with random pattern presentation order and added noise.

8.2.2 Performance characteristics

Given the various performances of the neural networks it is useful to look at network responses to specific patterns. Although we are hoping that the distribution of a particular sample training set is representative of the world population (or even the particular test set used to assess a network), different selections of training data will undoubtedly have different distributions of patterns in them. The representativeness of a particular set of patterns is partly due to the size of the set. A perfectly representative set would be complete, with every member of that category included. Smaller sets are necessarily less representative in real world problems, in that they must overlook some possible patterns. Aside from consideration of which individual patterns are excluded, the degree to which a reduced pattern set is less representative depends on the complexity of the category distribution, in terms of the processing that is performed on the data. If a classifying algorithm is using Gaussian distribution assumptions, and these are true of the actual data, then a minimal training set (perhaps consisting of those patterns at the centre of the various distributions) will contain sufficient information to generalise well. As the deviation of the pattern characteristics from those assumed becomes greater, we find that more exemplar data is required for an adequate level of representation. Choosing such data at random from the world population may not result in the best training set of that given size.

It is hoped that an analysis of the performance of a range of networks on specific patterns will enable us to identify typical and atypical patterns. Over-selection of typical patterns may result in an over-representation of some areas of the pattern space. Having repeated instances of very similar patterns will not lead to a network improving its performance over one having access to just one of those patterns. This amounts to redundancy in the training set. Less typical cases might serve some purpose in indicating outlying category members, whether boundary cases, near the edges of that particular category, or members of some isolated outlying set. If a pattern is so untypical as to be unique then its inclusion in the training set will only serve to assist the identification of itself (which may or may not be important, depending on that patterns significance and frequency of occurrence).

Identification of the characteristics of the input patterns might enable a more sophisticated approach to the selection of training data to be taken. There has long been discussion of the importance of 'boundary samples' for instance, a boundary sample being some pattern that lies close to the boundary of its, or another, class (e.g. Lenet 1977). Ahmad and Tesauro (1988) have shown experimentally that such patterns are better training data (for future generalisation) than the more paradigmatic samples. This is reflected in some training algorithms that focus error-correction on those patterns that are maximally erroneous first, lowering the threshold as the errors reduce. These can be thought of as boundary samples in the output space. Cheung *et al.* (1990) have developed modified back-propagation algorithms that either increase the frequency of "poorly trained patterns" in the training set or increase the weightings these patterns carry in the learning cycle.

Pattern characteristics are assessed in terms of two measures. Firstly, the ease of learning, that is, the mean learnt response to a pattern over a number of trials (compared to the target response). Secondly, the ease of generalisation, that is, the mean test response to a pattern over a number of trials. The mean responses are taken over a number of trials, each selected to ensure that a pattern was included in the appropriate set (i.e. training or test) of that network for which a response is measured. The response is taken as that output on the appropriate node for that pattern (i.e. the activation of the *longipes* node for a *longipes* pattern). The mean responses for each pattern are shown in Figure 43. From the frequency histograms we can see that the large majority of the patterns are able to be learnt by a network. There are three patterns that achieve less than 90% mean learnt performance (it should be noted that this is not due to the poor performance of only a few networks, the patterns being in the training sets of many (52, 49 and 55) networks each). The spread of the patterns in terms of mean generalisation is much wider. As one might expect, there is a high positive correlation between the mean training and generalisation performances: $r = 0.582$, $df = 98$, $p < .001$ (*two-tailed test*). This fact can be interpreted as meaning that there are

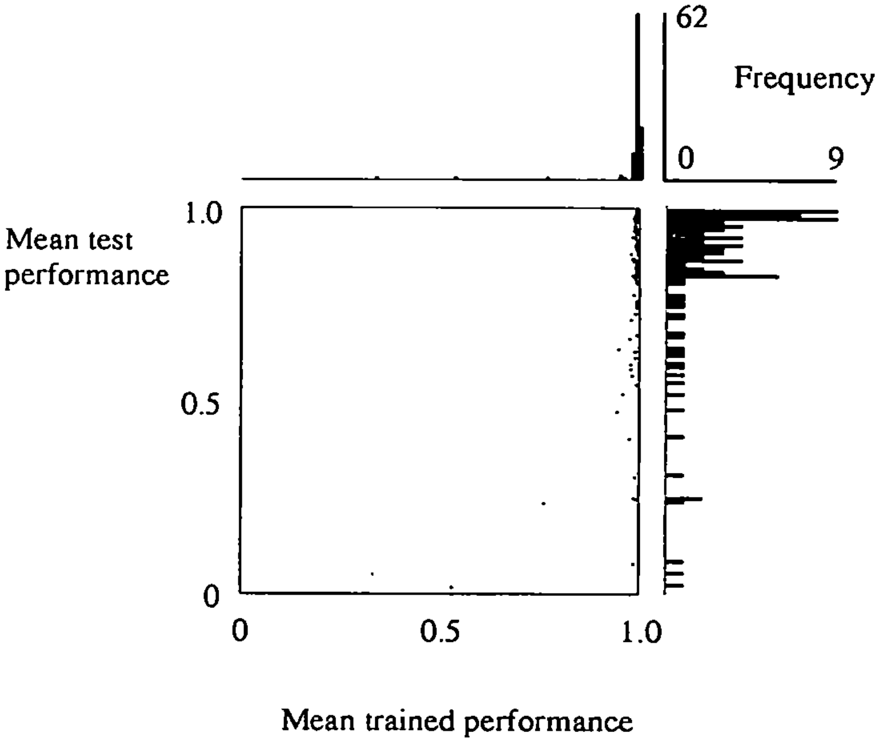


Figure 43: Mean training and generalisation performance of 98 patterns.

'rogue' patterns that are harder for a network to learn (presumably because they are dissimilar to the main corpus of data, and thus require separate representation), and that are not generalised–to well (presumably because they are not well represented in a randomly selected training set).

These rogue patterns can be considered in terms of human performance. An expert's responses to the *Ceratia* plankton can be regarded as being the human equivalent of the generalisation data gathered from the networks. Again we consider the classifications of the best three experts. It is then possible to compare the human and the network response to the patterns to see if the rogue patterns present as much difficulty to the humans as they do to the network.

PATTERN	MEAN RESPONSE (% CERTAINTY)		
	NETWORKS		
	TRAINING	GENERALISING	HUMAN
A (*C. arcticum*)	33 (33)	5 (4)	63
B (*C. longipes*)	66 (66)	23 (24)	16
C (*C. longipes*)	53 (53)	2 (0)	49
MEAN	98 (99)	81 (82)	74

Table XIII. Response to three *Ceratium* patterns.

Table XIII presents this data for the three patterns that have a mean *learnt* response of less than 0.9 on the appropriate output node for that pattern (interpreted as less than 90% certainty). It can be seen that the mean certainties of the responses across the pattern set are greatly above those of the rogue patterns, and that in two of the three cases a very poor (less than chance) certainty of human response is found.

The 'generalisability' of each pattern can be plotted for the network and the human case. This is done in Figure 44, and a slight correlation between network and human classification difficulty is found: $r = 0.187$, $df = 96$, $p < .05$ (*one–tailed test*). The same correlation measure was calculated using the classifications of all nine experts, resulting in a less significant correlation measure: $r = 0.148$, $df = 96$, $p > .05$ (*one–tailed test*). This partly due to the very wide scattering of decisions between the nine experts, some apparently guessing in many cases. It seems that the less expert find most (and not just the rogue) specimens to be of uncertain classification.

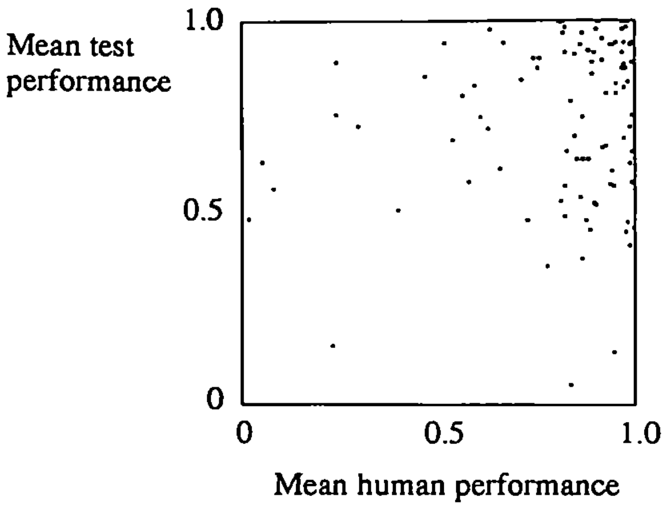


Figure 44: Mean network and human generalisation certainty of 98 patterns.

8.2.3 Analysis of network solutions

Given the two (weight) layer architecture of all the networks in the experiments run above, it is clear that all the internal processing of the data is done either going into, or coming out of, the hidden nodes. Each hidden node can be regarded as a *feature detector*, that is a node that selectively looks at the input signal, maximally responding to some feature of combination of features in the input layer, and hence providing evidence for the occurrence of that feature to which it is attuned to the output layer. The 'selective attention' is by virtue of the strengths of the connections from the input nodes to that hidden node. For a node to be maximally excited it requires as much incoming activation as possible. Given that the the input patterns are normalised, the way to impart a maximum signal to a node is to construct an input pattern with all the signal on that portion of the pattern that has the largest connection to the node. With patterns that are not of our own construction, however, one is likely to find that the signal is distributed across the vector. What a large weighting on one particular input line does tell us is that fluctuations in that element of the input are going to strongly affect that node. The amount of evidence a node provides for a particular feature is determined by its activation. Thus the analytical strategy is to look for what feature detectors are being developed by the networks, and to determine the importance of each. Rather than look at one particular solution found by one particular network, a comparison of number of well-performing solutions is done.

8.2.4 Feature detection

8.2.4.1 Fully-connected networks

Each hidden layer in the experiments presented above consisted of three hidden nodes. First we will consider the fully-connected network trials (using data from the experiment described in section 7.3.4, training with noise and random pattern presentation). It is desirable to compare the various network solution states with one another, to see if any commonalities exist. One cannot, however, compare the hidden layers of two networks by simply comparing the first hidden node of network A with the first hidden node of network B, and so on. The structure of the hidden nodes is unordered, that is, two networks performing identical processing of some data might do so on different nodes. A comparison must be made between correlated nodes, in the sense of both performing the same function. To compare two hidden nodes one needs to discover which node is correlated with which, if any at all. This was done by considering all possible permutations of matching two groups of three nodes, i.e. [A1=B1 A2=B2 A3=C3], [A1=B1 A2=B3 A3=B2], and so on. This

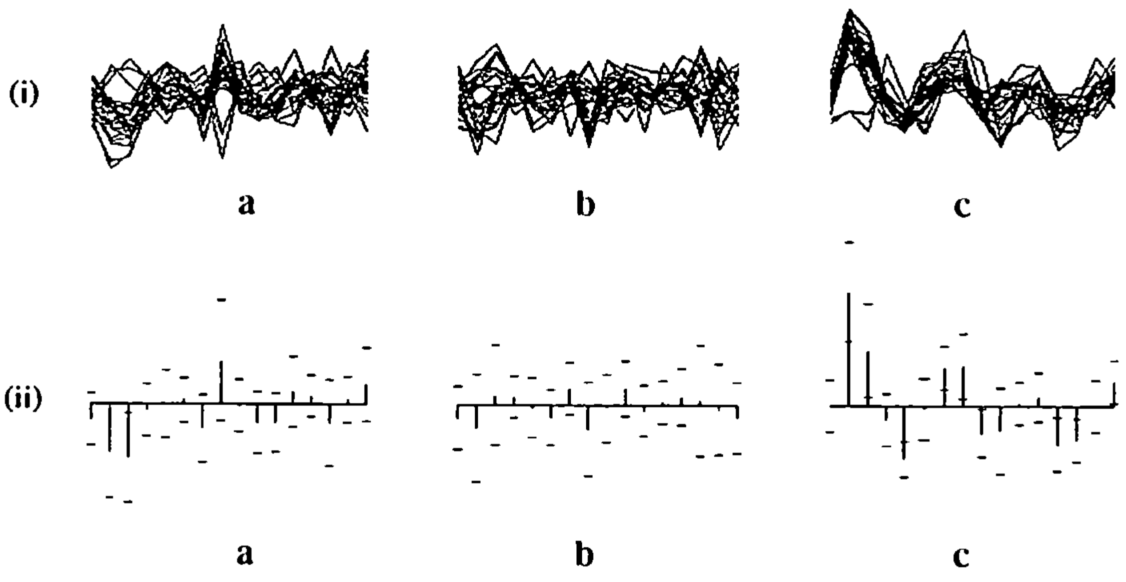


Figure 45: The weights going into the 3 hidden nodes of the 28 networks: (i) shows the 16 weights overlaid, (ii) shows the mean weights, and variance of ± one standard deviation.

gives six permutations to be searched for the best match. Each permutation gives three pairings of nodes to be compared, and the degree of match was calculated as the total Root Mean Squared distance between the incoming weight vectors of each node in a pair.

Figure 45(i) shows 28 sets of weights (all those networks with better than 95% generalisation, out of 100 total networks) plotted as line graphs on top of each other, and Figure 45(ii) as mean and standard deviation graphs. The sixteen elements of each diagram correspond to the bias weight plus the fifteen input weights to each hidden node.

Notable in Figure 45(i) is the strong structure on the node c illustrated. It appears that from arbitrary random initial states, and from being trained on differing training sets, many networks have come to develop a characteristic feature detector. From Figure 45(i) it can be seen that (with two exceptions) one node in each network has formed this particular detector. The strong peak on the left of Figure 45(i)c represents input node 1 (the bias is in position 1, the first frequency difference in position 2, and so on), and indicates that this part of the input signal is of significance to that particular hidden node. Lower spatial frequency information is represented at this end of the input signal, and the peak shows attention focussed on the difference between the first and second frequencies.

8.2.4.2 Limited receptive fields

In the experiments described in section 7.3.5, the inputs to each hidden node were limited to different parts of the input field. Because of this, each hidden node in a network can be identified with that node with the same set of connections in another network, thus rendering the node matching performed above unnecessary. Figure 46(i) shows 12 sets of weights (all those networks with equal to or better than 90% generalisation, out of 100 total networks) plotted as line graphs on top of each other, and Figure 45(ii) as mean and

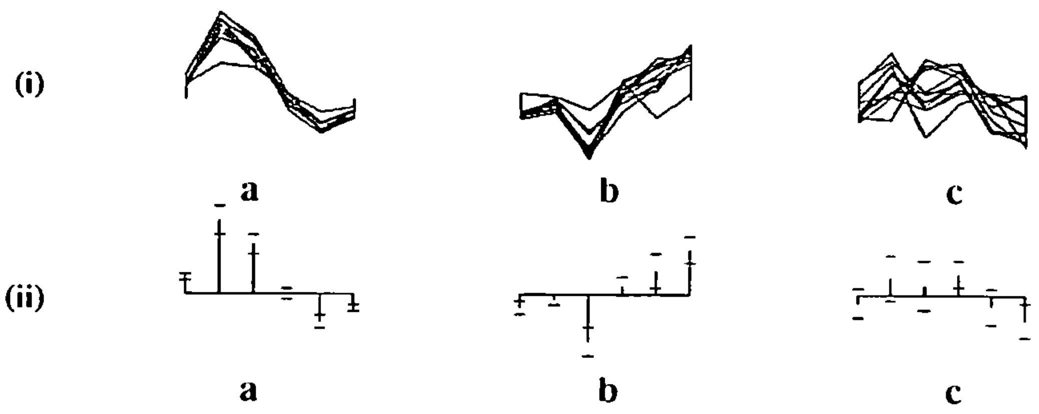


Figure 46: The weights going into the 3 hidden nodes of the 28 networks: (i) shows the 6 weights overlaid, (ii) shows the mean weights, and variance of ± one standard deviation.

standard deviation graphs. The six elements of each diagram correspond to the bias weight plus the five input weights to each hidden node.

Notable in Figure 46(i) is the strong structure on the node a illustrated. The strong peak on the left of Figure 46(i)a represents input node 1 (the bias is in position 1, the first frequency difference in position 2, and so on), as in Figure 45. As in this set of experiments each hidden node attends to a different part of the input, it can be more easily be seen that the lower frequencies are attracting similar weights across different solutions. There is a much greater degree of coherence between different weights assigned to the lower end of the input (compare the structure shown in Figure 46(i)a to Figure 46(i)c for instance. This is not entirely unexpected, since the gradients of the higher spatial frequencies will be due to higher frequency information in the image, less important for the grand structure, or shape, of the specimen.

8.2.5 Hidden node discrimination

To assess the importance of a particular feature detector, we can look at the significance of a hidden node on the output. One simple interpretation of the role of hidden node is to assess its discriminatory power with regard to the problem. All 15 input signals (plus the bias) are mapped onto an activation at each hidden node. For a hidden node to become a singly appropriate cue for a particular species, the activation caused by members of one species must differ from that caused by the other species. Any node that can discriminate well in such a fashion is a first-order discriminator, and may be used as strong evidence for a species by the output nodes.

Figure 47 shows the discriminatory power of the three hidden nodes in the best five performing networks. The response of a node to each input pattern is shown in the histograms, black and white representing the two species. It can be seen that certain nodes are separating the input patterns into their species better than other nodes. In the case of the first three nodes shown, representing one hidden layer of network 1, the third node is clearly responding differently to the two species to a greater degree than the other two nodes, and hence is a better indicator of species. This judgement can be given a qualitative value by considering the correlation between the responses of a node to the two populations. A high

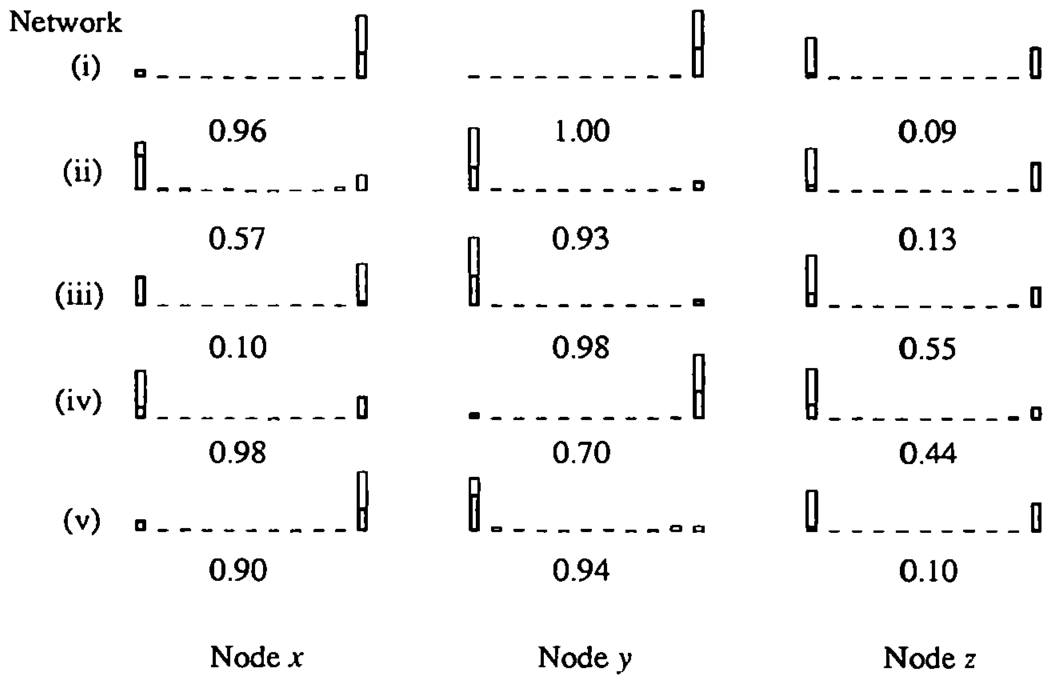
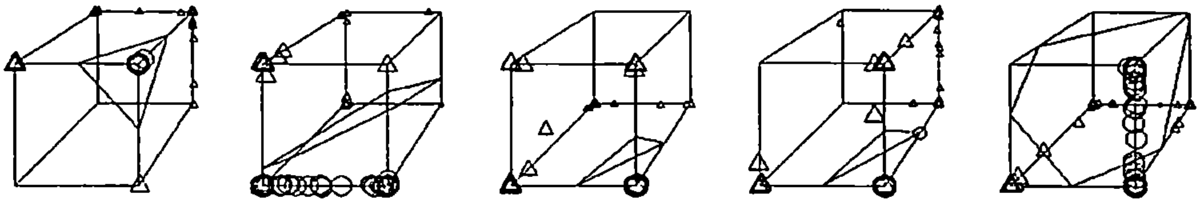


Figure 47: Five networks with three hidden nodes showing responses to input, black = *C. arcticum*, white = *C. longipes*. Numerals show correlation between responses to each species, (lower correlation = higher discrimination).

correlation would suggest that the responses to two different species were similar, and hence the node not discriminating well. Node *y* of network (i) has a high correlation between species response, and is a poor discriminator. Node *z* of the same network has a low correlation score, and is a good discriminator. One notable fact of this analysis is that the activations of the hidden nodes tend towards the limits of 0 and 1. The histograms in Figure 47 show this clearly. This effect may be due to the single layer of weights above the hidden nodes, resulting in a linear discrimination of the hidden layer representations.

Since each network considered has three hidden nodes, it is possible to plot the hidden layer activations of each input pattern as a three-dimensional diagram. Figure 48 shows these activations for the five networks considered above. Each of the 98 input patterns is represented by a symbol, a triangle for *C. longipes* and a circle for *arcticum*. The resultant activations on the hidden layer of a network are represented by position of the symbol in the cube, each axis standing for a different node. The plane dividing each cube shows the boundary where the output classification changes. For instance, network (i) shows a tight cluster of *arcticum* on one vertex of the cube. All specimen *arcticum* provoke a [1 1 0] response in the hidden layer (see Figure 47, network (i)).



(i) (ii) (iii) (iv) (v)

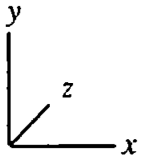


Figure 48: Hidden node activations for 5 networks,
○ = *C. arcticum*, △= *C. longipes*.

The second layer of weights connecting the hidden layer with the output layer have sliced off a small corner of the cube as classifying *arcticum*, leaving the remainder of the cube as classifying *longipes*. In the second example, it can be seen that the response to *arcticum* is restricted on two axes (y and z), but variable on the third (x). Again this is demonstrated in Figure 47. A comparison with Figure 48 illustrates the discriminatory power of the feature detector on node (iii) discussed above. The axis z representing node z, the effect of this node can be seen as limiting the set of *arcticum* patterns to the front planes illustrated in Figure 48. It can be seen in Figure 48 that the effect of this final layer of weights is not always as efficient as might be expected, and that some 'hand tuning' of these weights might result in a better classifier.

8.2.6 Weight symmetry

The implementation of a binary classifying network was not the single desired result. The major reason for doing such experiments was to show the feasibility of using neural networks techniques in that domain over a wider variety of data. For such a system to be of any practical use to (say) a marine biologist it would have to do more than distinguish *Ceratium arcticum* from *C. longipes*. The next stage would be to develop a network that could perform a larger number of discriminations. There are certain results obtained in the current study that are due to the binary nature of the discrimination, and it is not clear how these results relate to a larger number of possible classes (see also section 9.4.3).

The layer of weights evolved by the network feeding into the two output nodes tended to symmetry during training. Starting from random initial values, the weights feeding into output node 1 would become identical, except in polarity, to weights feeding into output 2. The reason for this is easy to see when one considers the desired outputs that the networks are being trained to produce. For any pattern, the target output is either [0 1] or [1 0]. Given that the two output nodes start with similarly small (but random) incoming connections, due to small initial connection weights, their outputs start similar (around 0.5). The error on one node is consequently around +1/2, and on the other -1/2. Thus the error signals are similar in all but polarity, and so the adaptations to the weights are similar in all but polarity. Even in cases where the outputs nodes' connections were initialised to significantly different values they would tend toward symmetry. Both have to interpret the same set of hidden activations, but to opposite interpretations. For any hidden layer activation pattern during training, one output node is trying to maximise its response, the other minimise its response. This can be seen by looking at the actual outputs produced, the two values tending to sum to unity. The two possible classes exhibit total redundancy, since the input is either one class or the other. No patterns belonging to neither of the two classes were used in training or testing the networks, hence the information of one output alone is sufficient to indicate the class of the input. In such a case we do not need two outputs at all, one output indicating either [0] or [1] would suffice. This is not the case where there are more than two possible classes, where it makes more sense to assign one output per class, and there is less redundancy in the output patterns.

8.2.7 Arbitration of multiple solutions

So far, it has been envisaged that a number of network trials be performed for some task, and the best performing network chosen from these for application. The various stochastic elements at play in the training of a network (the selection of training data, initial network conditions, pattern noise and so on) mean that one is likely to find a range of levels of performance to chose from. Rather than chosing a single network to apply to a problem, one can pool the results of a number of networks and use some criteria to make a decision based on this pooled knowledge. Hampshire and Waibel (1990) have such an arbitration scheme to improve the performance of a phoneme recognition network, although they use networks with different cost functions to produce a range of output values. In the current study the effect of arbitration between similar networks trained on similar training data was analysed.

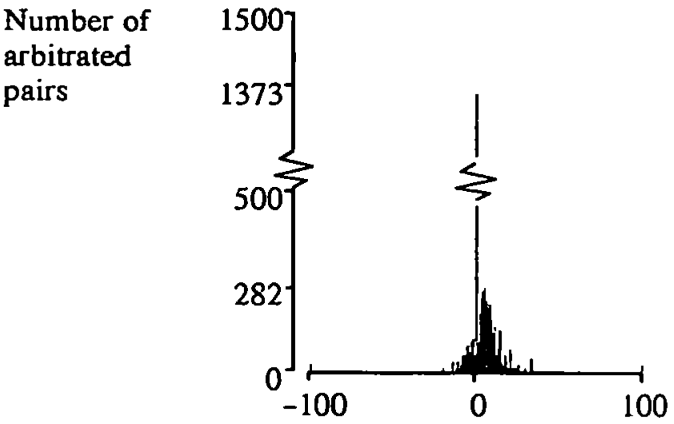
The premise behind the use of arbitration to improve classificatory ability is that the performance of one network augment the performance of another by correcting those mistakes of the second, and vice versa. It is hoped, for instance, that two networks, each with a performance of 90% (say), will not have entirely overlapping misclassifications on the pattern set, so that while their common error will remain uncorrected, errors unique to one network will be. Note that it is possible that one network may erroneously change the correct classification of the other's. The decision as to which network to choose where they disagree is made according to their output activations. Interpreting these as levels of belief, or certainty, one can choose the network with the most vehement convictions to over-ride the other.

There is of course no point in using identical (right down to the weights) networks for arbitration, since there will be nothing to arbitrate between. However, in the trials done, due to initial conditions and so on each network has a different set of parameters after training, and will behave differently to some degree. To assess the possible improvement in performance that arbitration gives between two networks, each possible (non-identical unordered) pair of 100 trained networks (giving [100×100−100]÷2) was taken and the resultant improvement in performance measured. Given that each network is exposed to a different set of training data, and hence tested on a different set of test data, it is important to measure each networks performance only on those items common to *both* test sets. Thus each network is given a new performance measure, taken across this shared test data. The post-arbitration decisions are then compared to these measures. Table XIV shows these results.

Pre-arbitration		Post-arbitration			*t*-test (related)		
Mean Score%	Best Score%	New Score%	New/Pre Mean	New/Pre Best	t	df	p
84.7	89.7	87.6	1.034	0.978	5.312	5048	< 0.01

Table XIV. Mean arbitration results.

Note that there is a 4.3% improvement in the arbitrated scores over the mean of the pre-arbitrated two scores. There is a 1.7% degrade in performance when comparing the arbitrated scores to the *best* of the two prior scores. In general, however, when applying two networks to a problem we will not know in advance which of the two is best over the world



Arbitration improvement over mean performance

Figure 49: Performance gain after pairwise arbitration.

population, hence measuring arbitration performance against the best network score implies access to information that one does not in fact have. Improving the performance over the mean is beneficial since we can be confident that *generally* using two networks is better than choosing one alone. Figure 49 shows the distribution of the arbitrated performances. As can be seen, there are many cases in which no change in performance, either degrading or improving, results from arbitration.

The above analysis is deficient in respect of the fact that each *system* (that is, combination of two networks) has had the advantage of a larger training set than either of the single networks alone. It is not surprising that an improved performance results. To assess the benefits of an arbitration scheme it is necessary to compare a single network system with a two-network system, where both systems have had *equivalent amounts* of training data. The amount of training data that a two-network system has had is calculated as the sum of the training set sizes of each network, minus the amount of shared training patterns. It is then possible to compare single versus double network systems, as is done in Table XV.

Pre-arbitration		Post-arbitration			*t*-test (related)		
Mean Score%	Mean Training Set size	New Score%	Mean Training Set size	New / Mean%	t	df	p
82.6		87.0		105.4	4.360	488	< 0.01
	37.0		38.0		1.025	488	> 0.1

Table XV. Mean matched arbitration results.

Selection of networks between certain bounds of training set size has to be made in order to compensate for the difference between the two distributions (one being larger than the other). A *t*-test statistic is given for the training set sizes to show that similar sets of trials are being compared.

We can see that there is a significant improvement in system performance when two networks are used as opposed to one. It is suggested that this is due to the different networks perhaps finding different solutions to the common problem, by virtue of different initial conditions, and being given (some) different training data. The good performance of each network is increased by the two correcting each others faults, mutual faults being less likely than singular faults.

8.2.8 Nearest Neighbour trials

Judging the network solutions to be effective raises the question of how simple the task actually is. A nearest neighbour cluster analysis technique (see section 2.3.6.4) was applied to the *Ceratia* classification task for a comparison of efficacy. Since there is no training schedule as such it is possible to perform many trials over different training data rapidly. 1000 trials were done, under as similar conditions to the network trials as possible. Selection of a test and training set from the database was done randomly, as was the selection of the size of the training set. No noise was added to the patterns in the training set since this would have effectively expanded the training set indefinitely. There is no sequential presentation of training items as in the condition 'order', since a complete search is done for each new input pattern. Comparative figures for t% cannot be given for the nearest neighbour method, since there is no training schedule, and hence no measure of training performance. A generalisation measure, g% can be calculated, and the effect of training set size on this figure is shown in Figure 50: $r = -0.377$, $df = 1000$, $p < .001$

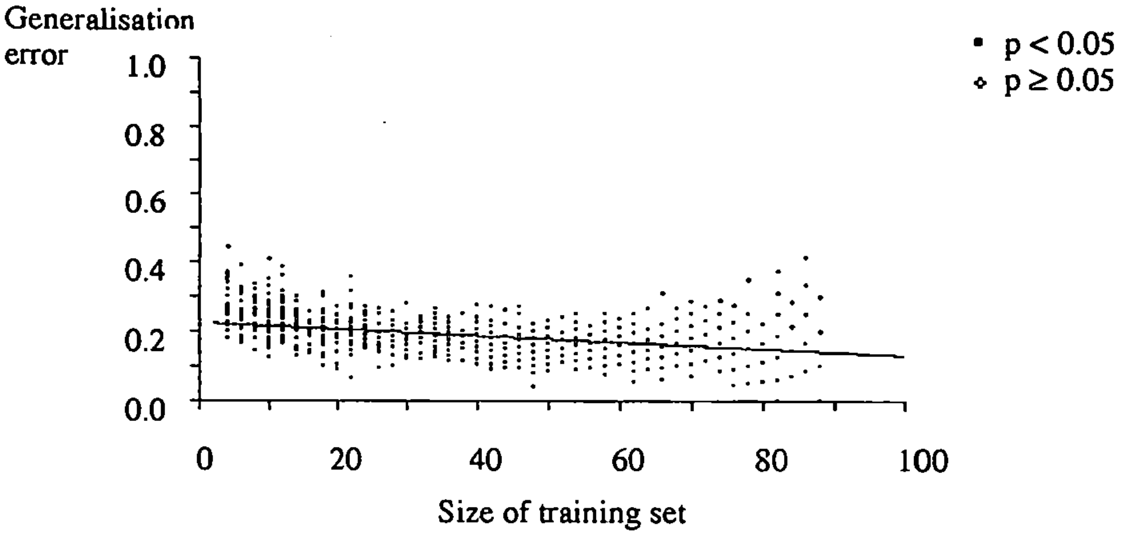


Figure 50: Generalisation error of 1000 nearest neighbour *Ceratium* trials. (*two-tailed test*). This diagram is more dense than Figure 36 due to the much increased number of trials performed. The effect of increased stratification can be observed as the size of the test set reduces, since the less test items there are, the fewer are the number of different scores it is possible to achieve. The results are summarised in Table XVI. The performance of the nearest neighbour algorithm averaged over all the 1000 trials was better than the mean performance achieved by the neural networks. The mean nearest neighbour performance is better than that achieved by the networks, but this is partly due to the fact that the nearest-neighbour algorithm attains high performances across a wide range of training set sizes, while the networks (in the noise and order condition) improve with an increase in the amount of these data, and thus their mean score is brought down by poorer performances over small training sets. Figure 51 shows the generalisation of the two techniques against size of training set. The nearest neighbour algorithm improves initially, but levels off as the training set size increases. The network algorithm shows constant improvement, and looks as if it may improve further as more training data are made

Trial Condition	Mean t	Mean t%	Mean g%	Trend Significance
Normal	23	98.2%	39.8%	------
Noise & Order	24	98.8%	76.9%	< .001
Noise & Order (Nearest Neighbour)	22	------	82.7%	< .001

Table XVI. Performance of networks and nearest neighbour on *Ceratium* frequency gradient trials.

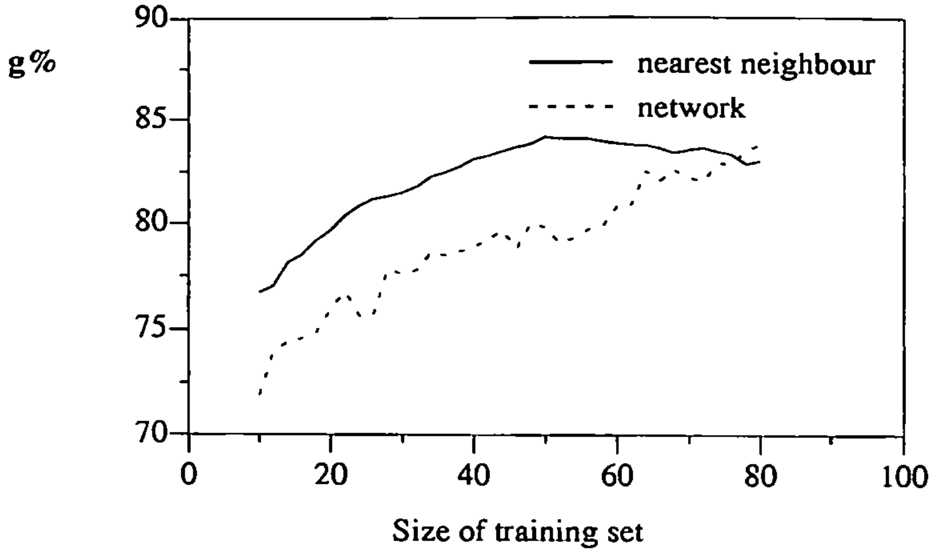


Figure 51: Generalisation performance of neural network and nearest neighbour *Ceratium* trials.

available to it. Tantalisingly, the network starts to better the nearest neighbour technique just at the limits of the current experimentation. However, we believe that the neural network system is shown to be a stronger technique than the nearest neighbour algorithm in view of firstly, its efficacy, and secondly, its computational advantages. It should be noted that the nearest neighbour technique is computationally expensive to apply compared to the network method, requiring memory for *n* training items (actually a database rather than a training set), and a search through each of these items for a nearest neighbour. It is exactly the extraction of information from the exemplars that allows a neural network to generalise without explicitly storing all the training patterns. An instantiated neural network will be able to classify at a much higher speed than a nearest neighbour algorithm, and requires much less memory.

8.3 SUMMARY

We have trained a number of back-propagation neural networks to discriminate between binarised images of two species of Dinophyceae, *Ceratium longipes* and *C. arcticum*. We have shown that several different types of solution to the task exist within such a neural network and that the first-order pattern classifications performed by the hidden layer nodes in each of the neural network examples tested have been shown to be similar in neural networks that had different initial starting conditions and random presentation sequences

Chapter 9. Discussion.

9.1 INTRODUCTION

This thesis has reported primarily on a successful attempt to develop a neural network system able to classify images of *Ceratia* plankton. This has involved the collection and labelling of images, the development of various pre-processing techniques for the images, finding a suitable network architecture for the task, training networks to perform the required task, and assessing the performances obtained from these devices. All this has been done with a view to the implementation of an actual device operating in the field, although this will require an extensive elaboration of the current work, and is discussed in section 9.4.4. However the results obtained to date are encouraging, with network performances approaching, and sometimes exceeding those of the expert, and repeatedly performing better than many of the less expert taxonomists currently performing this kind of classification.

A second aim of this reported work is to elucidate the nature of the feed-forward class of neural network model, with the hope that such knowledge might enable researchers to develop more effective networks generally. As has been stated before, it appears that neural network techniques can be applied to any number of problems in a wide variety of domains, yet we have little idea prior to attempting any specific task as to the feasibility of a neural network approach to that task. Much developmental work in the field consists of largely ad hoc attempts to resolve the issues of data representation, pre-processing, network architecture and so on. We have yet to find an all-encompassing theoretical framework that will enable an end-user to simply define a problem, specify the data and construct the required network. Until such a framework is available, network construction relies on various tools and techniques such as those presented here for developmental guidance. The current progress of the work has been possible by the development of a number of techniques for working in this domain. The assessment and labelling of the data sets we were concerned with was, perhaps unusually in image processing, not as straight-forward as one might expect. Furthermore, the limited amount of data to hand imposed additional constraints on the experimentation, and probabilistic and extrapolative techniques had to be developed to successfully work within them.

This final chapter discusses some of the consequences of the results presented in earlier chapters, and considers the current standing of the work presented, and where it might usefully be extended.

9.2 *CERATIA* CLASSIFICATION

9.2.1 Applying neural networks

Part of the appeal of neural network techniques is undoubtedly due to their (albeit superficial) neuromorphism. Many researchers have been giving up traditional AI techniques, regarding their reliance on ever increasingly long serial algorithms as unlikely to further advance machine intelligence a great deal. The key feature that neural networks share with the brain is massive parallelism. Second to this, is massively interconnected simple local processors. A third commonality (perhaps emphasised more due to the distinction between traditional AI and neural networks), is the ability to learn from experience. Neural networks, we are told, can learn by experience, that is, repeated exposure to their environment. This is good news for the researcher, since although we are weak on 'how-to's' for many cognitive tasks, we are strong on 'what-to's'. Generation of examples in many domains, including the current one, is easy. We can rely on the unknown processes of our (or an expert's) brain to inform a network what to do - a 'do as I do' approach.

This thesis has shown how neural networks can be applied to a specific classification problem with some success. One reason for attempting such a narrow task is as a feasibility study, i.e. to investigate the possibility of developing a larger system, a useful device capable of general application within the domain. This goal, if achieved, would enable the analysis of data to be carried out much faster and with more consistency than has previously been possible using human experts to classify the data. The chosen domain of investigation (marine plankton) is one of great importance for environmental and biological monitoring. As is generally true of the environmental sciences, the use of humans for analysis is a notable bottle-neck in the process of quantifying the sometimes extremely abundant environmental data. Given, in addition to this, both the increased recognition of (and hence desire for) taxonomic identification of marine organisms, and the reduction in numbers of personnel being recruited to and remaining in this field, then it becomes clear that the automation of this type of task is becoming increasingly important (Simpson *et al.* 1992). The two species (*Ceratium arcticum* and *C. longipes*) chosen for this preliminary work are morphologically similar and have sympatric geographical distributions. The classification and separation of these species is a real world task which is typical of those required of taxonomists. Extension of this work to a wider variety of similar species is likely to be regarded as useful for a number of reasons, for instance, the important economic consequences that populations of these species can have upon shellfish industries across the world necessitate the accurate observation and quantification of plankton blooms. Also, the consequential uptake of carbon into the oceans due to these organisms make their enumeration important in modelling the global climate, an increasingly important concern.

9.2.2 Network classification

The cognitive act of classification might appear to be a suitably modular task, such that one could develop a theory and perhaps instantiation of whatever it is we do when we recognise something as something. Pattern recognition itself though is the most basic act of intelligence, any other cognitive skill being unimaginable without some form of matching and abstraction, and (perhaps because of that) the most obscure. What we have in mind when we think of the act of classification is an act above this base level, a process typically consisting of an input to some sense, and then an attribution of that input to some defined class. Early research into cognitive categories used simple artificial constructs such as colours and geometric forms (e.g. Bruner *et al.* 1956, Johnson 1978) and failed to be successfully extended beyond these limits. Despite the success of a number of theories in a number of different domains, each tends to break down when they attempt to cover a wider area of ground. Classification of household objects, fruits, colours and faces may not be explained by one theory because they are different types of task (Smith and Medin 1981). Statisticians, understandably, have less qualms about picking, choosing, and even inventing the discriminant functions they need for a particular task than do cognitive scientists. What will be a suitable discriminatory function for a given set of data is an empirical question. A particular mathematical formula, serial algorithm or neural network structure might prove to be the most effective. A significant part of the appeal of the latter technique for many people is that suitable functions are 'discovered' by the network learning process itself, and are thus naturally suited to the data in question. However, the initial constraints of the network structure are as significant as the data to be learnt. Discovering a suitable network architecture for a particular problem is as empirical a problem as choosing a discriminant function. In fact, by selecting a network architecture one is dictating the number and type of parameters available for the mapping. Over-selection of parameters will lead to failure as surely as under-selection. The choice of pre-processing is also critical. Pre-processing inevitably reduces the amount of raw data, indeed, this is one of the primary uses for pre-processing in image processing. While one hopes that a particular technique retains that information essential for a particular task, it is possible to throw the baby out with the bath-water. It is safe to say that the extension of the current technique to a more complex classification problem is certain to prove more complicated than just adding extra hidden nodes and outputs, and some discussion of this is made in section 9.4.4.

9.2.3 The human case

In the work described in the previous chapters we have been less concerned with the need for a technique to 'ring true' (that is, mimic some behavioural data) than for the technique to be effective. An attempt was made to automate the classification of some data, namely images of *Ceratia*. The particular class of *Ceratia* was chosen for two main reasons: pragmatically, the two-dimensional nature of the specimens removes many possible difficulties in processing their images, and theoretically, the difficulty of the task is such that it qualifies as an expert classification, and yet interestingly remains partially undetermined. The levels of ability demonstrated by the various personnel to whom the test classification trials were given were surprisingly widespread. The amount of indeterminacy (i.e. lack of certainty over individual specimens) among even the best experts was equally surprising. It appears that while the majority of specimens are classifiable in a straight-forward manner, with the experts achieving a high degree of mutual consensus, there are certain individual specimens that are more contentious. By selecting a group of experts who where highly consistent (both internally and externally), we were able to define a threshold of certainty over image class to give us a set of reliably benchmarked data for use in training and assessing the networks. Data that were not reliably classified were discarded in the current study. The morphology of the species in question appears to intergrade, so that some borderline cases make for difficult discrimination. The extent to which this is a problem depends on the abundance of these contentious specimens, and the seriousness of the consequential erroneous evaluation of population sizes. On this first point, the degree of disagreement, amongst those three experts assessed as the most reliable, suggests that the number of disputed individuals amounts to a few percent. The consequences of this level of indeterminacy are not known, but given the small percentage of disputed cases, and the fact that very often the *change* in population levels is what is of interest, it does not seem to be a critical problem. Where such borderline cases might be of importance is in the training phase of a network. A more sophisticated approach would use contentious data to help define the class space. This is discussed further in section 9.4.1.

9.2.4 Assessing network success

When faced with a goal of a working network implementation, the bottom line naturally is 'does it work'. Quantifying what 'work' means in this context is the first task. The best working network would be that which had the highest performance over all that data that one wished to test it on. In a rigourous quality control situation one would intermittently test the network against the current competing classification technique, in this case humans. Given the partially indeterminate nature of the problem, the networks were in reality tested against experts over an edited, determined, data set. It is not clear how one would test a network over a complete set of data, including that small percent of cases which are of uncertain classification, except perhaps by comparing network uncertainty with that of the experts (producing a 'warts and all' network, see section 9.4.1). Unfortunately, in the current experimentation the limited amount of available labelled data was a real constraint on what could be done in the way of training and testing networks. Using a probability performance measure enabled the results of small test sets to be interpreted sensibly, and thus would be beneficial in those domains where labelled data is time-consuming, expensive or somehow difficult to obtain.

Looking for a trend in performance against training set size allowed us to assess the probable performance of a system in conditions where more data are available. The bottom line is that the trials performed showed that the techniques did work, sometimes to high degrees of accuracy, and that this performance appeared to be improving as the systems were trained on larger amounts of data. This result would imply that the collection of a greater number of labelled samples would enable still better performances to be obtained, although the extent to which this will be beneficial is not known. Many networks were obtained with performances above ninety percent correct classification (the best with a performance of 40 correct out of 41 classifications) – in the light of the indeterminacy of the data one cannot expect a much better performance than that to be achieved. This level of success is equitable with that of the most competent experts, and considerably better than that of other taxonomists tested. Although in a narrow domain, we have shown that neural networks are capable of performing a difficult, real world, classification. There is no reason to believe that such performances are unattainable across a wider, and more realistic, range of species, although it is not claimed that the extension of this work is trivially easy. If this can be achieved then not only will it be an impressive demonstration of the neural network technique, but an important and powerful tool will have been added to the field of marine science.

It was discovered that multiple trials were essential not only from a statistical point of view, in order to observe the possible behaviours of systems under different initial conditions, but also in order that one could select a good working net for application. It appears that in some cases in the domain in question, the network enters a local error minima and does not escape. The use of the frequency gradient was a key step in overcoming this problem, and the addition of various forms of noise during network training also helped prevent this. One curious result was that the addition of noise to the training patterns resulted in better classification of the original (i.e. noiseless) training patterns than did training on those very patterns themselves. It may be the case that networks trained on the original (i.e. noiseless) data discover local minima (albeit with a low error) that are less than optimal. The addition of noise to the input patterns deforms the error surface during a training epoch, by effectively redefining the desired input/output mapping. This temporary deformation may allow a network to escape a local minima of the true error surface. In addition to this, the closer the deformation of the patterns follows legitimate distributions of members of the world population, the more the deformed training set can be viewed as having extended the number of training patterns available to the network.

9.2.5 Networks versus Neighbours

In chapter 7 a comparison of the neural network techniques with the nearest neighbour technique was made. Comparable performances were obtained from each. How the two algorithms scale up to more sizable problems remains to be seen. The one obvious strength of the neural network algorithm over that of the Nearest Neighbour technique is its comparative computational simplicity. Once a network is trained, and the weights fixed, then a classification of an input is done by propagating the signals through the system - the computational cost of this is related to the size of the network only. For a nearest neighbour algorithm, the input must be compared with each of the exemplar patterns - the more 'training' patterns the longer the search. In order to assess the rival merits of the two techniques it is necessary to discover firstly the size of network, and secondly the number of exemplars required for a particular problem. All that can be said as regards the current study is that the networks have the computational upper hand. What need also to be borne in mind is that the massive parallelism of the networks enables a truly parallel hardware instantiation, thus increasing their speed many times, while ensuring that larger *wider* networks (i.e. with more hidden nodes but the same number of layers) operate without any extra cost. Although large networks of many nodes may be required for a complicated task, it is the number of *layers* of nodes that is the key issue in speed of the hardware, and this will undoubtedly scale up slower than the number of nearest neighbour exemplars required for large scale classification.

9.3 NETWORK OBSERVATION

9.3.1 Network observation

One common belief is that the operational transparency (i.e. the access we have to the parameters) of a neural network will allow us to observe what is going on. The theory is that once trained to perform some particular task, we can probe or dissect a network to see how it has done it. The need to examine a network in order to understand its workings come from the fact that we generally cannot predict the development of the final device from the starting state. Typically the network is learning autonomously. Unfortunately, the very complexity of the device may mean however that its constituent functions are not as transparent as the lists of weights, connections and activation functions et cetera that they are made up of. Non-linear equations are, theoretically, transparent, but the more complex they become the less able we are to understand them in any significant manner. There is still some observational advantage offered over a wet brain in the fact that we can isolate certain tasks for solution, and study the result of training a network on this task solely. We can also interfere with, or lesion, a network to see if its subsequent behaviour illuminates its workings.

It is occasionally possible to determine some functional properties of a network, such as a particular 'feature detector' node being constructed, all the more impressive a demonstration for the emergence of the property from seeming random parameters. In the current study, it was possible to determine the presence of simple 'feature detectors', or at least to see what portion of the input data was most pertinent to the classifications performed. Repeated learning trials showed that the networks were converging on similar solutions from random initial states, even when having different architectures. Inevitably though, simplification for the purposes of investigation causes problems itself. We now know that some simple operational devices are in principle logically complete (as is a Turing machine), but that does not mean that one huge feed-forward network will be able *in practice* to learn any degree of sophisticated behaviour we care to stipulate. The brain, certainly, is not a homogeneous network - there is much structure imposed on it before any learning takes place at all. The common over-simplification in the neural network field to the effect that all learning is in the adaptation of connections is exactly that, an over-simplification. Yet significant adaptive mechanisms can be constructed in spite of this simplification, and their consequences for theories of learning investigated. Back-propagation, as an algorithm for training a network, is not a credible biological model. Rather, it should be considered a credible solution to the credit assignment problem, which enables us to develop suitably structured feed-forward networks for performing particular mappings. Those interested in applications need have no qualms about the employment of such ad hoc techniques in the pursuit of a particular goal. For those who seek to elucidate *brain* mechanisms, the construction of particular mappings by back-propagation methods may be regarded as a tool, used without implying the existence of the various mechanisms used by the algorithm, any more than the use of computer simulations imply anything like a central processing unit in the brain.

9.4 IMPROVING PERFORMANCE

9.4.1 Selecting training data

An improved performance can undoubtedly be obtained with some consideration of the nature of the training patterns themselves. The selection of effective training patterns has been discussed, but in reality one is likely to wish to use all available training data, but somehow allow for different degrees of attention to be paid to different patterns. Scheduling the network so that it learns the central distribution first, by training on the most exemplary patterns, and then allowing it to add additional classification regions incorporating the less typical patterns is one strategy. There is the question of how to treat those patterns that cause confusion in the experts. One method is to train a network to emulate the expert 'warts and all', so that those patterns that produce unclear classification in the human do so in the network (perhaps by presenting diffident target activations to the network on presentation of these patterns). As mentioned in section 8.2.2, it may be beneficial to use contentious patterns in the training set, as in Lenet (1977). Alternatively, one might decide that such patterns are not worth attempting to learn given that there appears to be no credible classification of them anyway.

9.4.2 Arbitration

The use of arbitration in forming a final classification stemmed from a belief that the fusion of multiple solutions would provide stronger evidence for a species than reliance on one solution alone. The marginal improvement in the performances obtained were encouraging, although on reflection it seems that in the experiments performed the similarity of the solutions of the trained networks, although illustrating the robustness of the training algorithm, acts against the idea of arbitrating multiple (different) solutions. The more similar two solutions are the less they have to teach each other. A better technique might be to find truly alternative solutions, by use of different network architectures perhaps, or different cost functions (e.g. Hampshire and Waibel 1990), or even different output representations. Baxt (1992) has recently successfully extended this idea by training multiple networks to different input/output mappings, one mapping to maximise network specificity, another to maximise sensitivity. The extent to which improvement over generalisation given a training set can be obtained by dividing the data into separate problems has not yet be fully explored.

9.4.3 Default classes

The diagrams such as those in Figure 48 show that the networks are largely discovering one class, and assigning everything else to another. In the majority of the cases observed it was the case that *C. arcticum* was the 'discovered' class, and *longipes* thus classified by default (i.e. recognised as 'not *arcticum*'). This is illustrated by the compact distribution of the class of *arcticum* species in the unit cube, in some cases collecting in just one corner of the cube (implying only one possible set of hidden activations for that species). This is of course a perfectly acceptable strategy for a network in a binary classification task. One might wish the network to form a compact representation for both species, in which case a third class of 'noise', i.e. neither one nor other of the species, might be recognised. Quite how one would wish such a class to be classified is open to question, the addition of an extra output node rendering the situation no longer binary anyway. What exactly one would wish to include in the training set as noise is also difficult to answer. One possible solution would be to generate random vectors that are at least some threshold distance from any actual vector of a legitimate species.

The construction of default classes has been observed in non-binary tasks also. Where one has an n-class problem, with the n outputs assigned a class apiece, it is likely that a noisy output (i.e. not a member of one of the n legitimate classes) will be construed as belonging to one of those classes (of course where a winner-takes-all or nearest neighbour mechanism is used to determine class membership this is true by definition). The sigmoidal function tends to produce outputs saturating towards a value of 0 or 1 as one moves away from the training data, unfortunately these are often the very values chosen to represent class membership. It may be beneficial to construe outputs as certainty values and to reject any classifications below some threshold, or to code some non-unary representation as representing a null class (such as all zeros). In Figure 48 we can see instances where the default class (*longipes*) is not uniformly distributed about the hidden unit activation space, thus showing that this species is capable of being learnt in its own right, and not just as a default class.

9.4.4 Building large classifiers

As stated in section 9.2.2, a significant part of the appeal of neural networks for many people is that suitable functions are somehow 'discovered' by the network algorithm itself, and is thus naturally suited to the data in question. However, since in mathematical terms the network is typically (for instance) just discovering a suitable hyperplane, it is very possible that the solution discovered is less than optimum. The initial constraints of network size, architecture and connectivity are enforcing as rigourous limits on the solution as the choice of some mathematical discriminant function, or algorithm. Judd (1992) points out that since networks have to learn their behaviours, it is wise to be sure the cost of training them does not exceed the benefit of using them. Where topological constraints on learning may become more visible is when one attempts to enlarge on the current task, and design a system for the classification of a greater number of species. Using a feed-forward neural network with a few hidden nodes to discriminate between two species is one thing. Extending such a system to cope with (perhaps) an order of magnitude more species is another. One cannot expect a network of a particular size to learn an indefinite amount of material. Extending a network to more species will require more output nodes, or at least a more complex output representation, and more hidden nodes. As the network grows training times will increase dramatically. Consider that one might realistically wish to consider yet another order of magnitude more species in time, and it becomes apparent that simply adding more output nodes (and perhaps a few more hidden nodes) is not a very plausible approach.

One possible solution to a large classification task is by breaking it up into many smaller problems, each smaller than the whole, and relatively trivial for a network. The desired output for a pattern would be a collaborative effort of several networks. One way in which this might be done would be by a hierarchy of classifying networks, each trained on a simple (perhaps binary) discrimination task. These small tasks would be more easily learnt by each module of the system, than would the whole division of the input space. A final classification would be the result of a sequence of increasingly fine discriminations. The division of the task into appropriate sub–tasks might be achieved by manual division of the problem class space using some cluster analysis technique, or (more appealingly) by some automatic, data–driven, technique. One problem in knowledge representation is determining the resolution of the representations. If a system is asked to learn to distinguish between two patterns, the resultant discriminator may serve well on those two items alone, but may or may not extend well to consideration of another item. A hierarchy of classifiers should be able to be extended by the addition of extra modules without discarding the learnt discriminations that have gone before. This would be of benefit in designing a system that is able to cope with the addition of extra classes, surely an important consideration in a field that has a huge number of classes that might be usefully discriminated.

9.5 CONCLUSION

Neural networks are a new and opaque technology in pattern recognition. New in that they offer a paradigm of pattern recognition and concept formation radically different from previous ideas in artificial intelligence. Opaque in that the extent to which their recent impact on the field will be maintained as their novelty fades is unknown. Certainly there is an intuitive appeal in the idea of a *learning* system, especially one that starts with a minimum of assumptions about a task, and appears to adapt to an arbitrary environment in any way we desire. These ideas are of course an extravagant portrait of reality. No network yet designed can do this – the design of a basic network structure enabling it to learn specific tasks is a current preoccupation of researchers in the field. This thesis has attempted to investigate the application of networks to a particular real world problem, and had some success. It appears that the technology of neural networks is suited to the domain of marine plankton identification, and we remain confident that further progress will be made in this important application. Of equal use is the knowledge gained in attempting to apply networks to our real world problem. Extension of these ideas to wider problems will inevitably force us to redevelop our ideas about what is, and what is not, feasible for networks. Contemporary systems are the lower bound on this, the brain points the way to the upper. Where we finally come to lie on a developmental maxima is impossible to say.

Appendix I - Pop-11 code.

```
/*

                POP-11 Back-propagation simulation code.

          Copyright (c) Rob Simpson,  Plymouth Polytechnic, 1988.

*/

vars net size_list weights change;          ;;;
vars bias bias_change;                      ;;; Initialise network variables
vars Input Target learn momentum;           ;;;

vars net_set_up connect_all connect rand_fn; ;;; Procedure names
vars propagate correct activation RMS;      ;;;

define net_set_up(list);
/*
   Takes a list of any length, [N1 N2 ... Nn] and sets up network with
   'n' layers, layer 1 having N1 units, layer 2 having N" units, and so on.
   The arrays for unit activation, weights and biases are initialised.
   The Pth node in the Qth layer is stored in net(Q,P), and consists
   of a three element list, [activation INlist OUTlist], where activation
   is the units current response, INlist the list of nodes feeding to the
   node, and OUTlist a list of nodes it feeds out to.
*/
   vars layers x y most=-1;
   list -> size_list;
   length(size_list) -> layers;
   for x from 1 to layers do
      if size_list(x) > most then size_list(x) -> most; ;;; Biggest layer
      endif;
   endfor;

   newarray([1 ^layers 1 ^most]) -> net;
   newarray(boundslist(net)) -> bias;
   newarray(boundslist(net),0) -> bias_change;
   newarray(boundslist(net)<>boundslist(net)) -> weights;
   newarray(boundslist(net)<>boundslist(net),0) -> change;

   fast_for x from 1 to layers do
      fast_for y from 1 to most do
         newarray([1 3]) -> net(x,y);
```

```
        0 -> net(x,y)(1);            ;;; Activation of node
        newarray([1 ^(2*most*layers)],0) -> net(x,y)(2); ;;; IN list
        newarray([1 ^(2*most*layers)],0) -> net(x,y)(3); ;;; OUT list
        rand_fn() -> bias(x,y);
      endfast_for;
    endfast_for;

enddefine;

define connect_all();
/*
    Either takes two numbers, P and Q, and connects every node in layer P
    to every node in layer Q (feeding from P to Q), or connects every layer
    in the network to the layer immediately above.
*/
    vars x y m n;
    if stacklength() = 2 then
        -> n -> m;
      for x from 1 to size_list(m) do;
        for y from 1 to size_list(n) do;  ;;; Link specified layers
          connect(m,x,n,y);
        endfor;
      endfor;
    else
      for x from 1 to length(size_list)-1 do ;;; Link all neighbouring layers
        connect_all(x,x+1);
      endfor;
    endif;

enddefine;

define connect(a,b,p,q);
/*
    Takes four numbers, and connects node(a,b) to node(p,q), updating
    (a,b)'s OUTlist, and (p,q)'s INlist.
*/
    vars x y;
    1 -> x;
    repeat
      if net(a,b)(3)(x) = 0 then
        quitloop; endif;           ;;; Checked all connections
      if net(a,b)(3)(x) = p
      and net(a,b)(3)(x+1) = q then false -> x; ;;; Is there an existing connection?
    quitloop; endif;
    2 + x -> x;
  endrepeat;

  if x /= false then
    p -> net(a,b)(3)(x);
    q -> net(a,b)(3)(x+1);          ;;; Insert new TO connection

    for x from 1 by 2 to length(net(p,q)(2)) do
      if net(p,q)(2)(x) = 0 then
        a -> net(p,q)(2)(x);
        b -> net(p,q)(2)(x+1);   ;;; Insert new FROM connection
        quitloop; endif;
    endfor;
    rand_fn() -> weights(a,b,p,q);    ;;; Initialise with random weight
  endif;

enddefine;

define propagate();
/*
  Takes input from the 'Input' variable, and propagates the activations
  through the network to the last layer.
*/
  vars layer node x a b act total_act weight;

  for x from 1 to size_list(1) do
    Input(x) -> net(1,x)(1);        ;;; Put input values into lowest layer
  endfor;

  for layer from 2 to length(size_list) do      ;;;  2nd layer upwards
    for node from 1 to size_list(layer) do    ;;;  Every node in layer

      0 -> total_act; 1 -> x;
      repeat
        net(layer,node)(2)(x) -> a;
        if a = 0 then quitloop; endif;
        net(layer,node)(2)(x+1) -> b;

        net(a,b)(1) -> act;           ;;; Activation of lower node
        weights(a,b,layer,node) -> weight;
        (act*weight) + total_act -> total_act;
        x + 2 -> x;
      endrepeat;
```

```
            activation(total_act + bias(layer,node)) -> net(layer,node)(1);

         endfor;
      endfor;

enddefine;

define rand_fn() -> out;
/*
   Gives a random value between -0.1 and +0.1, used to initialise weights.
*/
   ((random(1.0)*2)-1)/10 -> out;
enddefine;

define activation(input) -> output;
/*
   Defines the sigmoid activation function.
*/
   if abs(input) > 50 then
      if input > 50 then 1
      else 0;
      endif;
         -> output;
      return;
   endif;
   1/(1 + exp(-input)) -> output;
enddefine;

define correct();
/*
   Back-propagates the error between the output activations and 'Target',
   in view of the various node activations, thus updating the biases
   and weights.
*/
   vars error layer node act sum p q x;
   newarray(boundslist(net)) -> error;

   for layer from length(size_list) by -1 to 1 do ;;; For every layer,

      if layer = length(size_list) then            ;;; For output layer
         fast_for node from 1 to size_list(layer) do
            net(layer,node)(1) -> act;
            (Target(node) - act)*act*(1-act) -> error(layer,node);
            bias(layer,node) +
      (learn*error(layer,node)*1)
      +(momentum*bias_change(layer,node))
         -> bias(layer,node);
      (learn*error(layer,node)*1)
      +(momentum*bias_change(layer,node))
         -> bias_change(layer,node);
   endfor;
else                              ;;; For hidden layers,
   for node from 1 to size_list(layer) do
      net(layer,node)(1) -> act; 0 -> sum; 1 -> x;
      repeat
         net(layer,node)(3)(x) -> p;
         if p = 0 then quitloop; endif;  ;;; Done every connection
         net(layer,node)(3)(x+1) -> q;
         (error(p,q) * weights(layer,node,p,q)) + sum -> sum;
         x+2 -> x;
      endrepeat;
      sum * act * (1-act) -> error(layer,node); ;;; Error of a node

      if layer /= 1 then
         bias(layer,node) + (learn*error(layer,node)*1)
         +(momentum*bias_change(layer,node))
            -> bias(layer,node);
         (learn*error(layer,node)*1)
         +(momentum*bias_change(layer,node))
            -> bias_change(layer,node);
      endif;
      1 -> x;
      repeat
         net(layer,node)(3)(x) -> p;
         if p = 0 then quitloop; endif;
         net(layer,node)(3)(x+1) -> q;

         weights(layer,node,p,q) +
         (learn*error(p,q)*act)+(momentum*change(layer,node,p,q))
            -> weights(layer,node,p,q);
         (learn*error(p,q)*act)
         +(momentum*change(layer,node,p,q))
            -> change(layer,node,p,q);
         x+2 -> x;
      endrepeat;
   endfor;
```

```
        endif;
    endfor;

enddefine;

define RMS() -> error;
/*
    Calculates RMS error over patterns. If no arguments given calculates
    error over all patterns, if two numbers P and Q given calculates error
    over patterns P to Q inclusive, if four numbers A, B, P and Q given
    calculates error over patterns A to B inclusive, and P to Q inclusive.
*/
    vars size a b x y outs n m q p top act num;
    false -> n;
    false -> p;
    if stacklength() /= 0 then -> n -> m; endif;
    if stacklength() = 2 then -> q -> p; endif;
    0 -> error;
    boundslist(net)(2) -> top;
    hd(rev(size_list)) -> outs;

    if n = false then
        if isarray(input) then
            boundslist(input) -> size;
            size(1) -> a; size(2) -> b;
        else
            1 -> a; length(input) -> b;
        endif;
    else n -> b; m -> a;
    endif;

    for x from a to b do        ;;; For each pattern from a to b,
        input(x) -> Input; target(x) -> Target;
        propagate();
        for y from 1 to outs do
            net(top,y)(1) -> act;
            ((act - target(x)(y)) ** 2) + error -> error;
        endfor;
    endfor;
    b-a+1 -> num;

    if p /=false then
        for x from p to q do        ;;; For each pattern from p to q,
            input(x) -> Input; target(x) -> Target;
            propagate();
            for y from 1 to outs do
                net(top,y)(1) -> act;
                ((act - target(x)(y)) ** 2) + error -> error;
            endfor;
        endfor;
        q-p+1 + num -> num;
    endif;

    sqrt(error / (outs*num)) -> error;

enddefine;
```

Appendix II – Plankton specimen images and task instruction sheet

The following pages give the instruction sheet used for the *Ceratium* classification task, and an example set of test sheets.

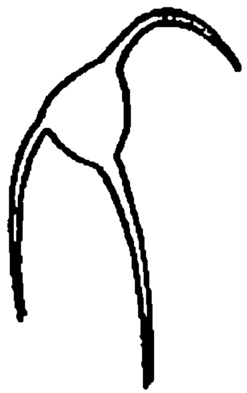
Ceratium Classification Task

This trial is part of an investigation into the classification of Plankton, namely *Ceratium* species. With these instructions should be nineteen sheets of paper, each with a dozen specimens on. We ask you to categorize, or classify, each specimen as a particular species. Given that most of the examples will be *Ceratium arcticum, horridum, triposi* or *longipes*, the initial letter 'A' (or H, T or L) will suffice for identifying a specimen. Using these abbreviations, please indicate which species you think each picture is of, putting your choice in the relevant box, ie. for the left-hand uppermost picture, use the left-hand uppermost box, and so on.

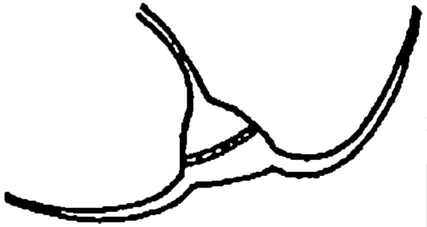
Along with each classification, we are looking for a 'confidence rating', indicating how sure you are as to your classification. A specimen which is very clearly *arcticum* therefore, perhaps a very typical one, might have a confidence rating of 100 percent, and could be classified as 'A 100'. If this is the case, a simple 'A' will suffice. A specimen which might be *longipes* but is more likely to be *horridum* could be classified as '40 L / 60 H', for instance. If you cannot decide which of two categories an individual falls into then a rating of '50 L / 50 T', (for *longipes* or *arcticum*) is appropriate, otherwise the highest rating will be taken as your final classification. There is an example sheet to make this clearer. Note that the ratings on this sheet bear no deliberate relation to the pictures. Thank you for your co-operation!

PML
Feb.
1991

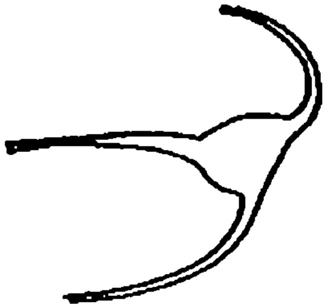
Page
12.1



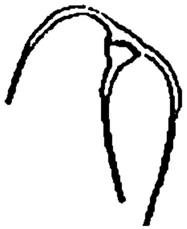
1



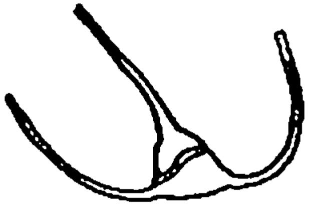
2



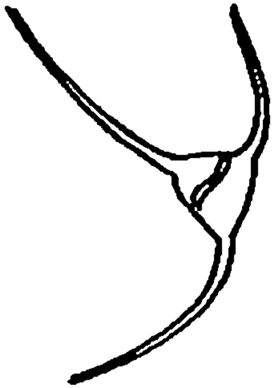
3



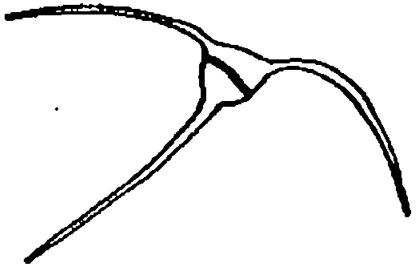
4



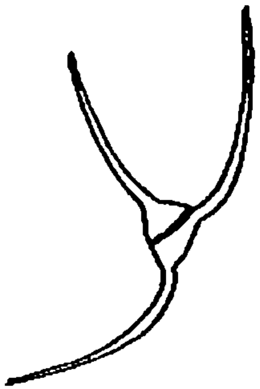
5



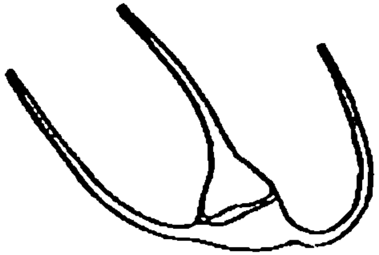
6



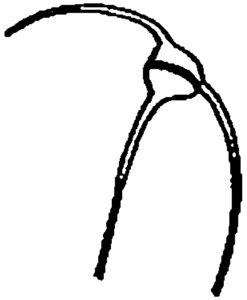
7



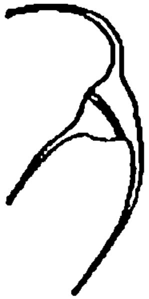
8



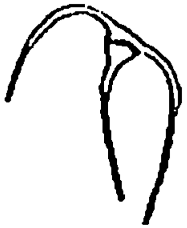
9



10



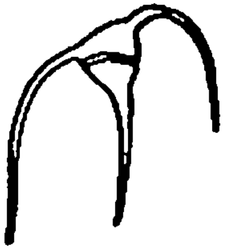
11

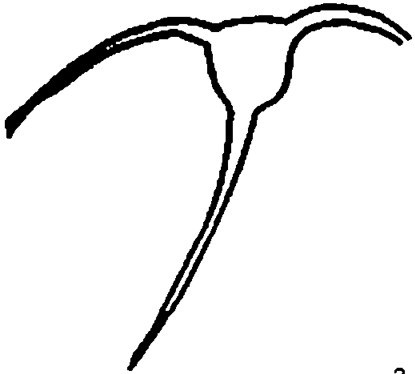


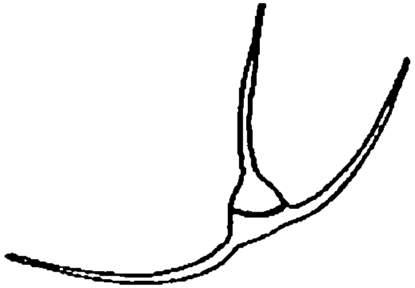
12

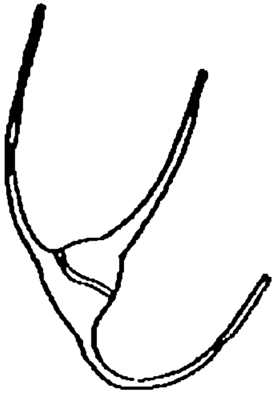
PML
Feb.
1991

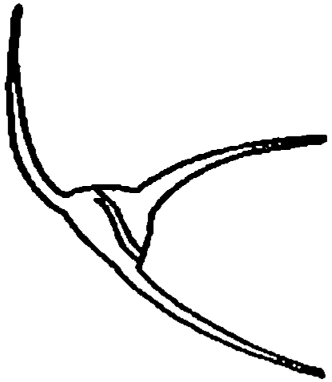
Page
12.2

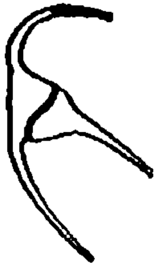

1

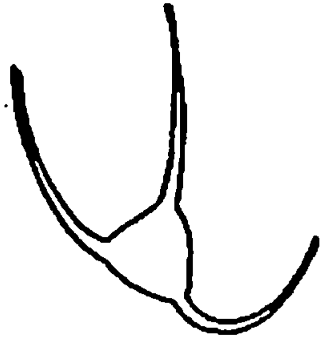

2

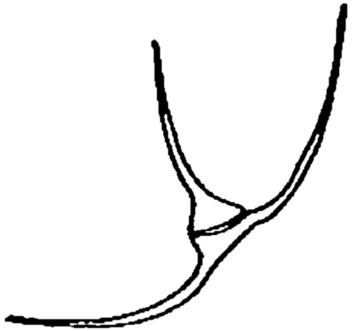

3

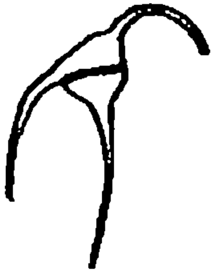

4

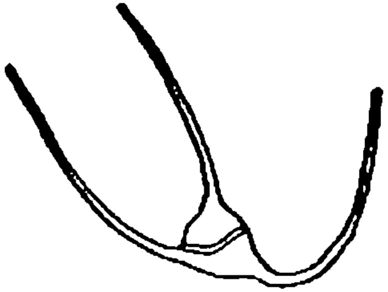

5

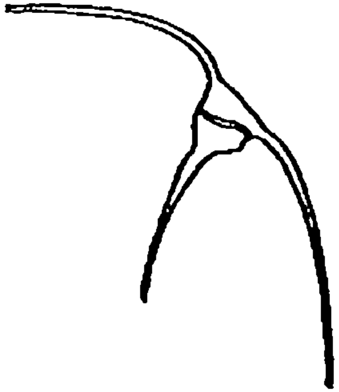

6

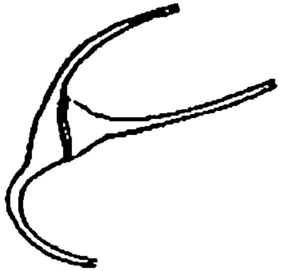

7

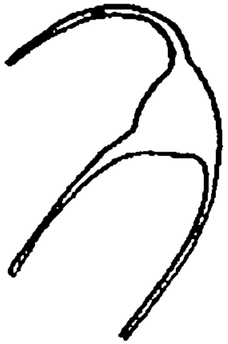

8


9

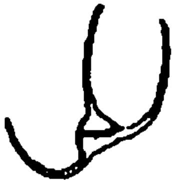

10


11

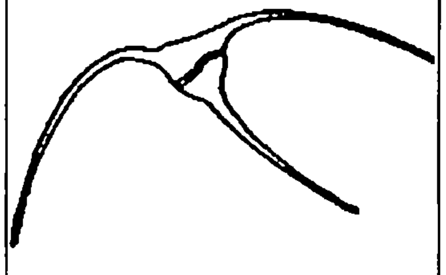

12



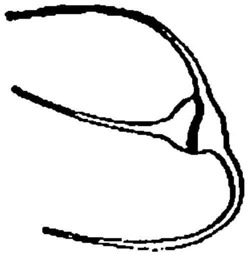
1



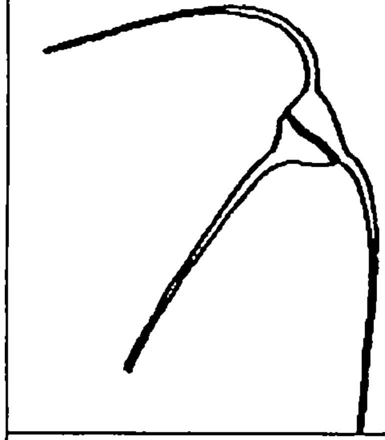
2



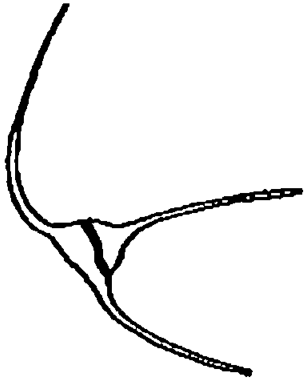
3



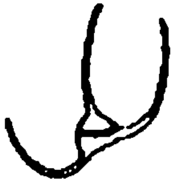
4



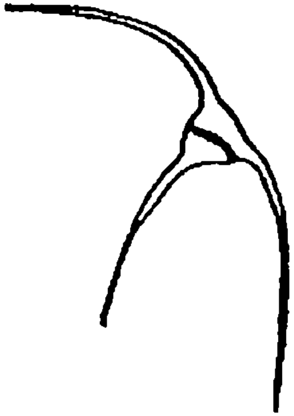
5



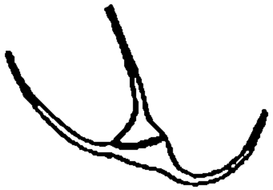
6



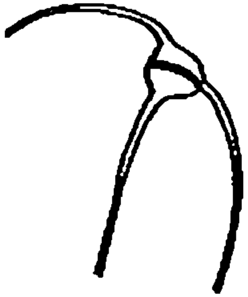
7



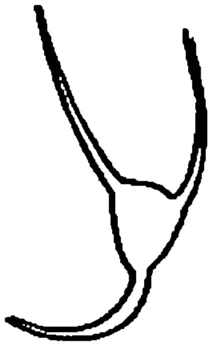
8



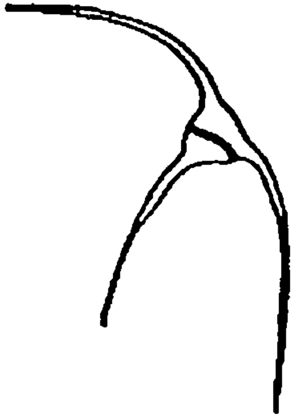
9



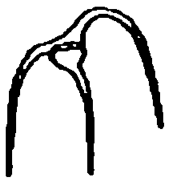
10



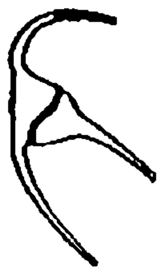
11



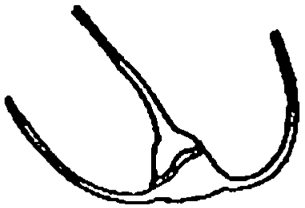
12



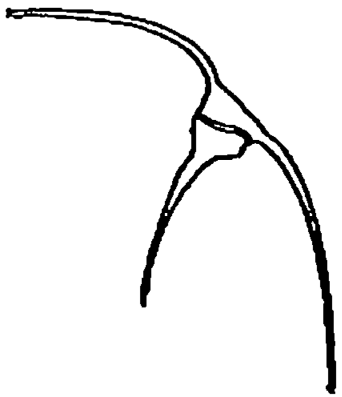
1



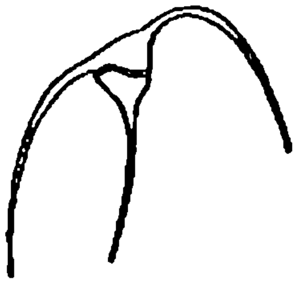
2



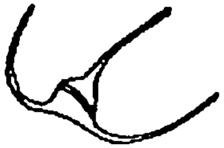
3



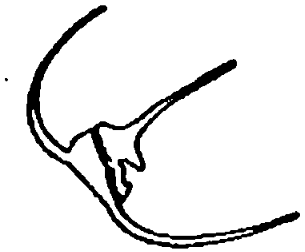
4



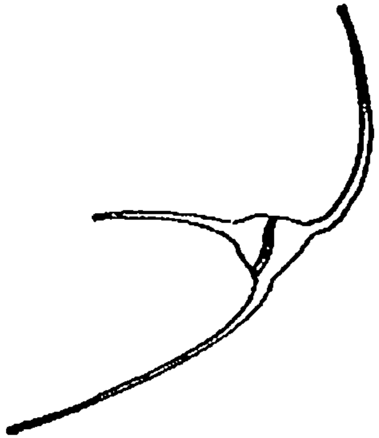
5



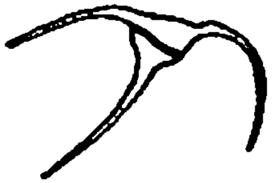
6



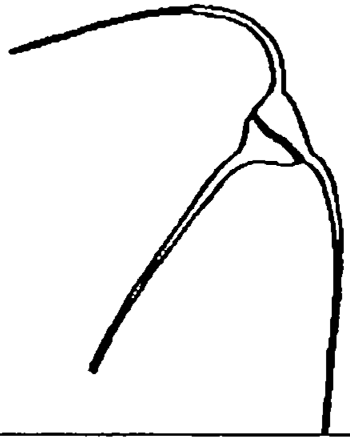
7



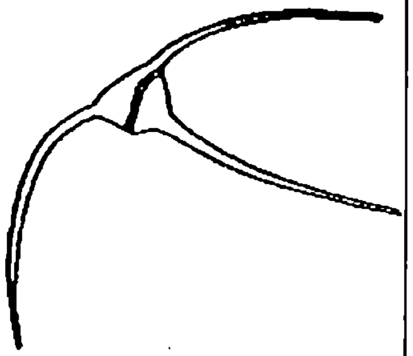
8



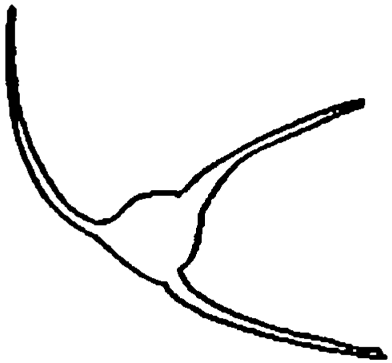
9



10



11

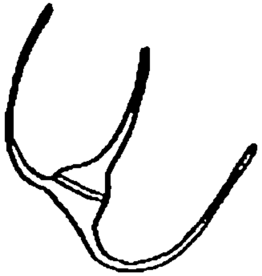


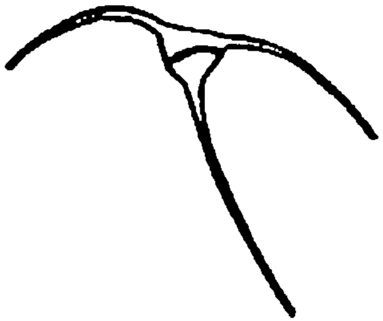
12

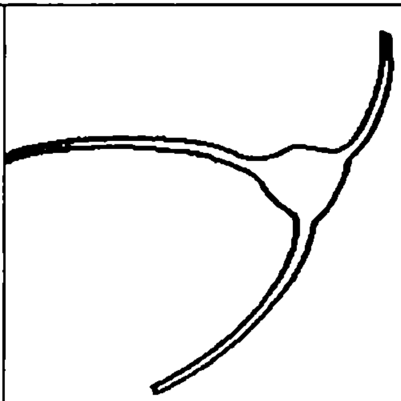
Ceratium classification trials, (c) R. Simpson 1991

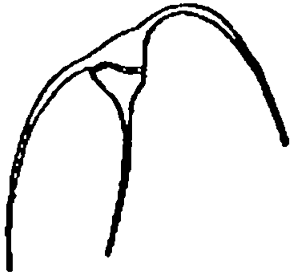
PML
Feb.
1991

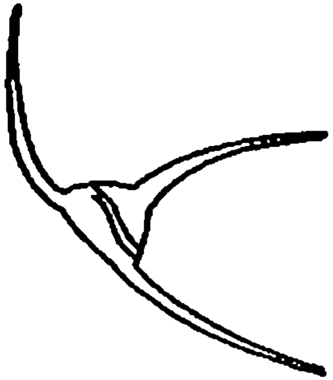
Page
12.5

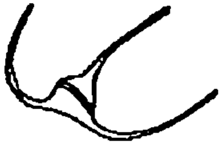

1

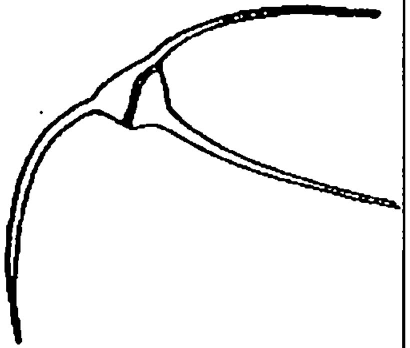

2

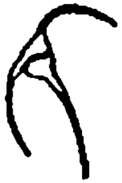

3

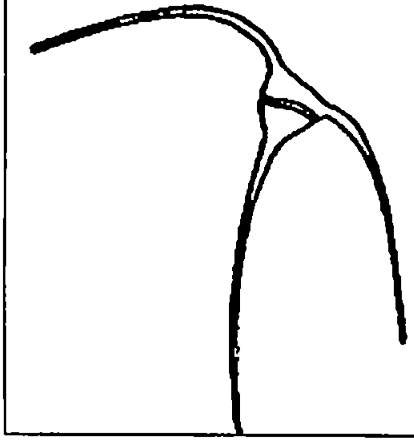

4

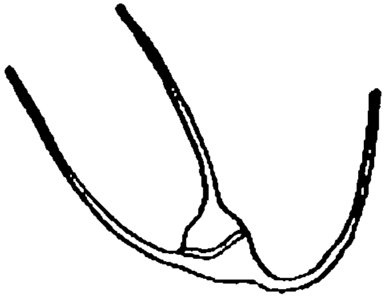

5

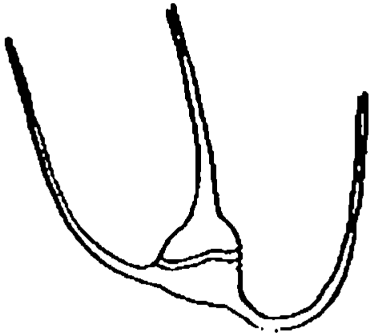

6

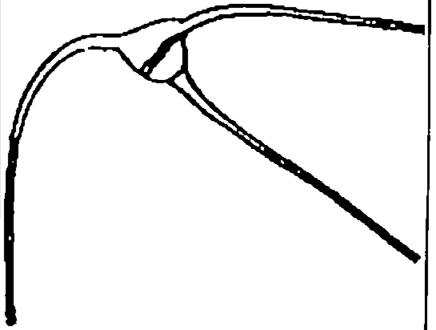

7


8


9

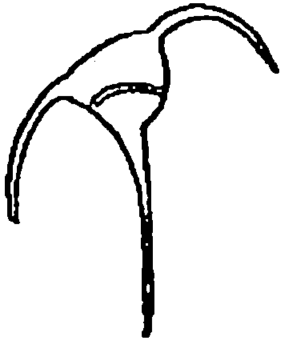

10

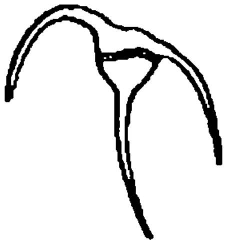

11

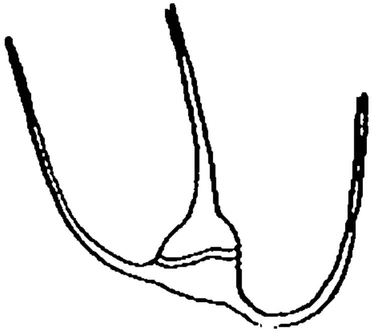

12

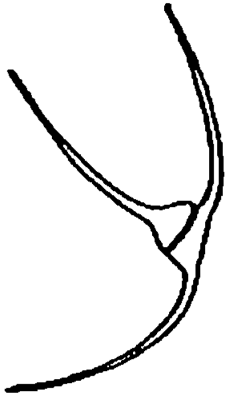
PML
Feb.
1991

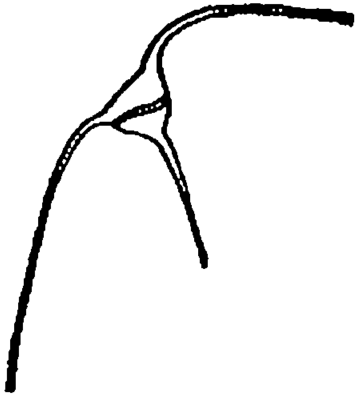
Page
12.6

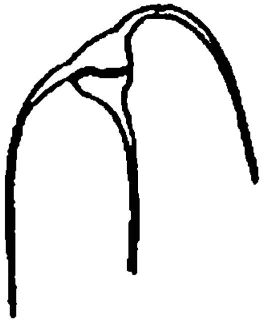

1

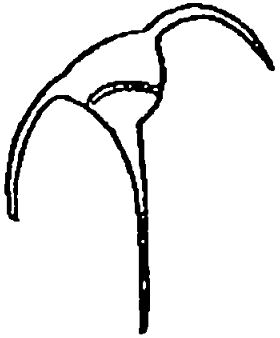

2

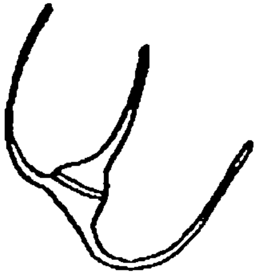

3

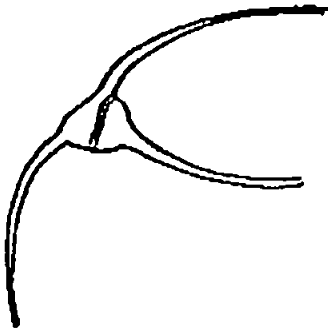

4

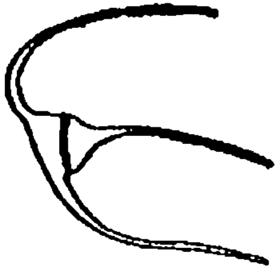

5

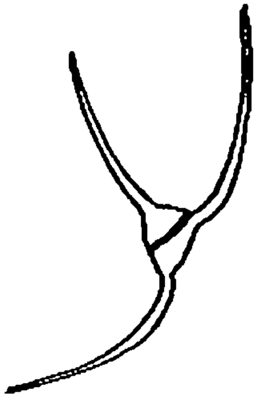

6

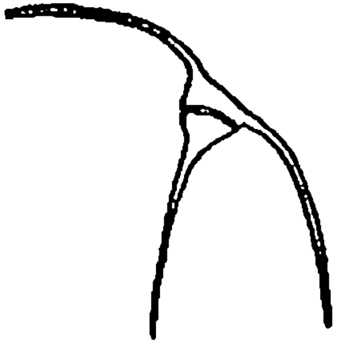

7


8


9

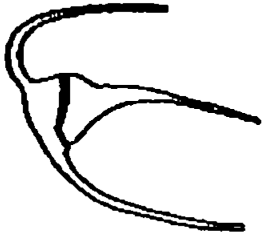

10

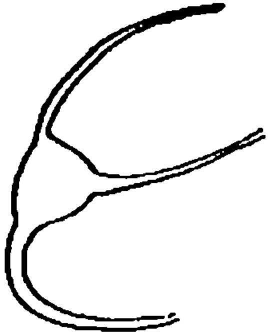

11

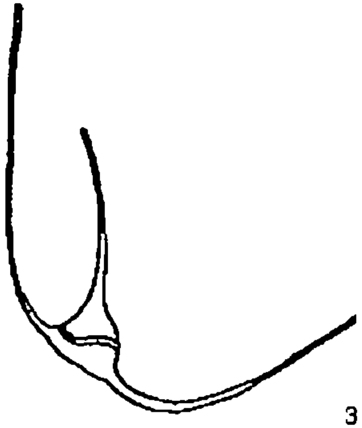

12

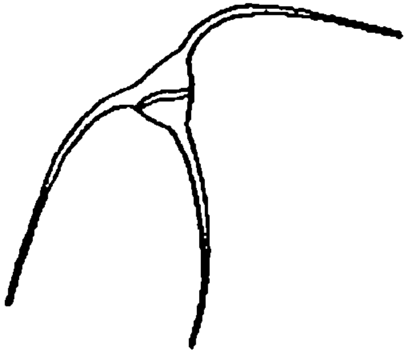
PML
Feb.
1991

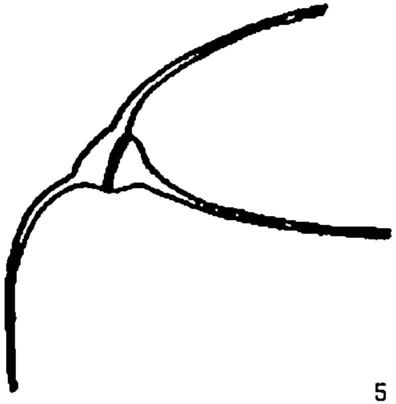
Page
12.7

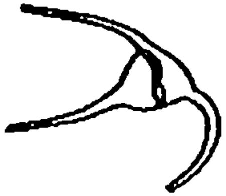

1

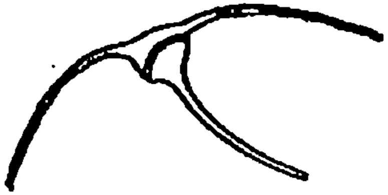

2

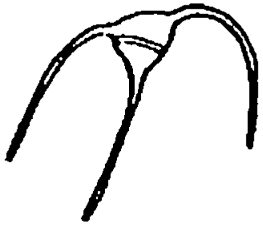

3

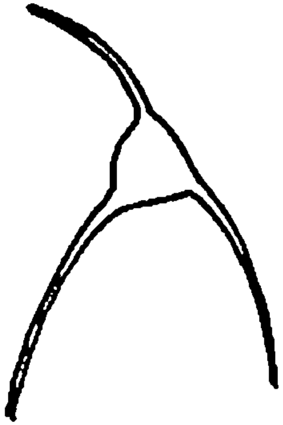

4

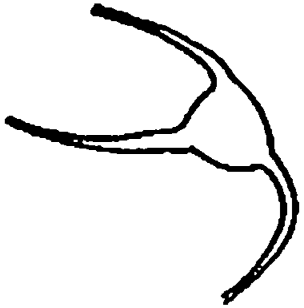

5

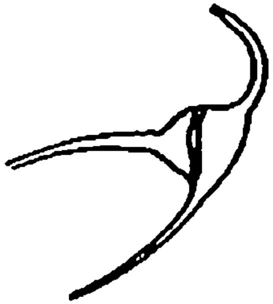

6

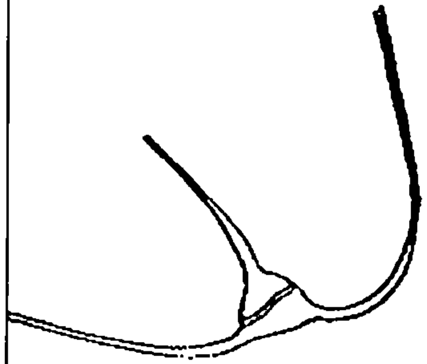

7

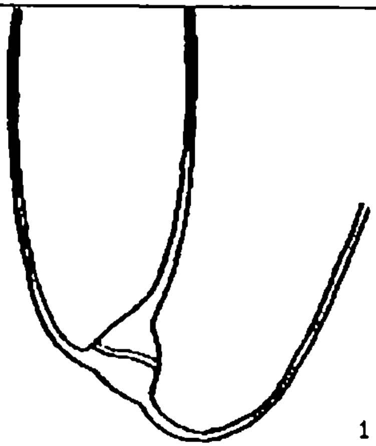

8


9

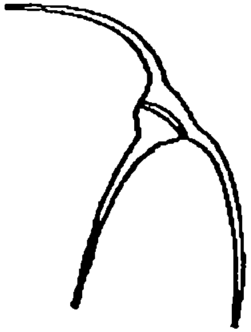

10


11

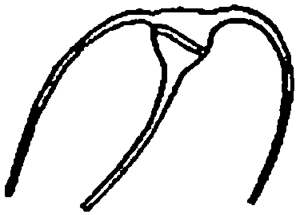

12



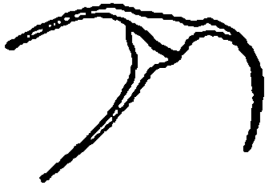
1



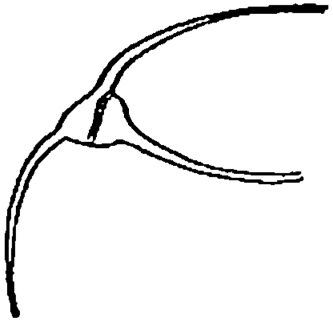
2



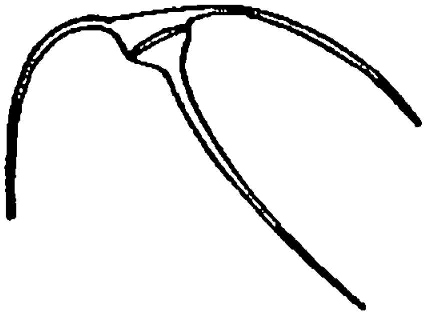
3



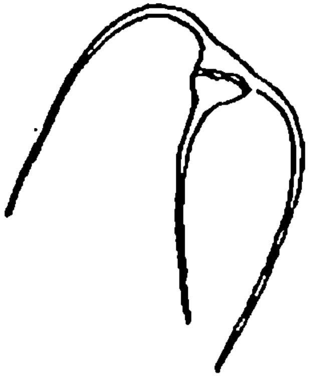
4



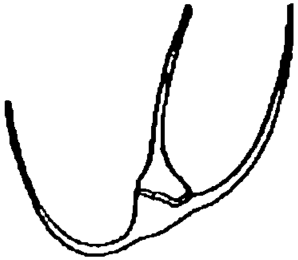
5



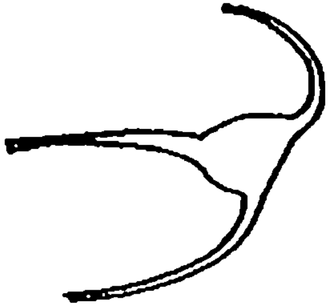
6



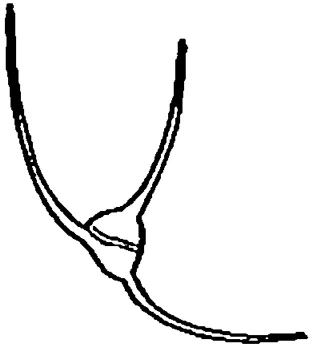
7



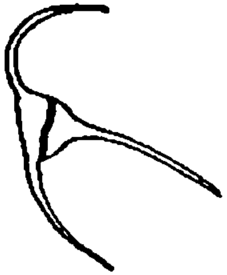
8



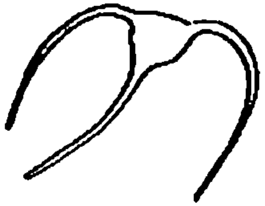
9



10



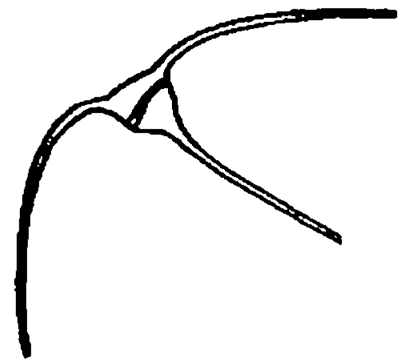
11

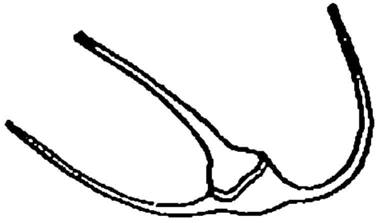


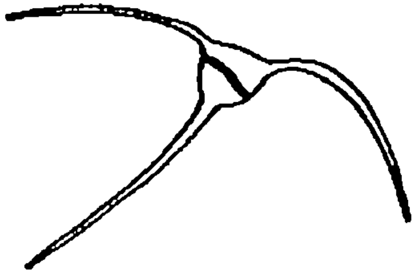
12

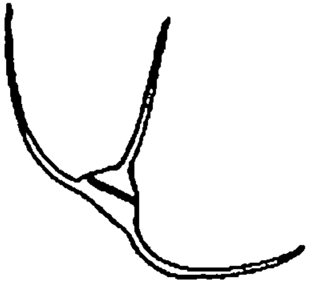
PML
Feb.
1991

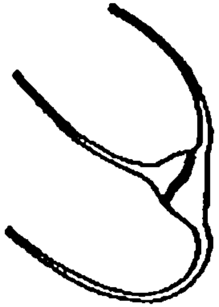
Page
12.9

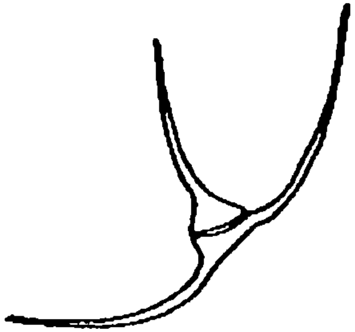

1

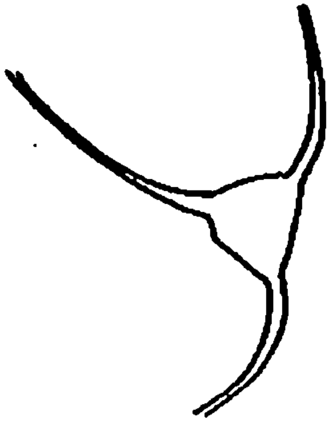

2

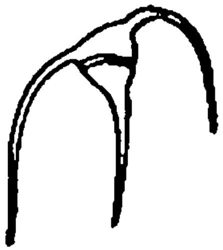

3

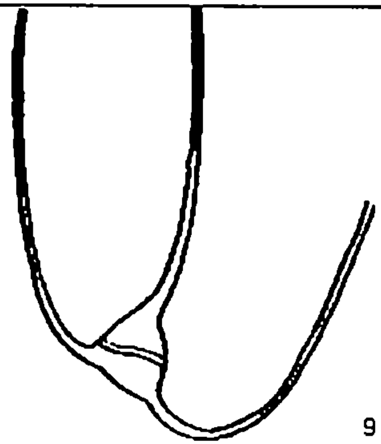

4

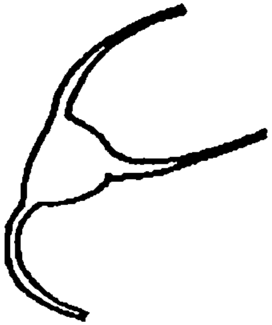

5

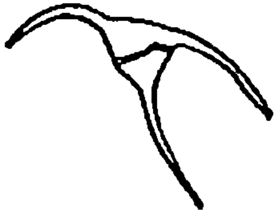

6

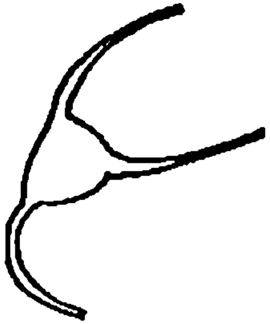

7

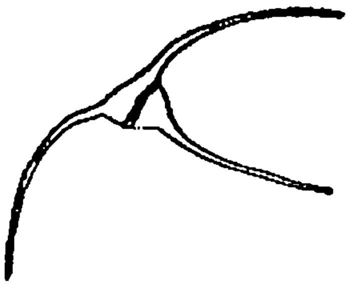

8


9

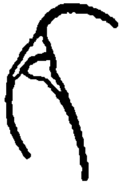

10


11

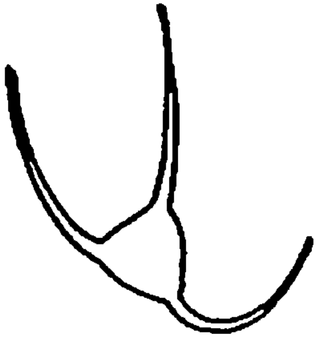

12



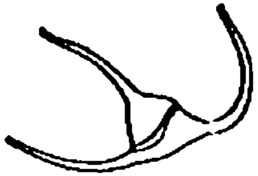
1



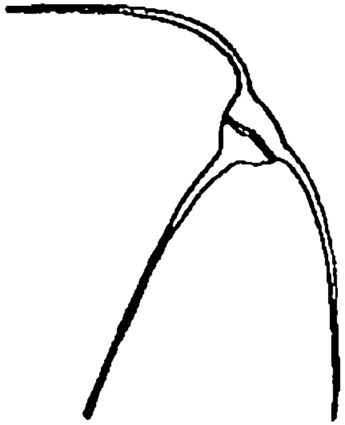
2



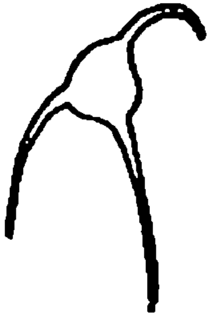
3



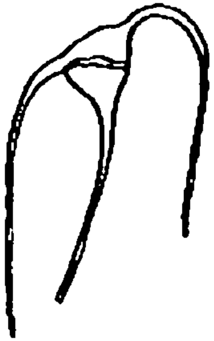
4



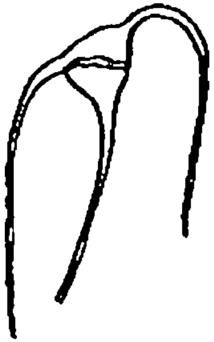
5



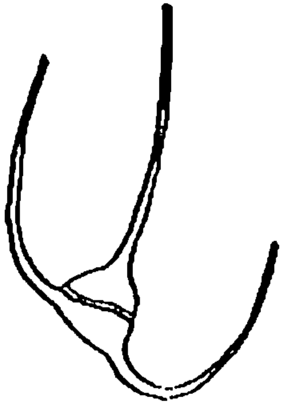
6



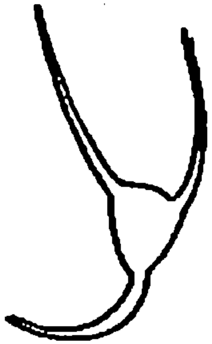
7



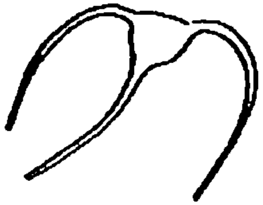
8



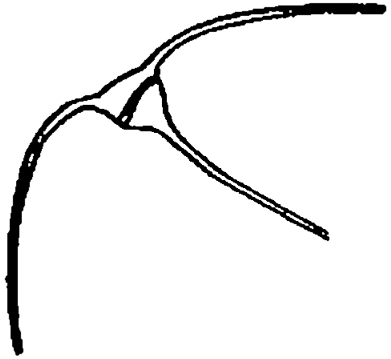
9



10



11

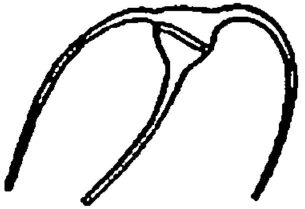


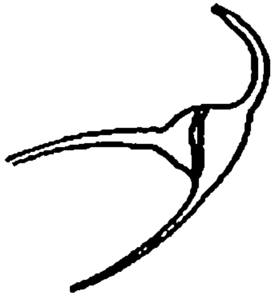
12

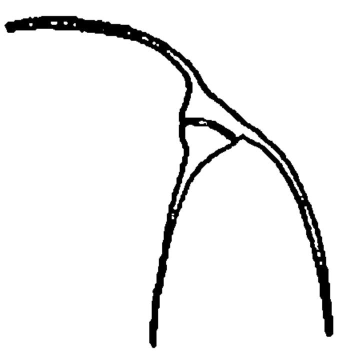
Ceratium classification trials, (c) R. Simpson 1991

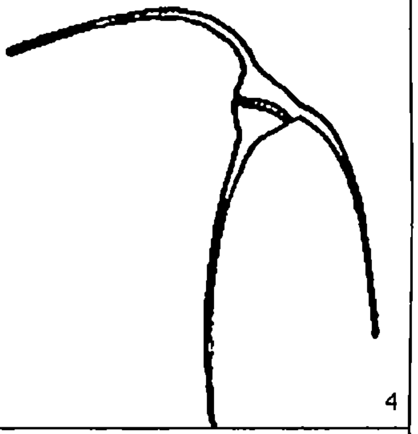
PML
Feb.
1991

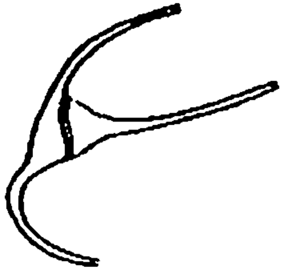
Page
12.11

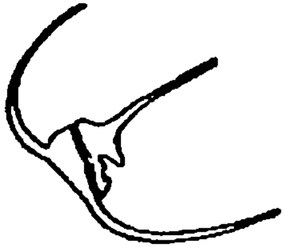

1

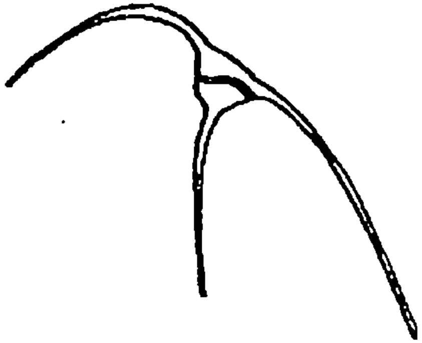

2

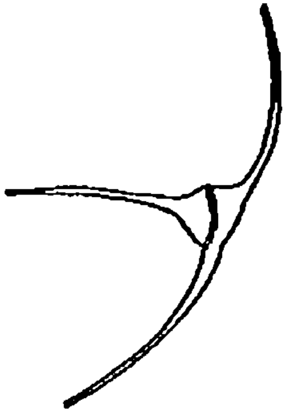

3

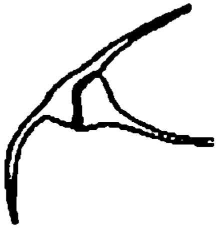

4

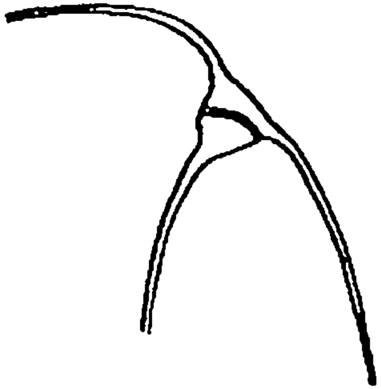

5

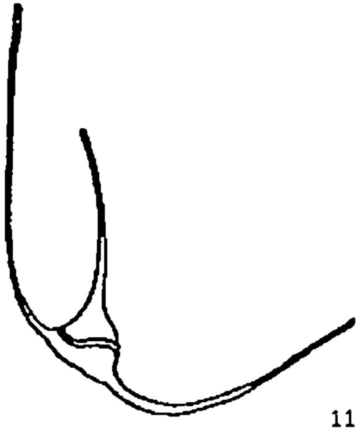

6

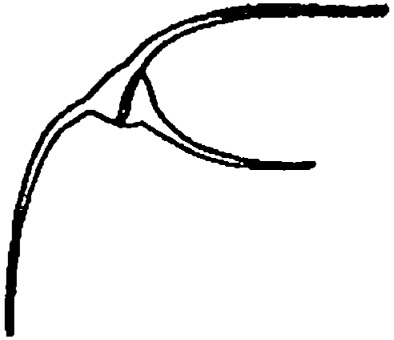

7


8


9


10

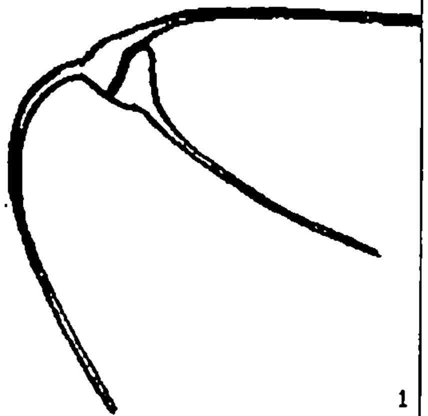

11

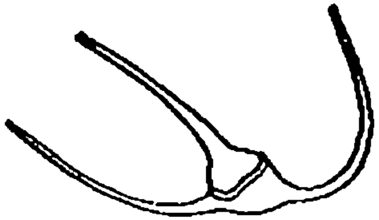

12

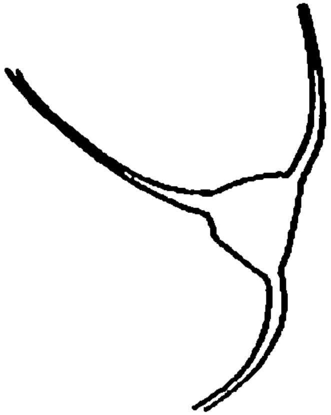
Ceratium classification trials, (c) R. Simpson 1991

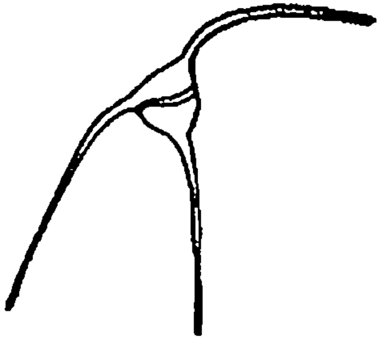
PML
Feb.
1991

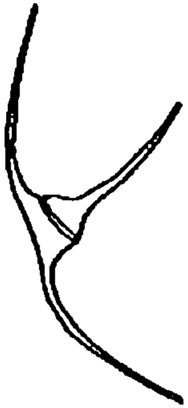
Page
12.12

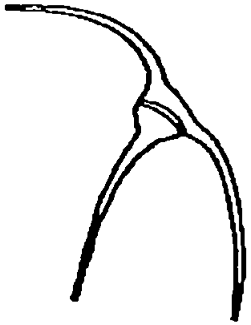

1

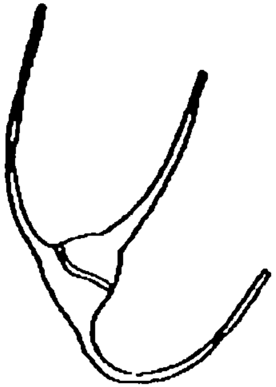

2

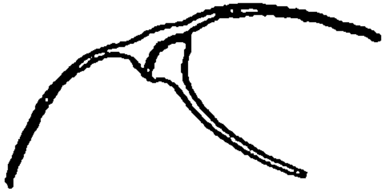

3

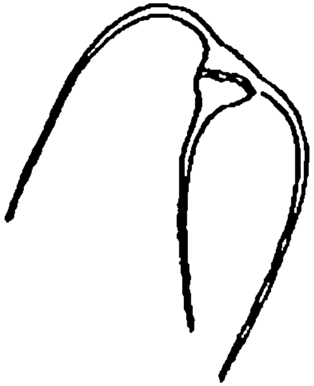

4

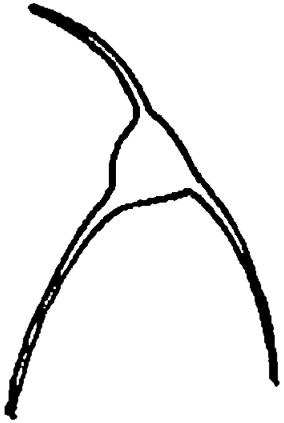

5

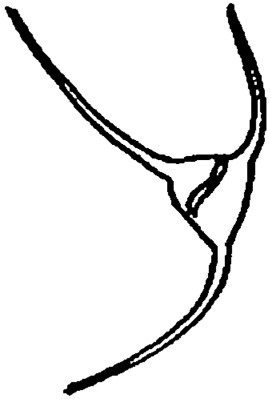

6

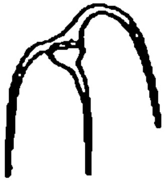

7

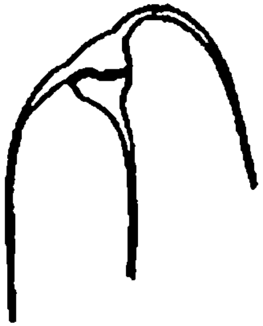

8


9

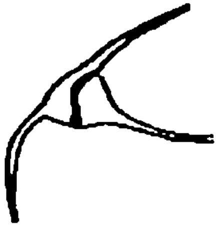

10


11

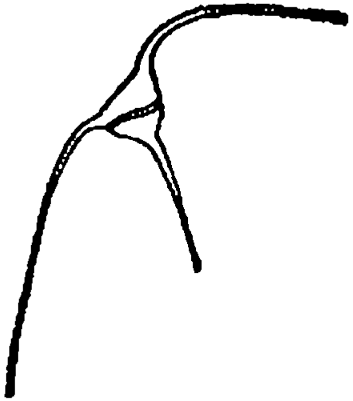

12



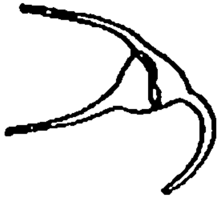
1



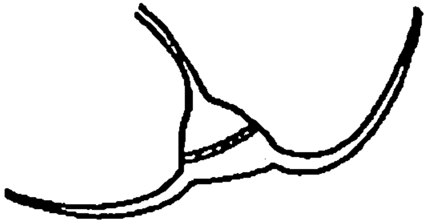
2



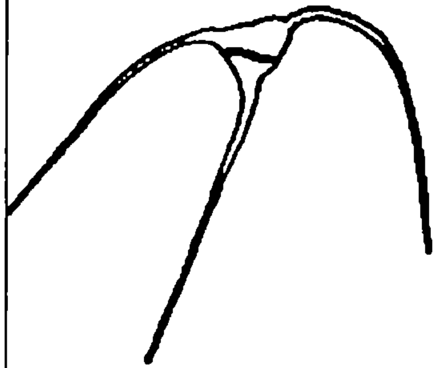
3



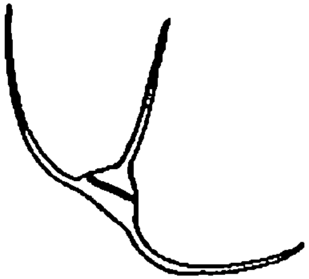
4



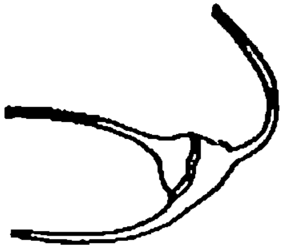
5



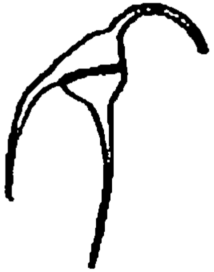
6



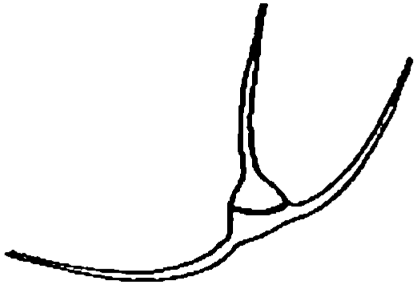
7



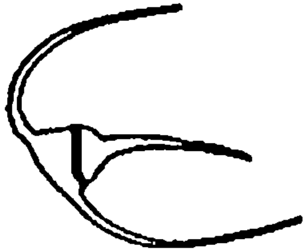
8



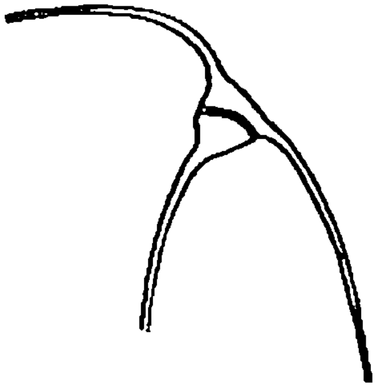
9



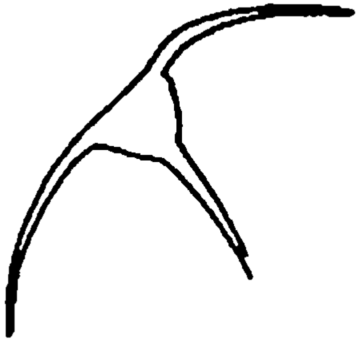
10



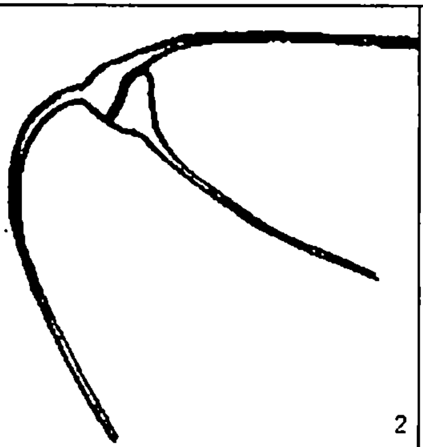
11



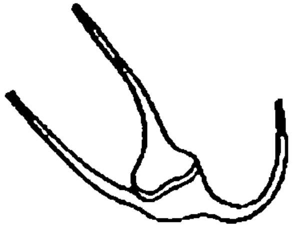
12



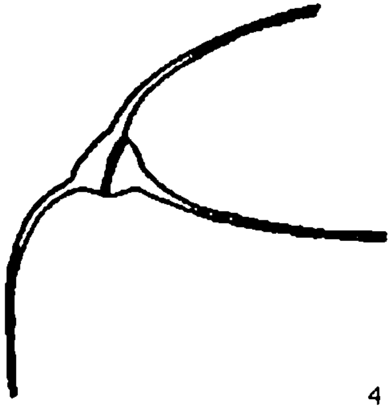
1



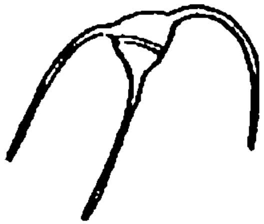
2



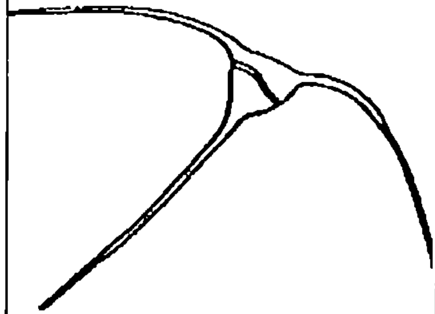
3



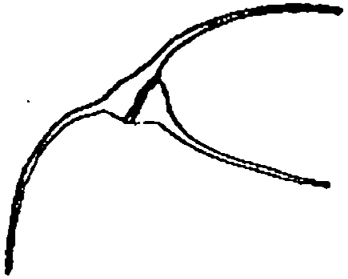
4



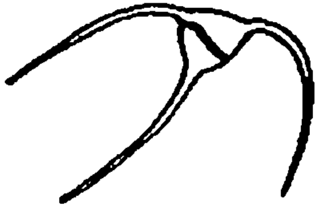
5



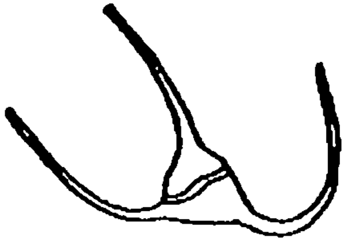
6



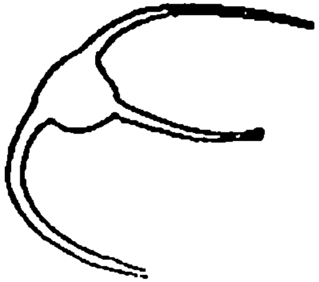
7



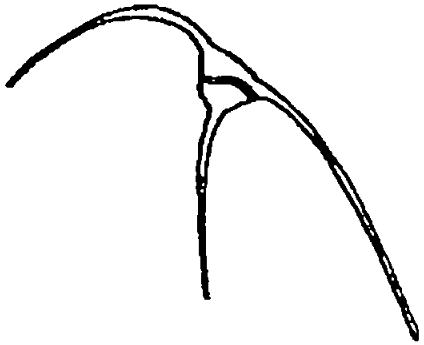
8



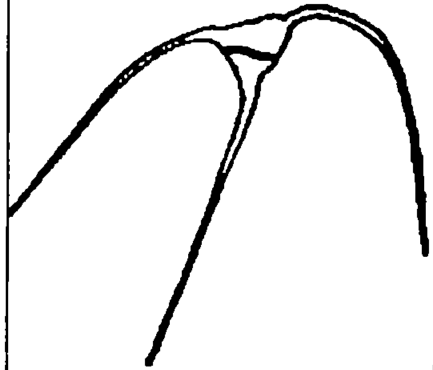
9



10



11

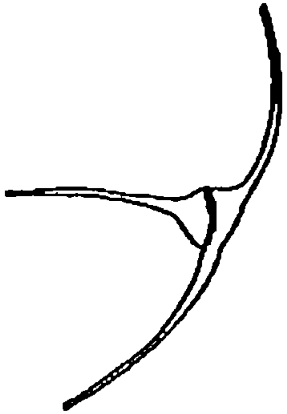


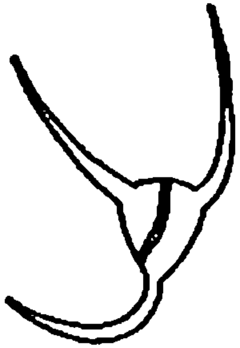
12

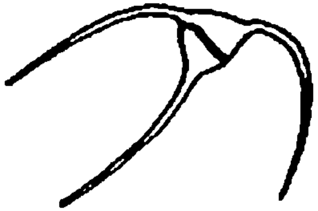
Ceratium classification trials, (c) R. Simpson 1991

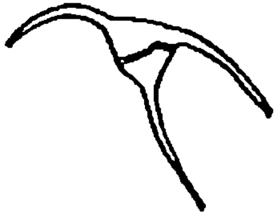
PML
Feb.
1991

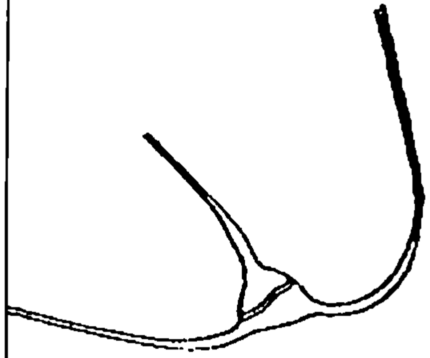
Page
12.15

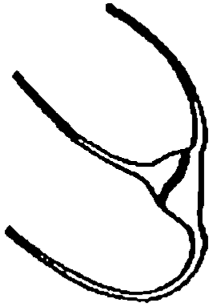

1

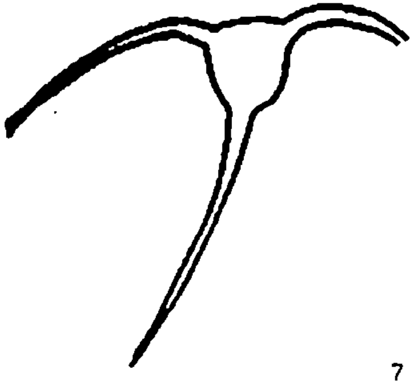

2

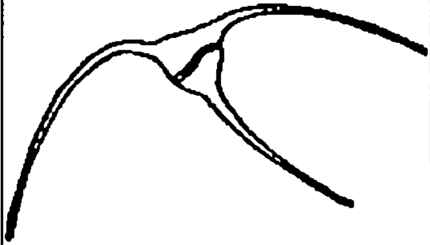

3

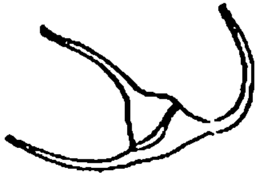

4

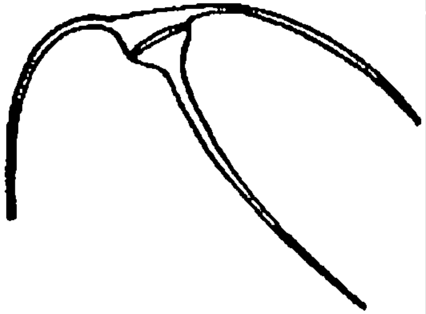

5

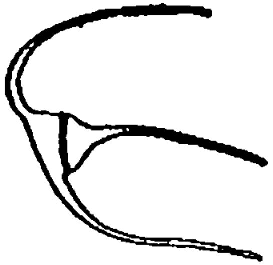

6

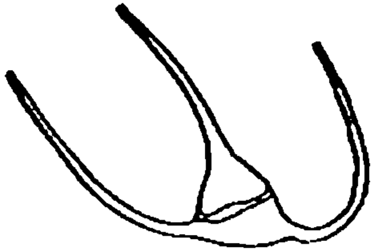

7


8


9

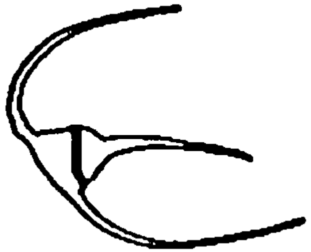

10

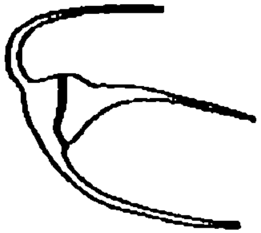

11

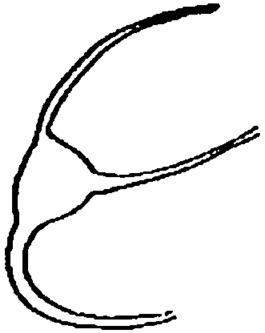

12

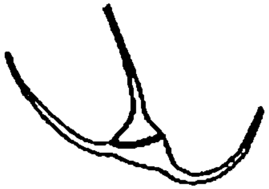
PML
Feb.
1991

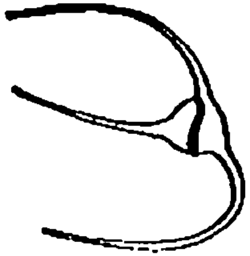
Page
12.16

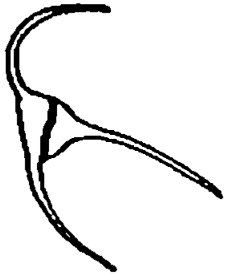

1

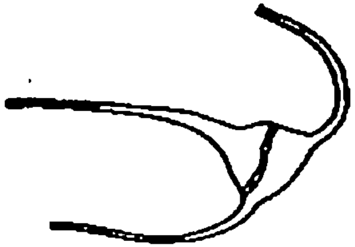

2

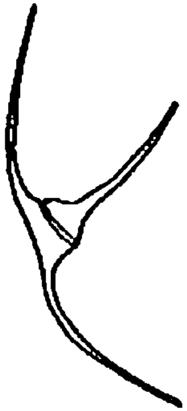

3

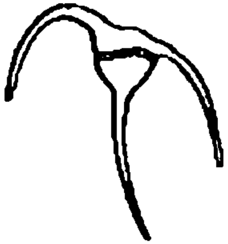

4

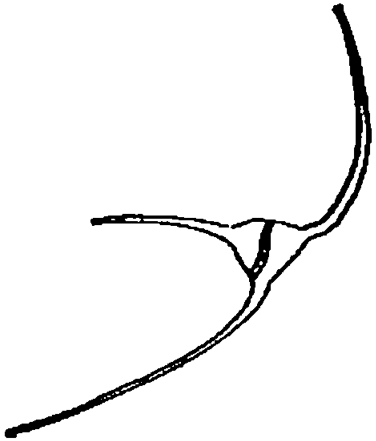

5

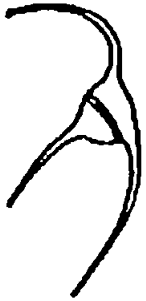

6

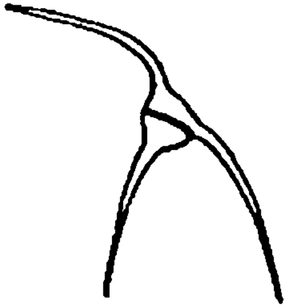

7


8


9


10

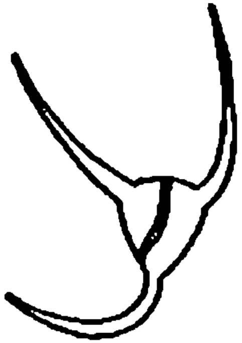

11

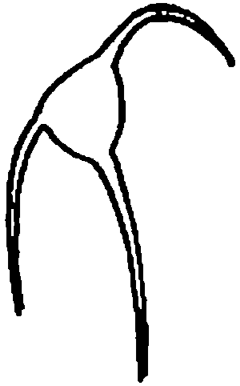

12

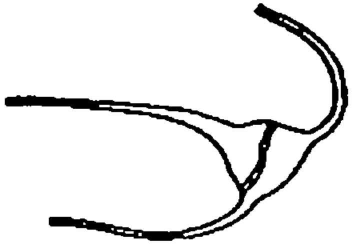
Ceratium classification trials, (c) R. Simpson 1991

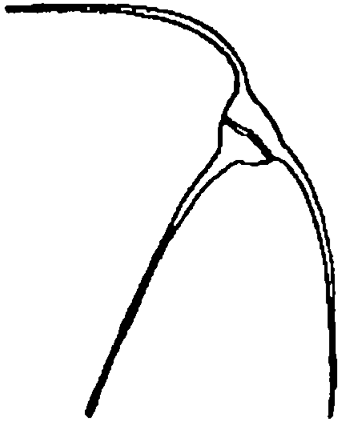
PML
Feb.
1991

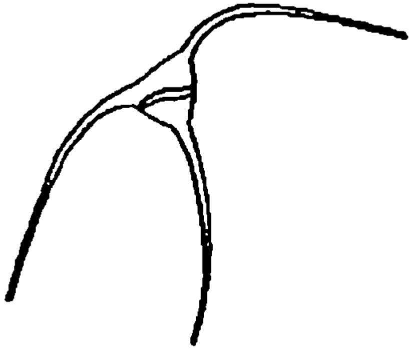
Page
12.17

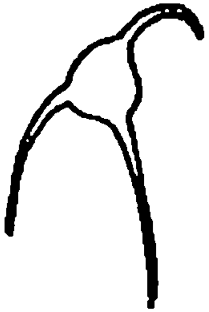

1

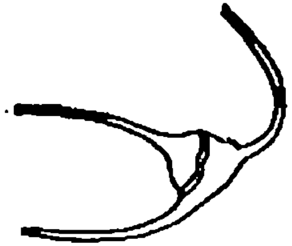

2

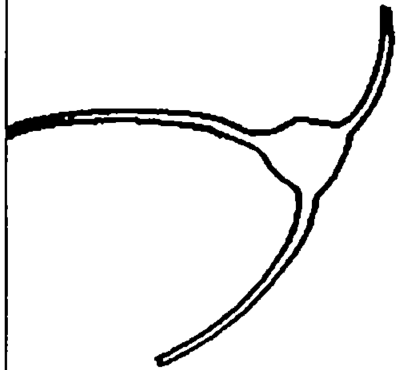

3

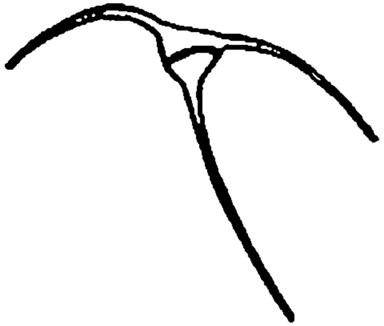

4

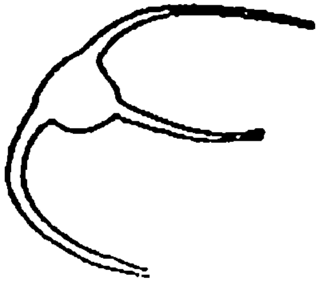

5

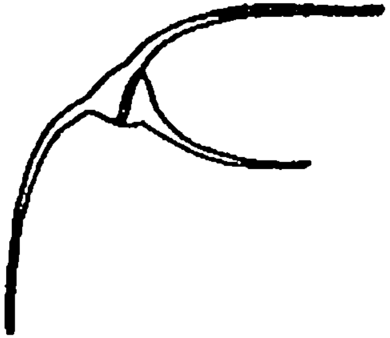

6

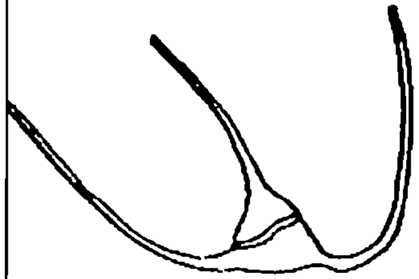

7


8


9

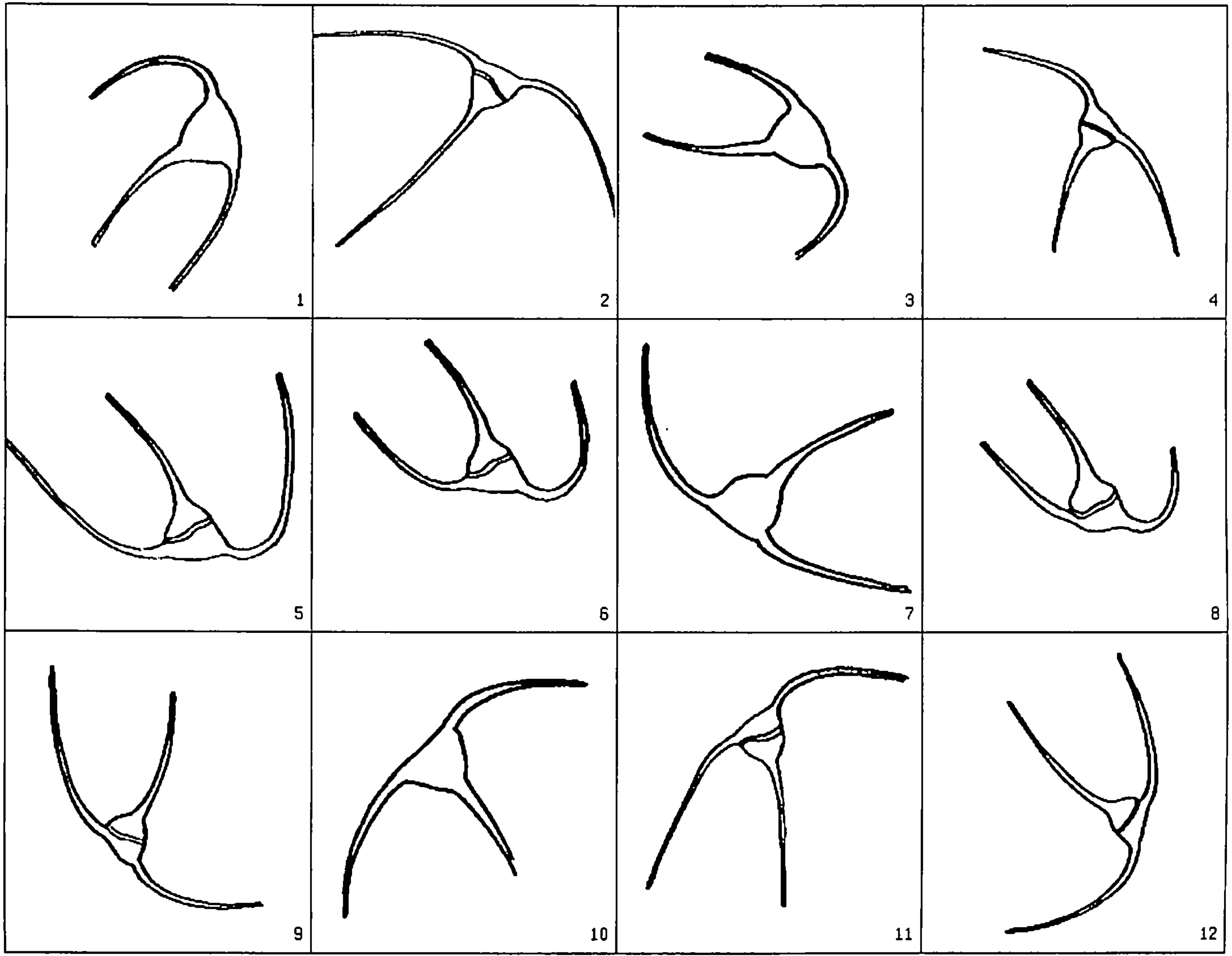

10


11


12

PML
Feb.
1991

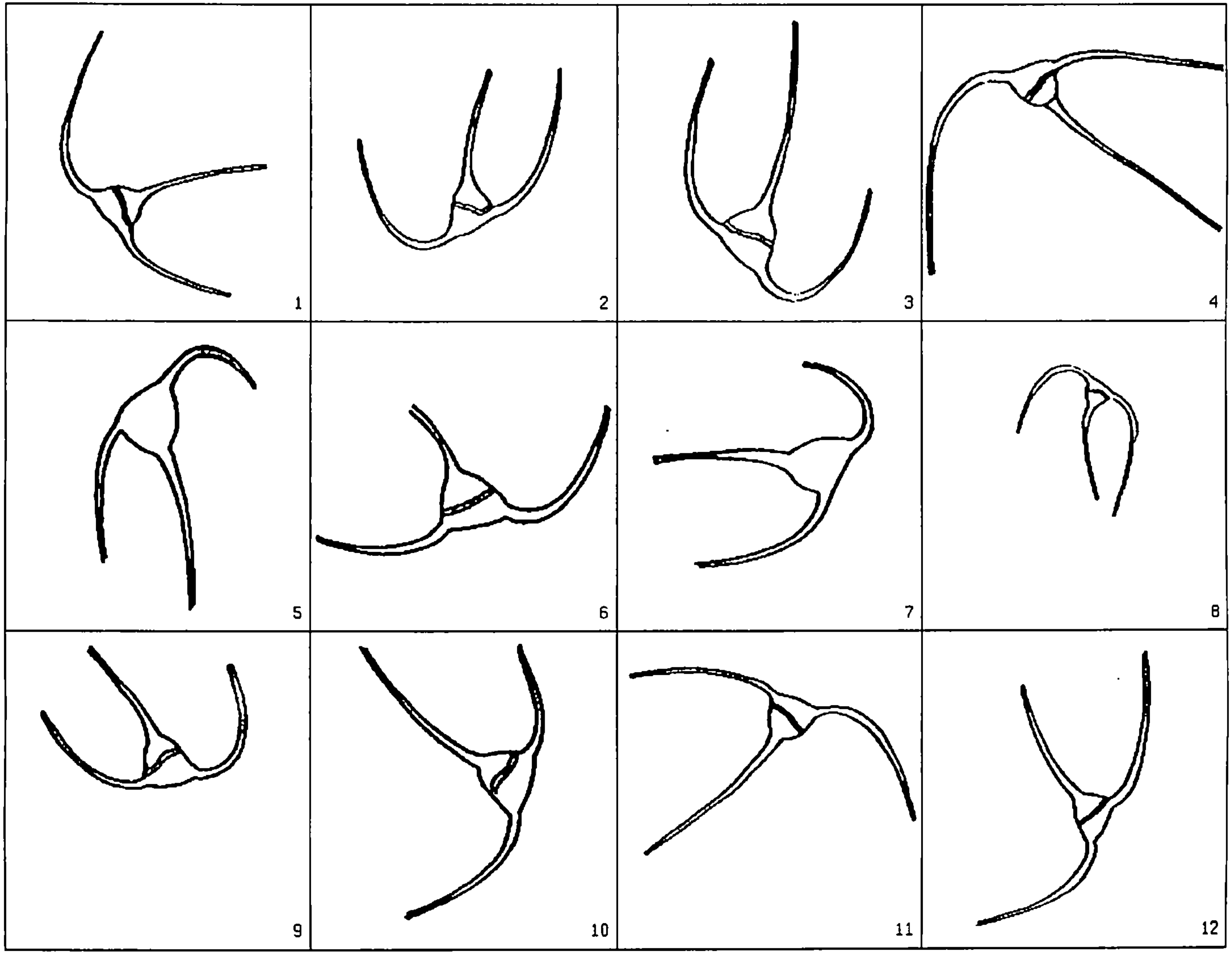
Page
12.18


1
2
3
4
5
6
7
8
9
10
11
12

Ceratium classification trials, (c) R. Simpson 1991

PML
Feb.
1991

Page
12.19


1
2
3
4
5
6
7
8
9
10
11
12

References.

Agin, G. L. (1980) Computer vision systems for industrial inspection and assembly, *Computer (IEEE), 13(5).*

Ahmad, S and G. Tesauro (1988) Scaling and generalisation in neural networks: A case study, *Proceedings of the 1988 IEEE Conference on Neural Information Processing Systems, 1*, 160–176, New York: American Institute of Physics.

Aleksander, I. (Ed.) (1981) *Artificial vision for robots*, London: Kogan Page.

Ali, D. L., A. L. Ali and K. S. Ali (1988) Connectionist experts systems for process planning, *Neural Networks, 1*, supp.1.

Alkon, D. L. (1989) Memory storage and neural systems, *Scientific American*, July.

Allred, L. G. and G. E. Kelly (1990) Supervised learning techniques for backpropagation networks, *Proceedings of the International Joint Conference on Neural Networks, 1*, 721–728, San Diego, CA.

Armstrong, S. L., L. R. Gleitman and H. Gleitman (1983) What some concepts might not be, *Cognition, 13*, 263–308.

Amari, S–I. (1967) A theory of adaptive pattern classifiers, *IEEE Transactions on Electronic Computers, EC-16*, 299–307.

Anderson C. H. and E. Abrahams (1986) A Bayesian probability network, *AIP Conference Proceedings 151: Neural Networks for Computing*, 7–11, New York : American Institute of Physics.

Anderson J. A. and E. Rosenfeld (1988) *Neurocomputing: foundations of research*, Cambridge MA: MIT Press.

Arai, M. (1989) Mapping abilities of three-layer neural networks, *Proceedings of the International Joint Conference on Neural Networks, 1*, 419–423, Washington D.C., U.S.A..

Attneave, F. (1950) *American Journal of Psychology, 63*, 516.

Barrett, R., A. Ramsey and A. Sloman (1985) *POP-11: A practical language for artificial intelligence*, England: Ellis Horwood Ltd.

Ballard, D. H. and J. Sklansky (1976) A ladder-structured decision tree for recognising tumours in chest radiographs, *IEEE Trans. Computers*, *25*, 503-513.

Barrow, H. G. (1989) A.I., neural networks and early vision, *AISB Quarterly*, *69*, 6-25.

Barrow, H. G. and J. M. Tenenbaum (1981) Computational vision, *Proceedings of the IEEE*, *69(5)*.

Baum, E. B. and F. Wilczek (1988) Supervised learning of probability distributions by neural networks, *Proceedings of the 1987 IEEE Conference on Neural Information Processing Systems- Natural and Synthetic*, 52-61, New York: American Institute of Physics.

Baxt, W. G. (1992) Improving the accuracy of an artificial neural network using multiple differently trained networks, *Neural Computation*, *4*, 772-780.

Boden, M. A. (1977) *Artificial Intelligence and Natural Man*, New York; Basic Books.

Brachman, R. J. (1983) What IS-A is and isn't: an analysis of taxonomic links in semantic networks, *Computer (IEEE)*, *16*, 30-36.

Brady, M., R. Raghavan and J. Slawny (1989) Back propagation fails to separate where perceptrons succeed, *IEEE Trans. Circuits and Systems*, *36*, 665-674.

Brooks, J. L. and S. L. Dodson (1965) Predation, body size and composition of plankton, *Science*, *150*, 28-35.

Brown, L. M., I. Gargantini, D. J. Brown, H. J. Atkinson, J. Govindarajan, G. C. Vanlerberghe (1989) Computer-based image analysis for the automated counting and morphological description of microalgae in culture, *J. Appl. Phycol.*, *1(3)*, 211-225.

Brown, R. and E. H. Lenneburg (1954) A study in language and cognition, *Journal of Abnormal and Social Psychology*, *44*, 454-62.

Bruner, J. S., J. J. Goodnow and J. G. Austin (1956) *A Study of Thinking*, New York: Wiley.

Campana, S. E. (1987) Image analysis for microscope-based observations: An inexpensive configuration, *Canadian Technical Report of Fisheries and Aquatic Sciences*, No. 1569.

Campbell, F. W. and J. G. Robson (1968) Application of Fourier analysis to the visibility of gratings, *Journal of Physiology (London), No. 197*, 551-566.

Carnevali, P., L. Coletti and S. Patarnello (1985) Image processing by simulated annealing, *IBM Journal of Research and Development, 29*, 569-579.

Carpenter, G. A. and S. Grossberg (1987) A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics and Image Processing, 37*, 54-115.

Carroll, J. B. (Ed.) (1956) *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf*, Cambridge, MA: MIT Press.

Cater, J. P. (1987) Successfully using peak learning rates of 10 (and greater) in back-propagation networks with the heuristic learning algorithm, *Proceedings of the first IEEE International Joint Conference on Neural Networks, vol. II*, 645-651, San Diego, CA.

Cheung, R. K. M., I. Lustig and A. L. Kornhauser (1990) Relative effectiveness of training set patterns for back propagation, *Proceedings of the first IEEE International Joint Conference on Neural Networks, vol. I*, 673-678, San Diego, CA.

Cover, T. M. (1969) Learning in pattern recognition, *In;* Watanabe 1969, 111-132.

Cover, T. M. and P. Hart (1967) Nearest neighbour pattern classification, *IEEE Transformation on Information, IT-13*, 21-27.

Cushing, D. H. and H. F. Nicholson (1966) Method of estimating algal production rates at sea, *Nature, 212*, 310-311.

Dasarathy, B. V. and B. V. Sheela (1977) Visiting nearest neighbours, *Proceedings of the International Conference on Cybernetics and Society*, September, 630-635.

Dudani, S. A. (1976) The distance weighted k-nearest neighbour rule, *IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-6*, 325-327.

Ehrenberg, C. G. (1839) Über noch jetzt zahlreich lebende Thierarten der Kreidebildung und den Organismus der Polythalamien, *Abhandlungen der Deutschen Akjademie der Wissenschaften zu Berlin*, 81-174.

Elman, J. L. and D. Zipser (1988) Learning the hidden structure of speech, *J. Acoust. Soc. Amer.*, *83*, 1615-1626.

Fiegenbaum, E. A. (1977) The art of artificial intelligence: themes and case studies of knowledge engineering, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 1014-29.

Feldman, J and D. Ballard (1982) Connectionist models and their properties, *Cognitive Science*, *6*, 205-254.

Fisher, R. A. (1925) Theory of statistical estimation, *Proceedings of Cambridge Philosophical Society*, *22*, 700.

Fisher, R. A. and F. Yates (1963) *Statistical tables for biological, agricultural, and medical research*, 14th ed., New York: Hafner Publishing Company, Inc..

Fix, E. and J. L. Hodges (1952) Discriminatory analysis - Nonparametric discrimination: consistancy properties, *Project 21-49-004*, *11*, Randolph Field, Texas: USAF School of Aviation Medicine, 280-322.

Garey, M. R. and D. S. Johnson (1979) *Computers and intractability: A guide to the theory of NP-completeness*, San Francisco, CA: W. H. Freeman and Co..

Garner, W. R. (1974) *The processing of information and structure*, New York: Wiley.

Ghosh, S., E. A. Collins and C. L. Scofield (1988) Prediction of mortgage loan performance, *Neural Networks Supplement: INNS Abstracts, vol. 1.*

Goodman, N. (1973) *Fact, Fiction and Forecast*, Indianapolis, IN: Bobbs-Merrill.

Gorman, R. P. and T. J. Sejnowski (1988) Analysis of hidden units in a layered network trained to classify sonar targets, *Neural Networks*, *1*, 75-89.

Gray, D. L., A. N. Michel and W. Porod (1988) Application of neural networks to sorting problems, *Neural Networks Supplement: INNS Abstracts*, *1*.

Gross C. G., C. E. Rocha-Miranda and D. B. Bender (1972) Visual properties of neurons in inferotemporal contex of the macaque, *Journal of Neurophysiology, 35*, 96–111.

Grossberg, S. (1976) Adaptive pattern classification and universal recoding. I. Parallel development and coding of neural feature detectors. *Biological Cybernetics, 23*, 121–134.

Grossberg, S. (1988) *Neural networks and natural intelligence*, Cambridge, MA: MIT Press.

Guttman, N. and H. I. Kalish (1956) *Journal of Experimental Psychology, 51*, 79.

GyÖrgyi, G. (1990) Inference of a rule by a neural network with thermal noise, *Phys. Rev. Lett., 64*, 2957-2960.

Hampshire J. B. and A. H. Waibel (1990) A novel objective function for improved phoneme recognition using time-delay neural networks, *IEEE Transactions on Neural Networks, 1*, 216-228.

Hanson, A. and E. Riseman (Eds.) (1978) *Computer Vision Systems*, New York: Academic press.

Hanson, S. and D. Burr (1988) Minkowski-r back-propagation: Learning in connectionist models with non-Euclidean error signals, *Proceedings of the 1987 IEEE Conference on Neural Information Processing Systems- Natural and Synthetic*, 348–357, New York: American Institute of Physics.

Hart, P. E. (1968) The condensed nearest neighbour rule, *IEEE Trans on Information Theory, Vol. IT-14*, 515-516.

Harvard Computational Laboratory (1955) *Tables of the cumulative binomial probability distribution*, Cambridge, MA: Harvard University Press.

Hayes, P. (1985) Second naïve physics manifesto, *In;* Hobbs and Moore (1985).

Hebb, D. O. (1949) *The organisation of behavior*, New York: Wiley.

Heider, E. R. (1972) Universals in colour naming and memory, *Journal of Experimental Psychology, 93*, 10-20.

Hinton, G. E., J. L. McClelland and D. E. Rumelhart (1986) Distributed representations, *In;* Rumelhart and McClelland 1986.

Hinton, G. E. and T. J. Sejnowski (1986) Learning and relearning in Boltzmann machines, *In;* Rumelhart and McClelland 1986.

Hobb, J. and R. Moore (Eds.) (1985) *Formal theories of the commonsense world*, New York: Ablex.

Hofstadter, D. R. (1982) An escape from the Boolean dream, *Metamagical Themas*, London: Viking.

Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational properties, *Proceedings of the National Academy of Sciences, 79*, 2554–2558.

Hopfield J. J. and D. W. Tank (1986) Computing with neural circuits: A model, *Science*, *233*, 625–633.

Horn, B. K. P. (1975) Obtaining shape from shading information, *In;* Winston (1975), 115–155.

Hornik, K., M. Stinchcombe and H. White (1989) Multilayer feedforward networks are universal approximators, *Neural Networks*, *2*, 359–366.

Hubel, D. H. and T. N. Wiesal (1977) Functional architecture of the macaque monkey visual cortex, *Proceedings of the Royal Society of London*, *198*, 1–59.

Hummel, R. A. and S. W. Zucker (1983) On the foundations of relaxation labelling processes, *IEEE PAMI*, *5*, 267–287.

Ishii, T. R. Adachi, M. Omori, U. Shimizu and H. Irie (1987) The identification , counting and measurement of phytoplankton by an image-processing system, *J. Cons. int. Explor. Mer.*, *43*, 253–260.

Jacobs, R A. (1988) Increased rates of convergence through learning rate adaptation, *Neural Networks*, *1*, 295–307.

James, W. (1894/1961) *Briefer Psychology*, New York: Collier.

Jarvis, R. A. and E. A. Patrick (1973) Clustering using a similarity measure based on shared nearest neighbors, *IEEE Transactions on Computers*, *C-22*, 1025–1034.

Jeffries, H. P., M. S. Berman, A. D. Poularikas, C. Katsinis, I. Melas, K. Sherman and L. Bivins (1984). Automated sizing, counting and identification of zooplankton by pattern recognition. *Marine Biology*, 329-334.

Jeffries, H. P., K. Sherman, R. Maurer, and C. Katsinis (1980). Computer-processing of Zooplankton Samples. In: *Estuarine Perspectives*, 303–316. V. Kennedy (Ed.) New York: Academic Press

Johnson, E. S. (1978) Validation of concept-learning strategies, *Journal of Experimental Psychology: General*, *107*, 237–266.

Johnson-Laird, P. N. and P. C. Wason (1977) *Thinking: Readings in Cognitive Science*, Cambridge: Cambridge University Press.

Jörgensen, E. (1923) Mediterranean Dinophysiaceae, *Report on the Danish oceanographical expeditions 1908–10 to the Mediterranean and adjacent seas*, *II*, *Biology*, 1–48.

Judd, J. S. (1990) *Neural Network Design and the Complexity of Learning*, Cambridge, MA: MIT Press.

Judd, J. S. (1992) Why are neural networks so wide?, *Proceedings of the 1992 International Conference on Artificial Neural Networks, Brighton, UK*, *1*, 45–52.

Julesz, B. (1960) Binocular depth perception of computer generated patterns, *Bell Systems Technical Journal.*, *39*, 1125–1162.

Kirkpatrik, S., C. D. Gelatt and M. P. Vecchi (1983), Optimisation by simulated annealing, *Science*, *220*, 671–680.

Kohonen, T. (1988) An introduction to neural computing, *Neural Networks*, *1*, 3–16.

Koopman, P., J. Gubbay, N. Vivian, P. Goodfellow and R. Lovell-Badge (1991) *Nature*, *351*, 117–121.

Krogh, A. (1991) Learning with noise in a linear perceptron, *NORDITA preprint.*

Lacouture, Y. and A. A. J. Marley (1991) A connectionist model of choice and reaction time in absolute identification, *Connection Science, 3*, 401–433.

Lakoff, G. (1987) *Women, Fire and Dangerous Things*, Chicago, IL: University of Chicago Press.

Larsen, J. and Ø. Moestrup (1992) Potentially Toxic Phytoplankton, 2, Genus *Dinophysis* (Dinophyceae), *ICES Identification Leaflets for Plankton*, 180.

Latous, S. (1984) Pattern recognition and automatic zooplankton classification by image analysis, *Doctoral Thesis, University of Rennes*, France.

LeCun, Y. (1989) Generalisation and network design strategies, *Technical Report CGR-TR-89-4*, Dept. of Computer Science, University of Toronto.

Lenat, D. B. (1977) On automated scientific theory formation: A case study using the AM program, *Machine Intelligence, 9*, 251–286, New York: Halsted Press.

Lendaris, G. G. (1990) A proposal for indicating quality of generalisation when evaluating ANNS, *Proceedings of the International Joint Conference on Neural Networks, San Diego, I*, 709–713.

Lippmann R. (1987) An introduction to computing with neural nets, *IEEE ASSP Magazine, 4*, 4–22.

Ljungqvist, A. and J. L. Simpson (1992) *J. Am. Med. Ass., 267*, 850–852.

Mackas, D. L., T. A. Curran and D. SLoan (1981) An electronic zooplankton counting and sizing scheme, *Proceedings of Oceans 81 Conference, Boston, MA.*

Matthews, J. B. L. (1967) *Calanus Finmarchicus* in the North Atlantic, *Bulletin of Marine Ecology, 6*, 159–179.

Marr, D. (1974) A note on the computation of binocular disparity in a symbolic, low-level visual processor, *MIT A.I. Laboratory Memo 327.*

Marr, D. (1978) Representing visual information, *In;* Hanson and Riseman (Eds.) (1987).

Marr, D. (1982) *Vision*, San Francisco, CA: W. H. Freeman and Co..

Marr, D. and T. Poggio (1976) Cooperative computation of stereo disparity, *Science, 194*, 283–287.

Marr, D. and T. Poggio (1979) A computational theory of human stereo vision, *Proceeding of the Royal Society of London*, Series B, No. 204.

McClelland J. L. and D. E. Rumelhart (Eds.) (1986a) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 2*, Cambridge, MA: Bradford Books/MIT Press.

McClelland J. L. and D. E. Rumelhart (1986b) A distributed model of human learning and memory, *In:* McClelland and Rumelhart (Eds.) (1986a).

McCulloch, W. S. and W Pitts (1943) A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys., 5*, 115–137.

McInery, J. M., K. G. Haines, S. Biafore and R. Hecht-Nielson (1988) Can backpropagation error surfaces have non-global minima?, *Department of Electrical and Computer Engineering Manuscript*, University of California at San Diego.

Minsky, M. A. (1975) A framework for representing knowledge, *In;* Winston (1975).

Minsky, M. and S. Papert (1969) *Perceptrons: An introduction to computational geometry*, Cambridge, MA: MIT Press.

Minsky, M. and S. Papert (1988) *Perceptrons: An introduction to computational geometry*, Expanded edition, Cambridge, MA: MIT Press.

Moore, T. E. (Ed) (1973) *Cognitive Development and the Acquisition of Language*, New York: Academic Press.

Nosofsky, R. M. (1986) Attention, similarity and the identification–categorization relationship, *Journal of Experimental Psychology: General, 115*, 39–57.

Olson R. J., D. Vaulot and S. W. Chisholm (1985) Marine phytoplankton distributions using shipboard flow cytometry, *Deep-sea Res., 32(10a)*, 1273–1280.

Olson R. J., E. R. Zettler and O. K. Anderson (1989) Discrimination of eukaryotic phytoplankton cell types from light scatter and autofluorescence properties measured by flow cytometry, *Cytometry, 10(5)*, 636–643.

Osherson, D. N. and E. E. Smith (1981) On the adaquacy of prototype theory as a theory of concepts, Cognition, 9, 35–58.

Page, C. and A. Pugh (1981) Visually interactive gripping of engineered parts from random orientation, *In;* Aleksander (Ed.) (1981).

Parker, D. (1982) Learning logic, Invention report, S81–64, File 1, Office of Technological Licensing, Stanford, CA: Stanford University Press.

Parsons, T. R. (1969) The use of particle size in determining the structure of a plankton community, *Journal of the Oceanographic Society Japan, 25*, 172–181.

Pavlov, I. P. (1927) *Conditioned Reflexes*, (G. V. Anrep, translator) London: Oxford University Press.

Peeling, S. M., R. K. Moore and M. J. Tomlinson (1986) The multi-layer perceptron as a tool for speech pattern processing research, in *Proc. IoA Autumn Conf. Speech and Hearing*.

Posner, M. I. and S. W. Keele (1968) On the genesis of abstract ideas, *Journal of Experimental Psychology, 77*, 353–363.

Quine, W. V. (1969a) *Ontological relativity and other essays*, New York; Columbia University Press.

Quine, W. V. (1969b) Natural kinds, *In;* Quine (1969a).

Roberts, L. G. (1965) Machine perception if three-dimensional solids, *In;* Tippett et al. (1965).

Rolke, M. and J. Lenz (1984) Size structure analysis of zooplankton samples by means of an automated image analyzing system, *Journal of Plankton Research, 6(4)*, 637–645.

Romig, H. (1953) *50–100 binomial tables*, New York: John Wiley and Sons.

Rosch, E. (1975) Cognitive representations of semantic categories, *Journal of Experimental Psychology: General, Vol. 104, No. 3*, 192–233.

Rosch, E. and B. B. Lloyd (1978) *Cognition and Categorization*, Hillsdale, N.J.: Lawrence Erlbaum.

Rosch, E. and Mervis C. B. (1975) Family resemblance studies in the internal structure of categories, *Cognitive Psychology, 7*, 573–605.

Rosch, E. (1973) On the internal structure of perceptual and semantic categories, *In;* Moore (Ed.) (1973), 111–114.

Rosenblatt, R. (1957) The perceptron: A perceiving and recognizing automation (project PARA), *Cornell Aeronautical Laboratory Report, 85-460-1.*

Rosenblatt, R. (1958) The perceptron: a probabilistic model for information storage and organization in the brain, *Psychological Review, 65*, 386–408.

Rosenblatt, R. (1959) Two theorems of statistical separability in the perceptron. *Mechanization of thought processes, Vol. 1*, London: H.M.S.O.

Rosenblatt, R. (1962), *Principles of neurodynamics*, New York: Spartan.

Rothkopf, E. Z. (1957) *Journal of experimental psychology, 54*, 94.

Rumelhart, D. E., G. E. Hinton and R. Williams (1986a) Learning representations by backpropagating errors, *Nature, 323*, 533–536.

Rumelhart, D. E., G. E. Hinton and R. Williams (1986b) Learning internal representations by error propagation, *In;* Rumelhart and McClelland (Eds.) (1986), 318–362.

Rumelhart, D. E. and J. L. McClelland (Eds.) (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1*, Cambridge MA: Bradford Books/MIT Press.

Rumelhart, D. E. and D. Zipser (1985) Feature discovery by competitive learning, *Cognitive Science, 9*, 75–112, Ablex Publishing.

Shapiro, S. C. and D. Eckroth (Eds.) (1987) *Encyclopedia of Artificial Intelligence*, New York: Wiley.

Sheldon, R. W. and T. R. Parsons (1967) A continuous size spectrum for particulate matter in the sea, *J. Fish. Res. Bd. Canada, 24*, 909–915.

Shepard, R. (1964) Attention and the metric structure of the stimulus space, *Journal of Mathematical Psychology, 1*, 287.

Shepard, R. (1987) Toward a universal law of generalization for psychological science, *Science, 237*, 1317–1323.

Sietsma, J. and R. J. F. Dow (1991) Creating artificial neural networks that generalise, *Neural Networks, 4*, 67–79.

Simpson, P. (1990) *Artificial neural systems: foundations, paradigms, applications and implementations*, New York: Pergamom Press Inc.

Simpson, R., R. Williams, R. Ellis and P. F. Culverhouse (1992) Biological pattern recognition by neural networks, *Marine Ecology Progress Series, 79*, 303–308.

Smith, E. E. and D. L. Medin (1981) *Categories and Concepts,* Cambridge, MA: Harvard University Press.

Solinsky, J. C. and E. A. Nash (1991) Neural-network performance assessment in sonar applications, *Proceedings of the IEEE Conference on Neural Networks for Ocean Engineering*, 1–12.

Sontag, E. D. and H. J. Sussmann (1991) Backpropagation separates where perceptrons do, *Neural Networks, 4*, 243–249.

Stornetta, W. and B. Huberman (1987) An improved three-layer backpropagation algorithm, *Proceedings of the IEEE First International Conference on Neural Networks: II*, 637–644, San Diego, CA: IEEE.

Swonger, C. W. (1972) Sample set condensation for condensed nearest neighbor rule for pattern recognition, *In;* Watanabe (1972), 511–526.

Tippett, J. P. (1965) *Optical and electro-optical information processing*, Cambridge, MA: MIT Press.

Tversky, A. and I. Gati (1982) Similarity, separability, and the triangle inequality, *Psychological Review, 89*, 123–154.

Ullman, Shimon (1984) Visual routines, *Cognition, 18.*

Van Wambeke F. (1988) Enumeration and sizing of planktonic bacteria by image-analysed epifluorescence microscopy, *Ann. Inst. Pasteur Microbiol, 139(2)*, 261–272.

Viles, C. L. and M. E. Sieracki (1992) Measurement of marine picoplankton cell size by using a cooled, charge-coupled device camera with image-analyzed fluorescence microscopy, *Appl. Environ. Microbiol., 58(2)*, 584–592.

Waltz, D. (1975) Understanding line drawings of scenes with shadows, *In;* Winston (1975).

Watanabe, S. (Ed.) (1969) *Methodologies of Pattern Recognition*, New York: Academic Press.

Watanabe, S. (Ed.) (1972) *Frontiers of Pattern Recognition*, New York: Academic Press.

Watanabe, S. (1985) *Pattern Recognition: Human and Mechanical*, New York: John Wiley and Sons.

Widrow, B and M. Hoff (1960) Adaptive switching circuits, *1960 WESCON Convention Record: Part 4*, 96–104.

Wilson, D. L. (1972) Asymptotic properties of nearest neighbour rules using edited data. *IEEE Trans. of Systems, Man and Cybernetics, Vol. SMC-2*, 408–421.

Wittgenstein, L. (1953) *Philosophical Investigation*, (Anscombe, translator) New York: McMillan Co..

Winston P. H. (Ed.) (1975) *The psychology of computer vision*, New York: McGraw-Hill.

Werbos, P. (1974) Beyond regression: New tools for prediction and analysis in the behavioral sciences, Ph.D. Dissertation, Harvard University.

Yarranton, G. A. (1967) Parameters for use in distinguishing populations of Euceratium Gran, *Bulletin of Marine Ecology, 6*, 147–158.

Zadeh, L. A. (1975) Fuzzy sets, *Information Control*, *8*, 338–353.