The Development Of Artificial Neural Networks For The Analysis Of Market Research And Electronic Nose Data

Andrew B Larkin

Submitted For The Degree Of Doctor Of Philosophy

Department of Engineering,

University of Warwick,

UK

March 1995

Table of Contents

Chapter 3. Statistical Clustering Techniques

Chapter 4. Self-Organising Maps and Memory Arrays

Chapter 5. Clustering with Interrogative Memory Structures

Chapter 8. Results

Chapter 9. Conclusions and Discussions

Appendix A.

Acknowledgements

The supervision of this project by Prof. D J Whitehouse and proof reading of this thesis by Joanne Larkin are gratefully acknowledged.

The author is also grateful to EPSRC; Parallax Management Consultancy Ltd, Coventry; and The Management Consulting Group Ltd, Coventry; for providing funding for this work, and to Benchmark research Ltd, Kent; for providing the market research data used, and to Dr Julian Gardner for providing the Electronic Nose data.

The help of Mr Rod Horrocks of Parallax Management Consultancy is gratefully acknowledged for providing support, finance and useful contacts.

Declaration

This thesis is presented in accordance with the regulations for the degree of doctor of philosophy. All work reported has been carried out by the author unless otherwise stated, including the production of this document.

Summary

This thesis details research carried out into the application of unsupervised neural network and statistical clustering techniques to market research interview survey analysis. The objective of the research was to develop mathematical mechanisms to locate and quantify internal clusters within the data sets with definite commonality. As the data sets being used were binary, this commonality was expressed in terms of identical question answers. Unsupervised neural network paradigms are investigated, along with statistical clustering techniques. The theory of clustering in a binary space is also looked at.

Attempts to improve the clarity of output of Self-Organising Maps (SOM) consisted of several stages of investigation culminating in the conception of the Interrogative Memory Structure (IMS). IMS proved easy to use, fast in operation and consistently produced results with the highest degree of commonality when tested against SOM, Adaptive Resonance Theory (ART1) and FASTCLUS. ART1 performed well when clusters were measured using general metrics. During the course of the research a supervised technique, the Vector Memory Array (VMA), was developed. VMA was tested against Back Propagation (BP) (using data sets provided by the Warwick electronic nose project) and consistently produced higher classification accuracies. The main advantage of VMA is its speed of operation - in testing it produced results in minutes compared to hours for the BP method, giving speed increases in the region of 100:1.

In the process of this research three papers were produced covering the three main areas of research. Two have been published, one is currently in press; both published works can be found in

appendix A.

Chapter 1
Introduction

1.1 General

Artificial neural networks have been the subject of much research over the past decade. Theory has developed in tandem with application areas and in many cases (including this one) theoretical research has had to be carried out as part of the development of the application area. Analysis with networks has penetrated many diverse fields of research such as speech recognition [1], stock price prediction [2] and the classification of data collected from an electronic nose [3]. Neural networks are used and researched by biologists, statisticians, psychologists, engineers, computer scientists and other workers. Engineering and computer science schools all over the world have developed theory and applied the techniques to new application areas. This, perhaps unique cross disciplinary development has led to the area advancing far more quickly than its tender years would suggest. The fundamental principle of networks being able to "learn" associations and inter-relationships makes them ideal for both modelling the real world and integration into it.

The parallel nature of the neural architecture has also created interest in its implementation in hardware. Various hardware implementations exist. Some have pursued implementation in the highly parallel environment of transputer arrays [4], others have created custom VLSI technology containing large numbers of nodes [5].

Analysis of interview forms is a subject area that has been in existence for decades. Many statistical techniques exist and the area has been developed into a highly scientific discipline used in areas as diverse as consumer surveys to psychological psychometric profiling. A state of the art interview form will have been designed

specifically for the easy extraction and classification of the information required. Statistical analysis techniques vary in their approaches but almost all aim to achieve the same result - classification. A small number of techniques has been developed which can further analyse data obtained from interview forms to find internal groupings and structure within a data set. The techniques known as clustering algorithms are rarely used as they do not guarantee a result and elude mathematical analysis. If an application area is decided upon, an interview form will be specifically designed to meet the requirements. This can be an expensive task, and if the population has already been interviewed the information may already be available and in these cases cluster analysis could be used.
Several attempts have been made to use supervised artificial neural networks to classify the results of surveys [6,7,8], but it has proved to be impossible to find literature on any attempts to cluster interview data in an unsupervised manner. This thesis, focuses on the cluster analysis of interview data using artificial neural network techniques. Existing self-organising unsupervised techniques have been used and compared directly with a "state of the art" statistical clustering technique. Development of the area has been carried out with the conception of new techniques. A supervised method of processing was also developed as a furtherance of the work.

The basis of this thesis is two fold. A comparison will be made between statistical clustering techniques and unsupervised neural network algorithms. The application area that necessitated this comparison was the need for the analysis of data taken during interviews in various engineering sector related projects. A statistician will tell you that interview forms should be designed in such a way that supervised classification techniques or contingency table analysis should be used, but this is of

little use if the interview has already been carried out or the user does not know what they are looking for in the data.

1.2 Market Research

Underlying the success of any company will be an element of competitive advantage. This "edge" can be gained in many ways, e.g. the implementation of efficient production techniques. Some elements however, will always remain key. Insight into areas such as the customers' needs and the company's image is of premium importance. One of the most commonly used techniques for obtaining information such as this is to survey the relevant people. To carry out these interviews, research companies will either target specific individuals or groups of people of demographic significance. When the identity of the individual is not important, street surveys are carried out. If there is a need to be more selective, telephone surveys or interviews by personal appointment can be used. Whatever the technique, the result is nearly always the same - a large pile of completed interview forms to be entered into a computer and analysed, to change the raw data into meaningful information. For the most part, analysis carried out will be relatively simplistic. Distributions will be calculated and then extrapolations made. Hypotheses can then be validated against the results and conclusions drawn. The under current behind all this, is that the commissioner of the survey is setting out to discover something very specific. The interview form will have been designed to probe the specific area of interest and the analysis that follows will be tailored to that end. Although there is not necessarily a built-in bias towards validating a specific fact, the methodology is one that could

miss other information contained in the results. The analysis techniques looked at in this thesis do not set out to validate; they simply look for recurring trends in the data. To take an example, if the management personnel of a company do not seem to be performing well, a survey could be commissioned to attempt to locate the problem areas. The questionnaire could probe many aspects such as their time management skills, motivation and team ethics. Traditional analysis would then calculate figures such as 65% of the team have a poor conception of how to motivate a work force etc. Using these techniques it could however, be overlooked that 30% of the team had almost identical question answers showing the need for a specific approach to training.

This thesis then, focuses on the processing of the interview forms. To reduce the problem to a manageable size, this work looks only at binary interview forms, i.e. all the questions are answered with one of two possible answers e.g.

Do you catch the X191 bus to Coventry? *(yes/ no)*

This does not place too great a restriction on the use of the techniques as most questionnaires can be reduced to binary answers. For example a question such as:

How old are you?

Can be reworked as:

Within which age range are you?

(0 - 19)

(20 - 29)

(30 - 39)

(40 - 49)

(≥50)

Each age range can then be taken to be a single binary question. In the above example a person aged 45 would tick the *(40 - 49)* option and this would be interpreted as negative answers to all the other options. Converting this into data with '1' representing a positive answer and '0' representing a negative one, the answer would look like: (0 0 0 1 0)

Taking each completed questionnaire individually, a binary vector can be formed from the answers given. As each questionnaire is converted, a data set (matrix) of binary vectors is formed, each vector representing one person's answers.

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

Table 1.1 An example of a binary data matrix

For people to have given the same response to a question, the column containing the information from that question must contain the same state for all the people (i.e. either '1' or '0'). For example, the eight people in table 1.1 above have answered question 3 identically. If the survey was being carried out by an Information

Technology Consultancy and the client was a company whose aim was to improve their utilisation of IT, question 3 might be:

3) Do you make use of the "Outline" mode in Microsoft Word 6.0? (Yes/ No)

If 100 employees of the client company were surveyed and it was found that 20 of the people had answered "no" to question 3, the consultant would then have good grounds to run a seminar for those people on that subject area.

In this trivial example the consultant could "paper process" to obtain the results without the need for complex algorithms. In reality, searching for the answers to one question does not provide the consultant with much information with which to do his job. A more complex example would be if 500 employees of the client company were surveyed using an interview form with 100 questions, and a processing technique found that 150 of the people had answered 25 of the questions identically. Then the consultant would be able to look at which questions had been answered in that manner and put together tailored seminars for those people based on the needs identified.

This thesis investigates processing techniques that are capable of finding clusters in data sets and thus are capable of performing the function described above. Artificial neural network techniques are compared against a statistical algorithm. A novel paradigm is also presented and benchmarked against existing techniques. Methods for comparing techniques are developed, and the results of analysis run on large data sets are presented. A practical example of the use of these techniques is also included with data taken from a recent survey of the forgings industry.

1.3 Artificial Neural Networks

Research into artificial neural networks is carried out in a diverse number of disciplines. The ideology stems from neurobiology, that found the brain to be composed of many simple interconnected processing units. It is obvious that the human brain has immense processing power and yet the processing carried out by individual elements is not immense. The power must therefore lie in the interconnection strategies and the sheer number of processing elements. Some researchers have devoted time to attempting to accurately model individual neurons or small numbers of them interconnected [9]. For the most part, researchers have taken a mathematically simplistic version of a neuron and built increasingly complex systems.

1.3.1 The Biological Neuron

The processing element of an artificial neural network is loosely based on the processing element of the human brain, the biological neuron. As this thesis is not in the field of neurobiology, very little will be said about the exact nature of a neuron, but for the sake of clarity and consistency a brief word of description will be given. The biological neuron can be viewed in the same manner as any processing element. It has inputs (dendrites), a central processing unit (cell body) and an output (axon). The neuron takes as input electrical potentials generated by the movement of charged ions and outputs a time-dependent voltage.

Voltages from other neurons arrive through connections known as synapses into the dendrites. They will either have a polarising or a depolarising effect on the potential of the cell body. On reaching a state of sufficient depolarisation within the cell body, an "action potential" is generated. This is a voltage spike which is propagated down the axon.

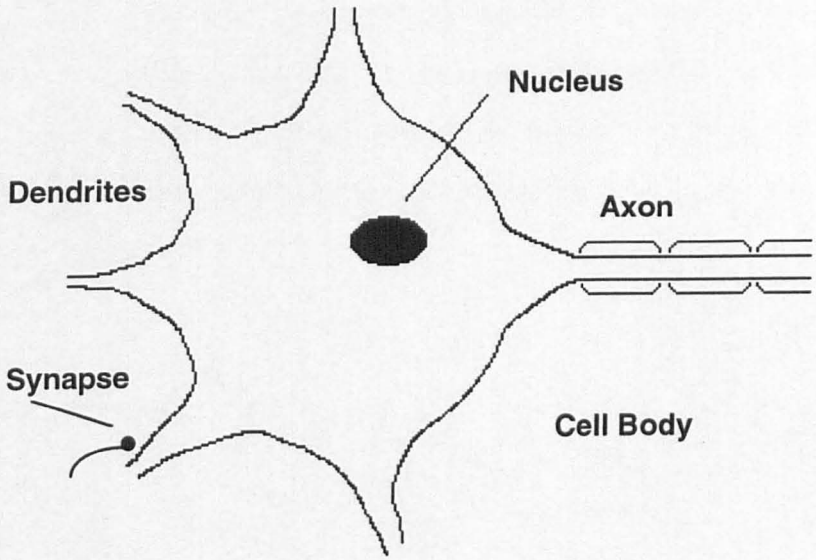


Figure 1.1 An outline diagram of a biological neuron

The operation is essentially then, one of thresholding. It could be said that the inputs are summed, thresholded and an output produced according to the thresholding function.

1.3.2 The Computational Neuron

The model used to create a computational neuron is the simplistic one outlined above. The neuron used in artificial neural networks has inputs, one output and an element of processing carried out in a cell body.

In the artificial models, the connections between the output of one node and the input of the next node are said to be "weighted interconnections". That is to say, that the connection has a weighting associated with it. Any signal travelling through the interconnection is then modified by the value of the weight. The modified inputs arriving at a node are then summed and a function applied to the total to derive the output for the node.

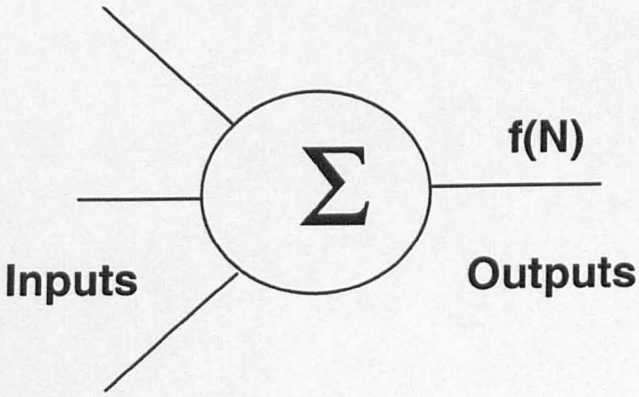


Figure 1.2 The computational neuron

Functions applied to the net input vary. Some simply output the net input thus having a one to one mapping; others use a threshold step function such as,

$f(N) = 1$, if $N \geq \theta$

$f(N) = 0$, if $N < \theta$

More commonly used is the sigmoid function (equation 1.1),

$$f(N) = \frac{1}{1+\exp(-N)} \tag{1.1}$$

1.3.3 The Network

The processing power of these artificial neurons is not in their individual strength, but their collective ability. By forming a group of neurons into a relatively simple network, an array of mathematical functions can be mapped. A typical network will have three layers (as shown in figure 1.3). The first layer carries out no processing but serves as a distribution mechanism. Most researchers refer to a network by its number of processing layers, thus the network in figure 1.3 would be a two layer network.

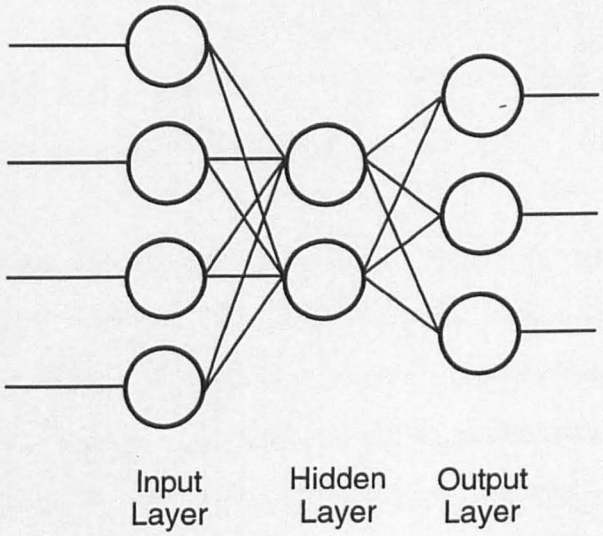


Figure 1.3 A topology for a two layer network

The network can be thought of as a system. It has inputs and one or more outputs. Presenting a vector to the input starts a propagation wave across the network. The input layer distributes the vector through the weighted connections to the hidden layer. The hidden layer calculates its net inputs and applies the selected activation function to produce an output vector, which is in turn fed through the second set of weighted interconnections to the output layer nodes. Again, net inputs are calculated, activation functions applied and an output vector produced. This vector is, however, the network output.

The power of the network is not just in its physical interconnection strategy, but in the values of the weights. With the correct weight values, the network can map mathematical functions, e.g. the network could map the logical AND function. Networks are often trained as classifiers. A historically significant example was the use of a network to map the exclusive OR function (originally significant because early attempts could not compute this function [10]). Here the network is classifying the input patterns into one of two possible output states.

The procedure for choosing the weights is therefore of primary importance. The derivation of the weight values for a network is specific to the problem in hand. To this end, all weight derivation techniques require the user to have examples of the problem which can be used to "teach" the network the classification or mapping. Many different approaches exist and are known as "learning rules". By far the most commonly used is a gradient descent technique known as Back Propagation [10]. Under this paradigm, example vectors from a data set are repeatedly presented to a network and the difference between the desired output and the actual output calculated. This difference is then propagated backwards through the network and

used to modify the weight values. With repeated presentation, the error being produced is gradually minimised (thus the name gradient descent).
Transfer functions and learning rules will be discussed in more detail in chapter 2.

1.3.4 A Brief History

To place this thesis in a historical perspective, a brief history of the research carried out in this field is now given. The aim is not to produce an exhaustive reference list of dates and people, but just to give a flavour of the development of the field of artificial neural networks.

Neural computing is not as new as might be expected. The invention of the first artificial neural network is normally cited to work carried out by Warren McCulloch and Walter Pitts [11] in 1943. In their work neurons were very much logic elements each mapping one logic function. McCulloch and Pitts recognised that the interconnection of these individual elements to form a network increased the possible computational power. The idea of thresholding a net input to a neuron was also introduced by them; however their networks are mostly used as logic circuits [12]. The learning law most commonly cited as being the first was designed by Donald Hebb in 1949 [13]. Simply put, he stated that if two neurons are firing simultaneously then the strength of the connection between them is increased. This statement was used as a base for others to carry out computer simulations [14]. Further work has also added to this rule so that connections are strengthened between two nodes that are both not firing.

The next major development was introduced by several researchers [15,16,18,10]; a concept known as "perceptrons". A class of artificial neural networks, this typically consisted of an input layer connected by adjustable weighted interconnections to associator neurons. The learning rule used iterative weight adjustment to converge to the correct weight values. It could be proven that if weight values existed to solve the problem, the learning rule would converge to them. This triumph led to many exaggerated claims being circulated but was short lived since Minsky and Papert(1969) showed that the networks were considerably restrained in what they could actually learn [10].

In a similar vein to the perceptron learning rule was the invention by Bernard Widrow and Marcian Hoff in 1960 [19] of The Delta Rule (also know as least means squares). Subtle differences between the perceptron learning rule and the delta rule gave improved generalisation performance, generalisation being the ability to correctly classify or respond to previously unseen input vectors.
During the 1970s and 80s several researchers [20,21,22] independently discovered a technique for propagating error backwards through multiple layer perceptrons. This technique became known as Back Propagation and was publicised by David Rumelhart et al. in 1986 [23].

Many other researchers have developed neural computing paradigms. Kohonen's work spanning the 1970s and 80s has developed the theory of self-organising maps and associative memories [24]. Self-organising maps are covered in chapter 2. Stephen Grossberg has carried out much work on the biological aspects of neural networks. Collaboration with Gail Carpenter led to the development of Adaptive Resonance Theory [25]. This technique is also looked at in more detail in chapter 2.

Work carried out by Fukushima [26] on character recognition has led to the development of a dedicated network structure called the neocognitron.
For a good introduction to this field Lippmann [27] should be consulted. In compiling this historical perspective information was taken from Fausett [28].

Put into context it can be seen that neural computing has been in existence for the same length of time as modern computing. It is still young as a research field and there have been many exaggerated and extravagant claims made which have led to a justifiable amount of cynicism and criticism being levelled. The success stories do show though, that neural networks are capable of high quality pattern recognition and can provide valuable insight in many application areas.

1.4 References

[1] Lippmann R; "Neural network classifiers for speech recognition"; The Lincoln Laboratory Journal, 1: 1988, pp. 107-124

[2] White H; "Economic prediction using neural networks: The case of IBM daily stock returns"; IEEE International conference on neural networks II, 1988, pp. 451-458

[3] Hines E L, Gardner J W and Fekadu A A; "Genetic algorithm design of neural net based electronic nose"; International conference on neural networks and genetic algorithms, University of Innsbruck, Austria, 13-16 Apr, 1993.

[4] Kechriotis G and Manolakos ESL; "Training fully recurrent neural networks on a ring transputer array"; Na: Northeastern Univ, Ctr Commun & Digital Signal Proc, Dept. Elect. & Comp. Engn, Boston, Ma, 02115, Microprocessors And Microsystems 1994 Vol.18 No.1 pp. 5-11.

[5] Graf H P, Jackel L D, Howard R E, Straughn B, Denker J S, Hubbard W, Tennant D M and Schwartz D; "VLSI implementation of a neural network memory with several hundred neurons"; in Denker J S (Ed.) AIP Conference Proceedings 151, Neural Networks for Computing, Snowbird Utah, AIP, 1986.

[6] Surkan A J; "Applications of neural networks to classification of binary profiles derived from individual interviews"; IEEE International Conference on Neural Networks II: 1988, pp. 467-472.

[7] Collins J M and Clark M R; "An application of the theory of neural computation to the prediction of workplace behaviour"; Personnel Psychology, 1993, vol 46, ISS 3, pp. 503-524.

[8] Ferner-Salvans P; "An epidemiological approach to computerised medical diagnosis"; Computers in Biology And Medicine, 1990, vol 20, ISS 6, pp. 433-443.

[9] Grossberg S; "The Adaptive Brain 1: Cognition, Learning,Reinforcement, and Rhythm, and The Adaptive Brain II: Vision, Speech, Language, and Motor Control"; Elsevier/ North-Holland, Amsterdam (1986).

[10] Minsky M and Papert S; "Perceptrons: An Introduction to computational geometry"; MIT PRESS, Cambidge, MA, 1969.

[11] McCulloch W S and Pitts W; "A logical calculus of the ideas immanet in nervous activity"; Bullitin of Mathematical Biophysics, vol 5, pp.115-133 1943.

[12] Anderson J A and Rosenfeild E, Neurocomputing; "Foundations of research"; Cambridge, MA; MIT Press, 1988.

[13] Hebb D O; "The Organisation of Behaviour"; New York; John Wiley and Sons; 1949.

[14] Rochester N, Holland J H, Haibit H and Duda W L; "Tests on a cell assembly theory of the action of the brain, using a large digital computer"; IRE Transactions On Information Theory; IT-2, 1956, pp. 80-93.

[15] Block H D; "The perceptron a model for brain functioning"; Reviews of Modern Physics; 1962, vol 34, pp. 123-135, 1962.

[16] Rosenblatt F; "The perceptron, a probabilistic model for information storage and organization in the brain"; Psychological Review, 1958, vol 65, pp.386-408.

[17] Rosenblatt F; "Two theorems of statistical separability in the perceptron; Mechanization of thought processes; Proceedings of a symposium held at the national physics laboratory"; November 1958; London: HM Stationery Office, 1959 pp. 421-456.

[18] Rosenblatt F; "Principles of Neurodynamics"; New York; Spartan, 1962.

[19] Widrow B and Hoff M E; Adaptive switching circuits; IRE WESCON convention records, 1960, part 4, pp. 96-104.

[20] Werbos P; Beyond regression, New tools for prediction and analysis in the behavioral sciences (Ph.D. thesis); Cambridge, MA: Harvard U. Committee on Applied Mathematics, 1974.

[21] Parker D; "Learning Logic"; Technical report TR-87, Cambridge , MA: Centre for computational research economics and management science, MIT, 1985.

[22] Le Cun Y; "Learning process in an assymetric threshold network"; In E. Bienenstock, F. Fogelman-Souli, & G. Weisbuch, eds. Disordered Systems and Biological Organization, NATO ASI Series, F20, Berlin: Springer-Verlag, 1986.

[23] Rumelhart D E, Hinton G E and Williams R J; "Learning Internal Representations by Error Propagation"; In (Rumelhart D E, McClelland J L) Parallel Distributed Processing Vol 1: The MIT Press: 1986, pp.318-362.

[24] Kohonen T; "Self Organisation and Associative Memory"; Third Edition: Springer-Verlag: 1989.

[25] Carpenter G A, Grossberg S; "The ART of adaptive pattern recognition by a self-organising neural network"; IEEE Computer Magazine; March 1988; pp. 77-88.

[26] Fukushima K; "Neocognitron: A hierarchical neural network model capable of visual pattern recognition"; Neural Networks, 1988, 1(2), pp. 119-130.

[27] Lippmann R P; "An introduction to neural computing"; IEEE ASSP Magazine April 1987 pp. 4-22.

[28] Fausett L: "Fundamentals of neural networks"; Prentice Hall Inc, New Jersey, 1994, pp.22-26.

Chapter 2
Neural Networks

2.1 Introduction

With the increase in availability of computer power, many new areas of research have developed. Virtual worlds are created inside computer memory and explored using powerful interfaces to stimulate and fool the mind. The mass storage of data in management information systems is now starting to shape the product life cycle in the corporate arena. The ability to process large quantities of information to obtain useful insight has become big business with powerful database tools being widely available. Mathematical modelling has also undergone a revolution; engineering systems are now designed and tested within a computer. In the engineering industry computers continue to take over repetitive tasks, increasing accuracy and speeding up production lines. Computers are widely used to monitor processes and collect data and more and more emphasis is being placed upon the ability of hardware and software to process data and produce information. Engineers want sharper object recognition, financiers want more accurate trend prediction and managers want further insight into the performance of their companies.

Sprouting from early attempts to model small areas of the human brain, research in the field of artificial neural networks has grown massively over the last decade. Neural networks are a mathematical model composed of three elements, a structure relating to a simplified brain cell (neuron) containing a node (cell body) and a weighted interconnection (axon, dendron, etc.). The types of networks being simulated vary considerably, but fall into two main categories; supervised and unsupervised. In the supervised scenario, the network is trained to recognise patterns by repeated presentation of examples and correction of the output produced. In the

unsupervised scenario, networks are only presented with the input patterns and are left to "cluster" them based on whichever mathematical paradigm the particular technique involves.

2.2 Supervised techniques

2.2.1 Classification

Classification could be quite neatly described as the "art of pigeon holing". The potential use for automated systems that can classify objects into pre-defined classes is enormous and impinges on virtually every aspect of life. An "object" can be a physical commodity such as toothed cogs or helicopters or a more abstract concept such as an odour. The production of an automated classification system involves producing a "black box" which when presented with an input representing a particular object, gives an output indicating the quality about that object that the user desires to know. In the example of the helicopter, a system could in theory be produced that when shown a picture of a such a vehicle, will inform the user of the type of helicopter in the picture. Obviously this process can be as simple or as complex as the user allows. If all the pictures presented show helicopters only in one orientation, the problem is much simpler than if the orientation is allowed to vary. Further complications can be added by the nature of the backgrounds in the pictures, and even whether the pictures were taken on sunny days or not. In some applications, classification can be a relatively simple task requiring nothing more than a thresholding system to distinguish between two commodities, but there are a vast number of applications in which such a system is not sufficiently complex. When

relationships between classes become non-linear (not separable by a linear threshold) and dimensionality in the data increases, more complex systems are necessary. Look-up tables are an obvious choice if the number of possible variations within a class is limited, but in many cases this is not viable. Systems are therefore needed that can classify accurately in complex non-linear, noisy environments and this is the area in which artificial neural networks trained in a supervised manner have proven successful. There has been some discussion as to whether the techniques used in this field are new. It would certainly seem to be true that statistical methods of analysis are in some cases mathematically identical to their artificial neural network counterparts [1], but what would also seem to be true is that networks provide a new approach to viewing this discipline.

2.2.2 Paradigms and Architectures

It is useful at this stage to distinguish between two concepts that are easily confused in this field. These are network architecture and learning paradigm. The architecture of a network is the physical structure in which the nodes are interconnected and the mathematical operation of individual nodes. Some would further separate, dissecting the physical interconnection from the mathematical operations. The learning paradigm is the algorithm that is used to attempt to derive optimum values for certain parameters in the architecture. Normally it is the weights which have their values derived during learning (see section 2.2.3).

2.2.3 The network Learning Paradigm

The most prolific learning algorithm used to date is known as "the generalised delta rule" [2]. This learning algorithm is one in a family of "back propagation algorithms", which in turn is a subset of the family of supervised algorithms. The other main genealogy of algorithms is known as unsupervised (and are less widely used).

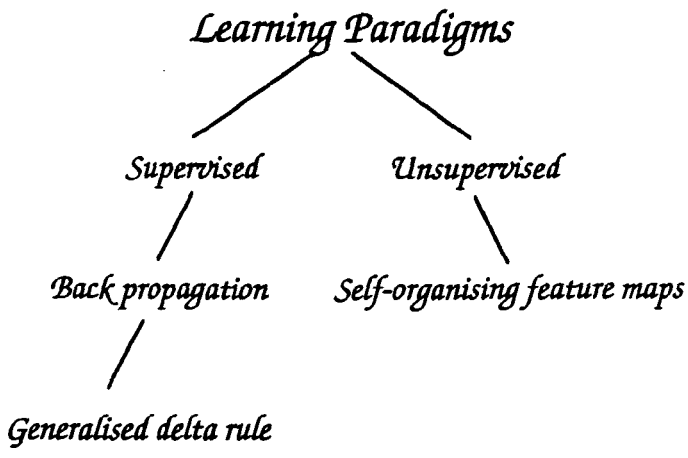


Figure 2.1 Network learning paradigm genealogy

The distinction between these two main groups is as follows. When using a supervised algorithm the data used in the learning process must have two distinct components, the first part being the input pattern vectors; and associated with each one must be the second part, the desired output pattern vector. The generalised delta rule proceeds broadly as follows. An input pattern is presented and the output calculated by propagating that pattern forward through the network (see chapter 1). The output produced is then compared with the desired output and the error vector

generated used to modify internal parameters of the network (often referred to as the "weights"). Each input pattern is presented in turn and this procedure carried out; this is known as "training" the network. The data set is cycled through until the overall error being produced declines to a predetermined level set by the user. There is a fair amount of art involved in stopping training at a suitable point, so that the network will have the capability to generalise onto data previously unseen. The next stage is obviously to test the network by presenting new data and monitoring the network's accuracy.

In the unsupervised family of paradigms, the data being used contains only one element, that being the input pattern vectors. The process of presentation is much simpler than that of the supervised paradigms as the input vectors are presented in turn. The network carries out calculations and modifies weight values purely on the basis of what it sees at its input nodes and has stored in its interconnections. Unsupervised paradigms are used mainly for exploratory data analysis, investigating the internal nature of the data presented. Supervised techniques are mainly used for building models in cases where all the training vectors have a desired output class associated with them. This makes supervised networks good for building classifying systems, but unusable in situations where there is no knowledge of the structure of the data.

2.2.4 Network Architecture

The architecture of a network is its physical structure. This consists of two main elements, nodes (or neurons) and weighted interconnections. A network architecture

is referred to by the number of layers of neurons, some only consider layers that carry out active processing. Figure 2.2 below shows a two layer network topology. The interconnections are said to be weighted. This means that any value being presented at one end of a connection is modified by the weight value as it passes through.

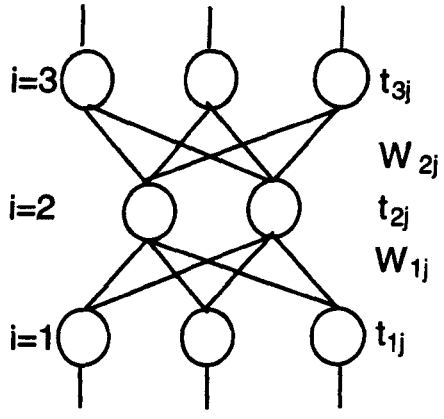


Figure 2.3 A Two Layer Network

Where t represents the output value at node j, and W represents the weight vector for layer i. Each node performs a calculation involving each of its input weight values combined with the associated output values from the nodes in the previous layer. Transfer functions used for nodes vary; the most commonly used methods all involve calculating a net input from the sum of the products of the weight values with the output values from the previous layer (equation 2.1).

$$p_{i,j} = \sum_j t_{i-1,j} w_{i-1,j} \tag{2.1}$$

Where p is the net input. A function is normally applied to the net input before it is presented to the next layer. Many variations exist, but the main function is shown in equation 2.2, (i.e. the sigmoid or logistic function given by equation 1.1).

$$f(p_{i,j}) = \frac{1}{1+e^{-p_{i,j}}} \tag{2.2}$$

Where $f(p)$ is the transfer function for node (i,j). More could be said on this subject and on supervised paradigms in general but the main theme of this work is unsupervised learning. The reader is referred to Freeman/ Skapura [3] who give a fuller account.

2.2.5 Training and usage

A supervised network is said to undergo a "training" or "learning" process. This involves applying the learning paradigm to the network architecture. Data will normally be supplied specifically for this process and ideally would contain both good and bad examples of each class to be learnt. Each pattern vector supplied for the training phase is presented in turn. The data is propagated through the network from layer to layer along the weighted interconnections until an output is produced. This output is compared with the target output for the given vector supplied in the training data and the error between the two figures is used to modify the weight values of the interconnections. The exact nature of the method of modification depends upon the learning rule being used. The weight values for the network converge to reach an optimal value where upon the presentation of any of the training vectors will produce output acceptably close to the target output. The skill of network training is to stop the learning phase at a point at which the values being

produced are acceptable but whilst the presentation of previously unseen vectors still produces a reasonable response. This is the concept of a network being able to generalise onto data that it has not been trained on, and in the supervised scenario this is normally the main objective. If training is allowed to continue for too long, the network will classify the training examples with a high degree of accuracy but be unable to classify any previously unseen examples. This is known as "Overfit".
A well trained network should be capable of obtaining a high degree of accuracy on previously unseen data. This network would then fulfil the requirements laid out above for a good classifier.

In theory the next step in the production of a complete system is to take the successful architecture and associated weight values for the interconnections and fabricate the concept in hardware, either using custom VLSI techniques or one of the increasing number of specialist neural network VLSI chips. The system can then be associated with whichever pre-processing hardware is necessary and be embedded into its particular application environment. Many papers have been published detailing application areas from image compression [4] to blood-vessel detection in angiograms [5].

Systems of this nature obviously fill a gap in the market place. Notice however, one significant assumption that is made in the production of such a system - that the user knows the classes contained in a data set, how many are present and to which class each training vector belongs. This is true in many applications, but not all. To take a medical example, if a disease is known to be affecting a group of people, but there is no knowledge as to why, exploratory work must be undertaken. Doctors may well send out questionnaires to both people suffering from the illness and people known

to be free from it. From the difference in the questionnaire responses, the doctors would look to pinpoint which conditions were common to the people with the illness that were alien to the unaffected group. The complexity of this task obviously depends on the size of the questionnaire. If there were several hundred questions the task would require hours of complex analysis to identify trends within the data set. A system that would be immensely useful would be one which divided the data set into groups (clusters) based on common qualities in the data, and this is the nature of unsupervised learning rules. It is immediately obvious that an unsupervised system can never achieve the same degree of accuracy as a supervised system on a complex data set since the concept of generalisation does not really apply, but such an exploratory tool could be immensely useful in a number of applications. Statistically these techniques are known as *clustering algorithms*; in the terminology of artificial neural networks they are *unsupervised techniques*.

2.3 Unsupervised techniques

2.3.1 Kohonen's Self-organising feature maps

The most widely used unsupervised paradigm was invented by Kohonen and is entitled "The Self-Organising Feature Map"(SOM) [6]. Application areas from 3-Dimensional Planar-Faced Object Classification [7] to involved medical imaging [8] have been published involving SOMs. The architecture of a SOM consists of two layers as can be seen in figure 2.3.

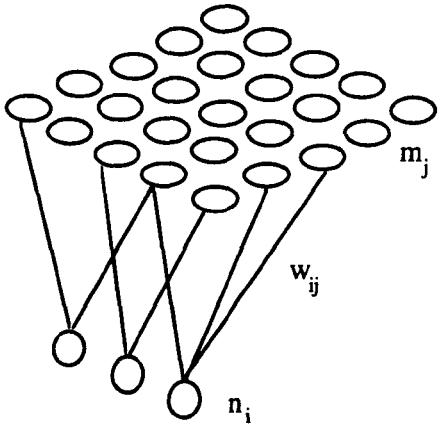


Figure 2.3 The topology of a self-organising map

(All connections not shown for clarity)

Where n_i is the ith node in the input layer, w_{ij} is the weight element and m_j is the node in the map layer. The input layer is a one dimensional vector of nodes and the second (or map) layer is a two dimensional matrix of nodes. Every input node is connected to every output node. In addition to this interconnection, the nodes in the map layer are considered to have neighbours. Two different types of output layer topology are commonly used; in the first the nodes are arranged in a square matrix style and in the second they are arranged in a hexagonal matrix style by offsetting alternate rows.

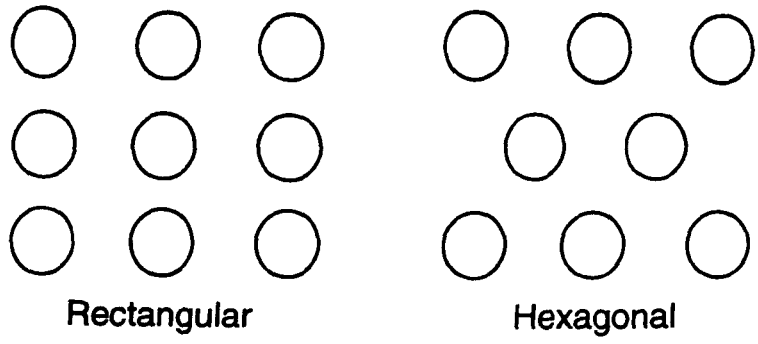


Figure 2.4 Rectangular and hexagonal neighbouring

The interconnections between the input and map layer are weighted and are initialised randomly. The process of operation is then as follows: a vector is presented to the input layer; the Euclidean distance is calculated between the input vector (V) and every weight vector (W_j); and the node in the map layer with the closest weight vector (W_X) to the input vector is activated. Mathematically,

$$|V - W_x| = \min_i |V - W_i| \tag{2.3}$$

where

$$|V - W_x| = \sqrt{\sum_j (v_j - w_{ij})^2} \tag{2.4}$$

The weight vectors of the active node and its neighbours are then modified to bring them closer to the input vector using equation 2.6. The next vector is then presented to the input layer, and this process carried out again. The result of continually

cycling through the data set carrying out this process is to enforce order into the weight space. Kohonen, in his literature [6] has proven that the algorithm orders vectors in one dimension and leaves it to the reader to expand the proof for the 'n' dimensional case. Dayhoff [9] gives a readable account of this proof.

2.3.1.1 The neighbourhood

When modification of the weight vector for the active node takes place, the weight vectors of all associated neighbours are also modified. An important part of this algorithm is how the neighbourhood of a node is defined. In theory it would be sensible for the amount by which weight vectors are modified to be defined by some sort of sinc function centred on the active neuron. In practice a widely used kernel is the square function. All the nodes within a given radius are modified by the same amount. The initial radius is set by the user at run time and it is decreased linearly with each presentation of the training set until it reaches unity using equation 2.5.

$$d = \left[d_o (1 - \frac{t}{T}) \right] \tag{2.5}$$

Where t is the current training iteration, T is the total number of training iterations and d_0 is the initial radius.

2.3.1.2 The Learning rate

The equations that govern the weight updates are shown below (equation 2.6).

$$\Delta w_{ij} = \alpha(v_j - w_{ij})$$

if node i is in the neighbourhood being updated,

else

$$\Delta w_{ij} = 0 \qquad (2.6)$$

and

$$w_{ij}^{new} := w_{ij}^{old} + \Delta w_{ij}$$

Where W is the weight vector, α is the learning rate and V is the input pattern vector.

As can be seen, the function of the learning rate is to determine by how much the weights will be modified. The user sets the initial learning rate at run time by experience and it decreases linearly with each presentation of the training set until it reaches zero using equation (2.7).

$$\alpha_t = \alpha_o(1 - \frac{t}{T}) \qquad (2.7)$$

Where t is the current training iteration and T is the total number of training iterations.

2.3.1.3 The number of iterations

The learning rate and the radius can decrease linearly and converge to their respective values to terminate the training cycle. The answer lies in a third user set parameter, the number of iterations of the training set. Once it is known how many times the training set will be presented, it is a simple task to calculate the values to decrement the learning rate and radius by to fulfil the above criteria.
If pure clustering is being performed and the training set does not contain any information as to the classification of the vectors, the user is faced with a difficult task to decide when to terminate training. The organisation of the map layer can be inspected by presenting the data vectors in turn (without modifying the weight space) and recording which map nodes were activated. A picture can then be built up in three dimensions of how the map has ordered the data. When is the ordering optimal? is a question that is of course extremely difficult to answer as it is not just dependent on mathematics but also on the nature of the application and the interpretation of the output in the real world.

2.3.2 Adaptive resonance theory

2.3.2.1 Introduction

In unsupervised learning, two techniques continue to be discussed and used, self-organising maps being the most widespread. The other is Adaptive Resonance Theory (ART) developed by Grossberg [10]. Artificial neural networks have roots in many subject areas and some obvious affiliations with neuronal biology. ART developed more as an exploratory tool for answering questions about learning

networks than as a tool for solving engineering problems. ART was developed in the search for, among other things, an answer to a problem he described as the "stability/ plasticity" dilemma. Stated briefly, this problem concerns how a network can forget irrelevant input whilst remembering cogent ones, and how a system can retain knowledge previously acquired whilst continuing to learn new information. In essence how can a system remain adaptive (plastic) and yet be stable?

The resultant network is one of the more difficult paradigms to understand. Applications of ART are not as widespread as those of other architectures. Application areas do exist though and range from Group technology [11] in the manufacturing arena to machine vision [12]. Confusion easily arises when attempting to understand ART, as parts of the network have been labelled with psychological terminology. What follows now will be a brief explanation of the topology and of how pattern matching takes place.

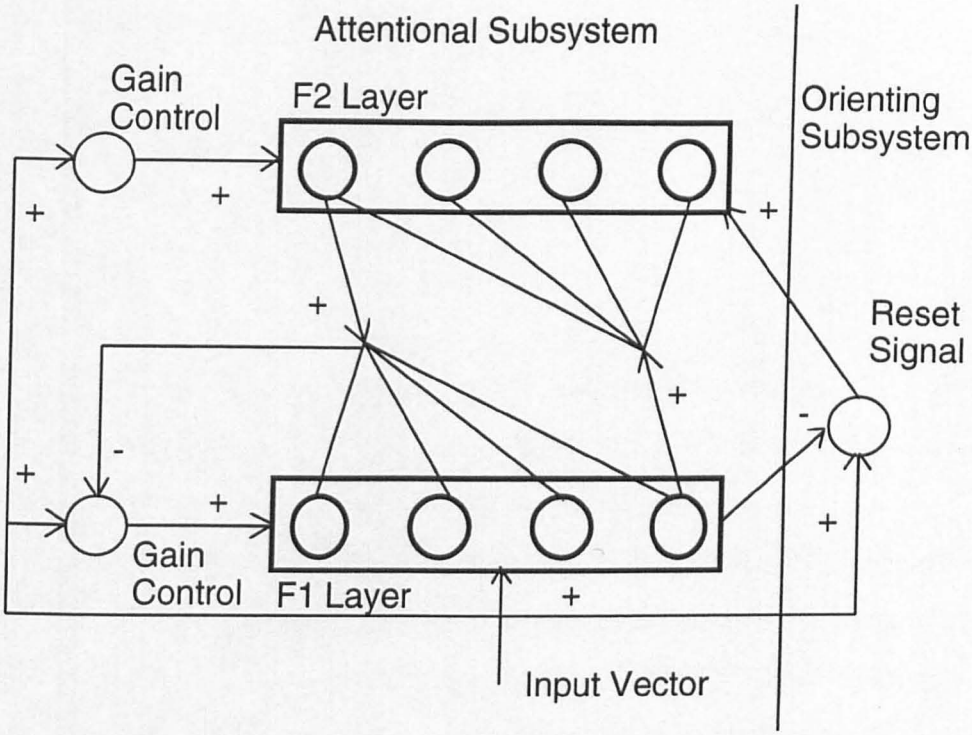


Figure 2.5 The topology of ART (Picture taken from "Neural Networks, Algorithms, Applications, and Programming Techniques" by Freeman/ Skapura)

ART is essentially a two layer network. Every node in the input layer F1 is connected to every node in the second layer F2 and vice versa. Together with a gain control element, these layers are labelled as the "attentional subsystem". The other major component of the architecture is labelled the "orienting subsystem". This is comprised of an element for providing reset signals to the second layer F2.

2.3.2.2 The Layers F1 and F2

The nodes in F1 and F2 obey the same equation (2.8).

$$\varepsilon \frac{dx_k}{dt} = -x_k + (1 - Ax_k)J_k^+ - (B + Cx_k)J_k^- \tag{2.8}$$

The parameter ε is a timing parameter. As will be explained in section 2.3.2.4, the activity between F1 and F2 has to occur at a higher speed than the time that is taken for a weight value to be modified. ε is used to control this.
J_k^+ is the total excitory input to node v_k. Likewise J_k^- is the total inhibitory input. A, B and C are bounding parameters set to keep the calculations within a fixed and manageable range. All parameters are non-negative. Nodes in F1 are denoted v_i. nodes in F2 are denoted v_j. where $i = 1,2, ..., M$ and $j = M+1, M+2, ... , N$. Thus equation 2.8 becomes,

$$\varepsilon \frac{dx_i}{dt} = -x_i + (1 - A_1 x_i)J_i^+ - (B_1 + C_1 x_i)J_i^- \tag{2.9}$$

and

$$\varepsilon \frac{dx_j}{dt} = -x_j + (1 - A_2 x_j)J_j^+ - (B_2 + C_2 x_j)J_j^- \tag{2.10}$$

The inputs to nodes are calculated in the usual manner, by summing the product of the weight vectors with the pattern vectors. What is unusual is that each node has inputs from the gain control unit to take into account. The nodes in the F1 layer also have the inputs from the original pattern vector. The inputs are gathered into excitory

(those positively labelled in figure 2.5) and inhibitory (those negatively labelled in figure 2.5). All inhibitory inputs are summed and all excitory inputs are summed. They are then introduced into the equations given above as previously explained. The output of a node in the F1 layer can be assumed to equal unity if $x > 0$, or zero if this is not true. This is also true in part for the nodes in F2. The second layer F2 is a "winner takes all" layer. This means that one and only one node can be active. To achieve this, the nodes are interconnected laterally and "compete" with each other to be active. This is known as a "competitive layer". The node with the highest activation value has its output set to unity; all other node outputs in F2 are forced to zero.

2.3.2.3 Gain Control and the Orienting Subsystem

The output of the gain control unit is unity if and only if there is an input vector present and F2 is not currently processing, otherwise the output is zero.
To understand the operation of the orienting subsystem, let $|I|$ be the width of the input vector. If the weights on all the input connections to the system are P, the total excitory input becomes $P|I|$. If the number of active connections from the output of F1 to the system is denoted $|X|$ and the associated weights are equal to Q, the total inhibitory input becomes $Q|X|$. The output of the system only switches when its input becomes non-zero.

$$P|I| - Q|X| > 0 \qquad \text{or} \qquad \frac{P}{Q} > \frac{|X|}{|I|}$$

P/Q is referred to as the vigilance parameter (ρ) and is set by the user at run time.

So, system reset is initiated when

$$\rho > \frac{|X|}{|I|}$$

2.3.2.4 Recall and learning in ART1

ART learns "on-line". With back propagation there are two distinct phases, learning and then recall or testing. This distinction does not exist with ART. When a vector is presented, a cycle is entered into. At the termination of this cycle, ART will have either "learnt" the new vector (often referred to as a pattern in the context of ART), or it will have strengthened its knowledge of the fact that it has seen that vector previously.

When an input pattern is presented to the F1 layer, it is also presented to the gain control unit and the orienting subsystem reset element, both in an excitatory capacity (as shown in figure 2.5).

Processing governed by equation 2.9 is carried out in the F1 layer resulting in an output pattern being produced . This output pattern is presented both to the orienting subsystem as an inhibitory connection and to the nodes on the second layer F2 via the bottom-up weighted connections. Learning in ART is continuous and modification of a weight vector takes place every time a vector passes through its connection. As was explained for equation 2.8, there is a an apparent delay in this learning; modifications are carried out much more slowly than the activity that occurs between the F1 and F2 layers. As the excitatory pattern and the inhibitory pattern presented to the orienting subsystem are identical, no reset signal is produced. The F2 layer calculates its net inputs and formulates its output values. As a result of the competition, a pattern of activity appears in the F2 layer with one node becoming the overall winner. As with the F1 layer an excitatory gain signal is applied to each node. The same signal is applied to all the nodes in the same layer. This is known as a "non-specific" signal. The output values from F2 are presented both to the gain

control unit as an inhibitory connection and to the nodes in the F1 layer as an input via the top down weight vectors. The gain control having received an inhibitory signal becomes inactive. At this stage in the proceedings a rule known as the **2/3 rule** comes into play. This rule states that, *a node can be active if and only if, two of the three possible inputs to it are active*. As the gain control unit is inactive and therefore not supplying the non-specific input signal, the 2/3 rule dictates that only the nodes with both active input from the pattern presented from the F2 layer and active input from the original pattern vector will in turn be active. Obviously the pattern of activity at F1 becomes the intersection of the original pattern vector with the vector presented from F2. As this new pattern of activity leads to a new output pattern from F1, the vectors being presented to the Orienting Subsystem now differ. This mismatch causes a non-specific reset signal to be sent to the nodes in the F2 layer as an excitatory connection. The reaction of the nodes in the F2 layer depends upon their current state. Inactive nodes are not affected. Active nodes are reset to an inactive state and this is sustained for a predetermined period of time, set to prevent the reset nodes from winning the next pattern matching cycle.

With the lack of output from F2, the inhibition of the gain control unit is removed, resulting in the original pattern vector being presented afresh to the F1 layer. The above cycle is then moved through again with the proviso that the nodes placed inactive by the orienting subsystem remain inactive and are thus incapable of wining the competition. This cycle continues until either a match is found, in which case no reset signal is generated, or until all the nodes in F2 have been reset, in which case an uncommitted node is assigned the pattern and the weight vectors begin to learn the pattern. If a match is found or a node is committed to a pattern, the connections remain active to allow the network to settle into a resonant state. In this state the

pattern of activity between the F1 and F2 layer oscillates to and fro; as this happens the weights are modified to aid the retention of the pattern vector presented using the following equations. The bottom-up weights are modified on V_j only.

$$z_{ji} = \frac{L}{L-1+|X|}$$

if V_i is active else the weight value is set to zero. z_{ji} is the weight value. The top down weights from V_j are modified by setting them to unity if V_i is active; otherwise they are set to zero.

Two terms are used to describe various features of ART's memory. The patterns of activity that develop across the nodes are described as "short term memory traces" and the weights between the two layers are described as "long term memory traces". Several variations of ART now exist, the two main ones being ART1 and ART2. The main distinction is that ART1 only takes binary input vectors, where as ART2 takes real valued inputs.

A mathematical account of the operation of ART can be found in Carpenter and Grossberg [12] or Freeman/ Skapura [13]. In the explanation of ART given above, information from both these sources has been adapted.

2.4 References

[1] Sarle W S; "Neural Networks & Statistical Models"; Proc. 19th SAS Users Int. Conf. April 1994.

[2] Rumelhart D E, Hinton G E and Williams R J; "Learning internal representations by error propagation"; In (Rumelhart D E, McClelland J L) Parallel Distributed Processing Vol 1: The MIT Press, Cambridge, MA, 1986, pp.318-362.

[3] Freeman J and Skapura D; "Neural Networks, Algorithms, Applications and Programming Techniques"; Addison Wesley, 1992, pp. 17-30.

[4] Qiu G, Terrell T and Varley M; "Improved image compression using backpropogation networks"; Proc. Neural Network Applications and Tools, Sept, 1993, IEEE Press, pp.73-81.

[5] Nekovei R and Sun Y;"Backpropagation Network And Its Configuration For Blood-Vessel Detection In Angiograms"; Univ Rhode Isl, Remote Sensing Lab, Bay Campus, Narragansett, Ri, 02882 Univ Rhode Isl, Dept. Elect. Engn, Kingston, Ri, 02881, IEEE Transactions On Neural Networks 1995 Vol.6 No.1 pp.64-72.

[6] Kohonen T: "Self-organisation and AssociativeMemory"; Third Edition: Springer-Verlag: 1989.

[7] Delbimbo A, Landi L and Santini S;"3-Dimensional Planar-Faced Object Classification With Kohonen Maps"; Univ Florence, Fac Ingn, Dipartimento Sistemi & Informat, Via S Marta 3, I-50139 Florence, Italy, Optical Engineering 1993 Vol.32 No.6 pp.1222-1234.

[8] Manhaeghe C, Lemahieu I, Vogelaers D and Colardyn F; "Automatic Initial Estimation Of The Left-Ventricular Myocardial Midwall In Emission Tomograms Using Kohonen Maps"; State Univ Ghent, Dept Electr & Informat Syst, B-9000 Ghent, Belgium State Univ Ghent Hosp, Dept Intens Care, B-9000 Ghent, Belgium, IEEE Transactions On Pattern Analysis And Machine Intelligence 1994, Vol.16 No.3 pp.259-266.

[9] Dayhoff J; "Neural Network Architectures; An introduction:" Van Nostrand Reinhold: 1990, pp. 163-191.

[10] Carpenter G A and Grossberg S; "The ART of adaptive pattern recognition by a self-organising neural network"; IEEE Computer Magazine; March 1988; pp. 77-88.

[11] Moon Y and Kao Y; "Automatic generation of group technology families during the part classification process"; Adv. Manuf. Technol., 1993, 8, pp. 160-166.

[12] Kolodzy P; "Multidimensional machine vision using neural networks"; Proc. IEEE First Int. Conf. on Neural Networks, San Diego, CA, June 1987, pp. II-747-II-758.

[13] Carpenter G and Grossberg S; "A massively parallel architecture for a self-organising neural pattern recognition machine"; Computer Vision, Academic Press, 1987, 37, pp. 54-115.

[14] Freeman J and Skapura D; "Neural Networks, Algorithms, Applications and Programming Techniques"; Addison Wesley, 1992, pp. 291-339.

Chapter 3

Statistical Clustering Techniques

3.1 Introduction

Analysis of market research data has traditionally been carried out by statistical methods. Magazines, newspapers and news broadcasts are often littered with statistical quotes from surveys carried out. Companies commission interview-based research to discover their market penetration, customer awareness and new product areas. In these situations the researcher is aware of the subject to be investigated or highlighted. A questionnaire is designed specifically to gather the information and present it in a palatable form for analysis. Contingency/ ranking tables and Analysis Of Variance (ANOVA) can then be used (amongst other techniques) to turn the raw data into meaningful information. The analysis tools used in this scenario are tried and tested but although they are reasonably impartial, it cannot be denied that the survey is specifically commissioned to look at a particular aspect. Although this does not necessarily bias the proceedings, the process cannot be said to be truly impartial.

3.2 Clustering Algorithms

Although perhaps less common, another thread exists in research. In a relatively small number of cases, the researcher does not know what they are setting out to prove. There is no hypothesis to be put to the test. To take a medical example, if a researcher realised that an increasing number of people were dying from a previously unrecorded illness where there existed no immediately obvious link between the cases, a possible approach to the problem would be to complete questionnaires for the people who had died, those dying and also for a number of people apparently unafflicted. The questionnaire could cover everything that could possibly be

relevant. What would be needed to process this information would be a technique that could study the inter-relationships within the data set. This category of techniques exist, and is known as clustering algorithms. In general, they make no assumptions about the data other that its dimensionality. The algorithms vary in the amount of information they must be given in order to establish clusters.

3.2.1 Partitioning and Hierarchical Algorithms

Two main varieties of clustering algorithm exist; partitioning and hierarchical. Partitioning techniques require the user to set the number (K) of clusters to be located. The data set will then be split into K clusters, with each vector being present in only one. All the vectors are assigned to at least one cluster. The user must run the algorithm many times with different values to K to locate the best clusters (unless it is known how many should be present).

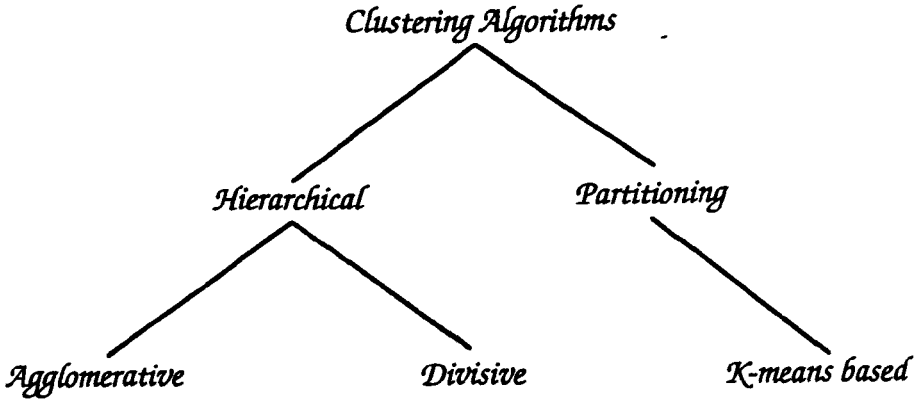


Figure 3.1 Clustering algorithm genealogy

Hierarchical techniques do not require a value for K. Two opposing varieties exist agglomerative and divisive. Divisive techniques start with the data set as one

complete cluster and repeatedly sub-divide (under the supervision of a rule base) until no further sub-division is possible. Agglomerative techniques approach the problem in reverse, starting with each vector assigned to its own cluster. The algorithm then recursively joins clusters until all the vectors are in the same group.

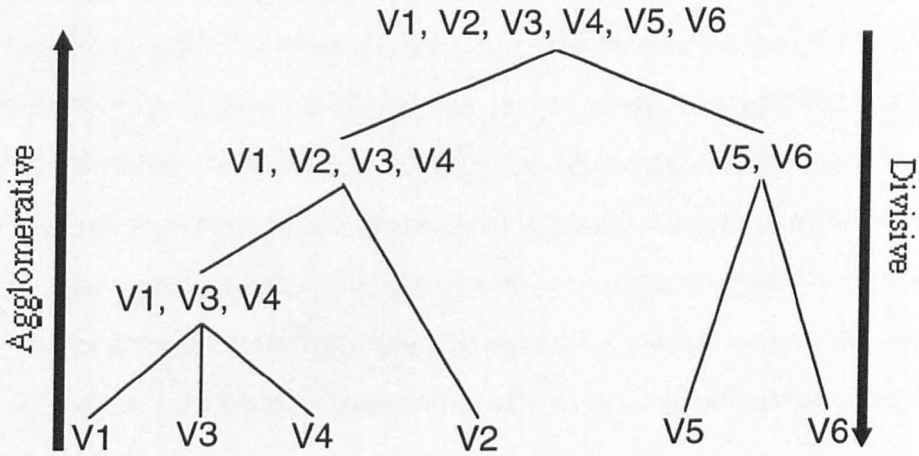


Figure 3.2 The tree structure produced by agglomerative and divisive algorithms

Thus it can be said that these techniques produce a hierarchical output (as shown in figure 3.2). This is often output as a tree diagram. The researcher then faces the dilemma of where to cut the tree.

Variations on these themes exist, and software packages have been developed to carry out clustering. Many clustering algorithms are unsuitable for use on large data sets. Using hierarchical techniques, it is necessary to output the tree diagram produced. Whilst this is reasonable for data sets with a few tens of vectors, the computer file produced can rapidly become unmanageable with a few hundred vectors. The use of random access memory by some techniques is also rather extravagant, and this can prohibit their use on large data sets.

Gower [1] compared three hierarchical techniques, one agglomerative and two divisive. He concludes that for general purpose work the agglomerative technique should be used. It is however stressed that the common ground between the techniques is that they attempt to maximise some measure of inter-cluster distance. Care must be taken to ensure that the measure of distance being used is appropriate to the problem in hand. Milligan [2] carried out a more substantial Monte Carlo study on fifteen clustering algorithms of varying descriptions, using data sets with various degrees of artificially produced noise present. He concludes that hierarchical techniques performed relatively poorly with the introduction of noise to the data sets. K-means techniques exhibited excellent recovery of clusters when seeds were selected using the group average method (the K-means algorithm is explained in section 3.4).

3.3 Clustering binary data

When analysing binary data the researcher is faced with an immediate decision. Binary data can be manipulated using any of the clustering algorithms available for processing data sets containing real integer or floating point data (as the binary space is purely a sub-space of the Euclidean space, see chapter 6). Existing techniques can also be customised for use with binary data. Customised algorithms for use on binary data sets still however rely on the use of similarity/ dissimilarity or distance-based measures. In choosing which of the measures to use, the researcher must know whether the data is symmetric or asymmetric. That is to say, do both of the binary states carry the same weight. Similarity measures exist for both cases (Kauffman/ Rousseeuw give an excellent coverage of this [3]).

3.3.1 Monothetic Analysis

Along with standard techniques which can be customised for binary use, there also exists a class know as monothetic analysis. Algorithms in this class can make specific use of the binary nature of the data. A variable is picked (using a user selected similarity measure), and the data set is then divided into two subsets (from the state of that variable in each vector). A single variable in each of these subsets is then selected and the same process carried out for each cluster. The algorithm continues until no further division of any of the sub-clusters can be made. The selection criteria is to select the variable which has the most similarity with all the other variables. There are various methods for measuring similarity. The recursive, divisive nature of the algorithm places it in the hierarchical group. Kauffman [4] gives an example of a monothetic algorithm for processing binary data.

It is theoretically possible to cluster binary data using both partitioning and hierarchical techniques. However, the size of the data sets being analysed in this research restrict this. Hierarchical algorithms output a tree diagram for the user to inspect and draw conclusions from. It was found in practice, that analysing a tree diagram with more than a few tens of vectors represented, was a physical impossibility. The paper output rapidly became unmanageable, with output threatening to be in the region of 400 sheets for a data set of around one thousand vectors.

3.4 FASTCLUS

3.4.1 Introduction

Placing the restraint of data set size on the algorithms available rapidly reduced the number of options, the only suitable one to emerge being the SAS Institutes FASTCLUS [5]. This algorithm is a variation on the K-means method [6] with some similarity to Harigan's Leader algorithm [7]. This is a partitioning method, relying on the user to supply K, where K is the required maximum number of seeds (and hence clusters). Alternatively, the minimum radius between cluster seeds can be specified. K-means techniques select K cluster seeds and then assign the vectors in the data set to their nearest seed. Nearest, in this case, being computed by Euclidean distance. As vectors are added to a cluster, the seed is recalculated to be the mean vector. When all vectors have been assigned, the process is restarted using the new seeds. The algorithm concludes when the movement in the seed vectors is adequately small. The implementation of this algorithm in the SAS software package has been optimised for large data sets (100 - 100,000 vectors). The output available from the software is also tailored to handle large numbers of vectors. Obviously, the initial seeds selected have a great bearing on the length of

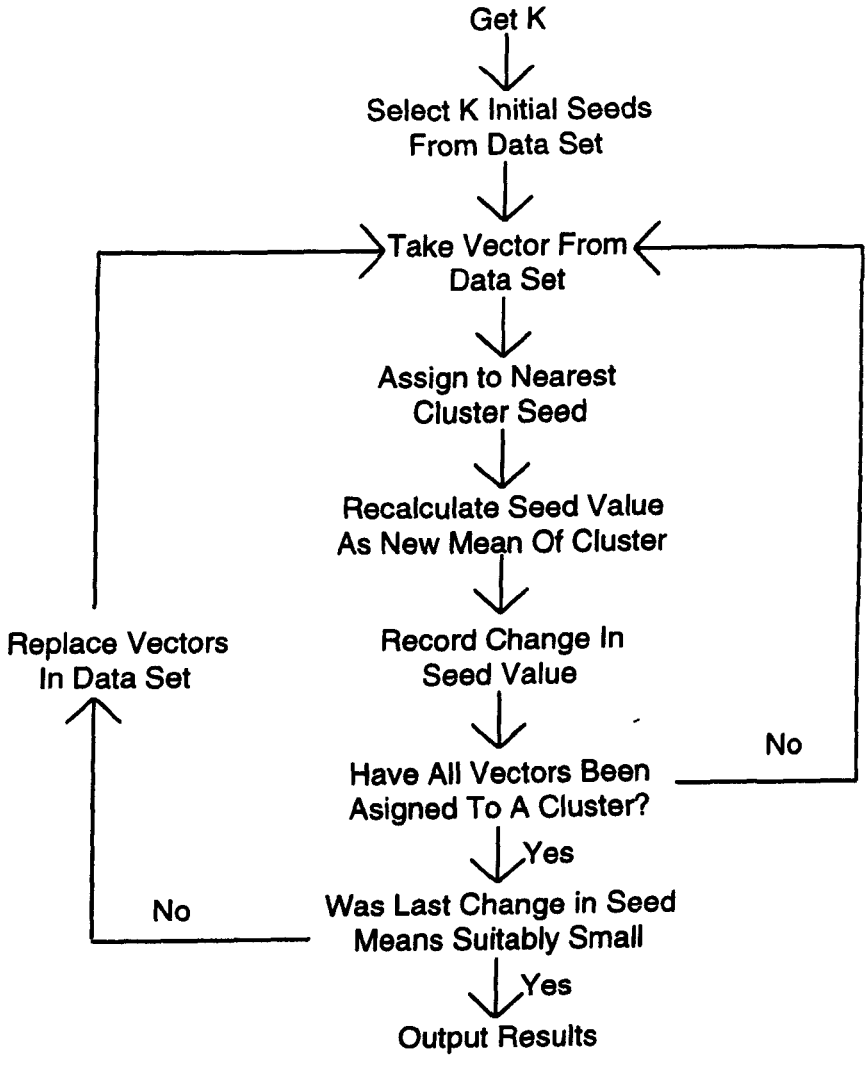


Figure 3.3 Flow chart for K-means

time taken for the algorithm to converge. With large data sets, optimising the seed selection is of great importance. Milligan [2] found that unless robust procedures were used for finding the initial seed the algorithm would fail to recover clusters with any degree of accuracy. The SAS manual states that only a few passes over the data are necessary for convergence.

"The FASTCLUS procedure differs from other nearest centroid sorting methods in the way the initial cluster seeds are selected. The initialisation method of PROC FASTCLUS guarantees that if there exist clusters such that all distances between observations in the same cluster are less than all distances between observations in different clusters, and if you tell PROC FASTCLUS the correct number of clusters to find, then it always finds such a clustering without iterating. Even with clusters that are not as well separated, FASTCLUS usually finds initial seeds that are sufficiently good so that few iterations are required." [5]

3.4.2 Missing Values in FASTCLUS

The data sets under investigation in this thesis are prepared so as not to have any missing values. It is quite possible however that a data set may have, and it is therefore important for an algorithm to cope with this. FASTCLUS has built-in mechanisms for coping with missing values; such a vector is prohibited from becoming a cluster seed. When calculating distances between vectors, the formula is modified to cope with missing data. If a vector contains only missing values (i.e. no data) it is excluded from the analysis. The modified Euclidean distance formula used when a vector has a missing value is as shown in equation 3.1:

$$d = \sqrt{(n/m)\sum_i (x_i - s_i)^2} \tag{3.1}$$

Where,

d = the measured distance.

n = the number of variables.

m = the number of variables with non missing values.

x_i = the value of the ith component of the vector.

s_i = the value of the ith component of the seed.

The summation is only carried out over variables with values.

3.4.3 Seed selection

As has been stated, seed selection is the criteria that most affects speed of computation and thus ability to handle large amounts of data. The rule base used in FASTCLUS is as follows.

- The first vector without missing values becomes the first seed.
- From then on, the vectors are worked through in turn; each vector has the chance to either become a new cluster seed (until maximum number reached), or replace an existing seed.

3.4.3.1 Criteria for becoming a new cluster seed

For a vector to become a new cluster seed it must meet the following criteria:

- The maximum number of seeds must not already exist.
- The vector must contain no missing values.
- The distance between the vector and all previous seeds must be greater than the user-specified minimum radius.

3.4.3.2 Criteria for replacing an existing seed

For a vector to replace an existing cluster seed it must first meet the following criteria:

- The vector must contain no missing values.
- The vector must not have qualified to be an existing seed.

It must then pass either the first or second test outlined below.

First test

- The minimum distance from the vector to any of the other seeds must be greater than the minimum distance between two existing seeds.

If this test is passed the seed to be replaced must be selected. This selection is performed as follows:

- The two closest existing seeds are taken in turn and replaced by the vector.
- In each case, the distance from the "other" seed (i.e. the unreplaced one) is measured to the closest existing seed.

The seed that is replaced is the one with the shortest distance.

Second test

- The vector can replace the closest seed to itself if, the minimum distance to any of the other seeds from the vector is greater than the minimum distance from the seed to be potentially replaced to any of the other seeds i.e. if the vector is further from the others seeds than the subject of the replacement.

If a vector fails both tests, it is passed over in the process for the selection of seeds. The minimum radius option defaults to zero. Not specifying a radius does not mean that the clustering algorithm will not converge. It simply means that longer will be taken to find the seed vectors, as the calculated mean vectors are likely to move considerably during the phase that establishes the seeds.

3.4.4 Output produced by FASTCLUS

FASTCLUS has various options for producing output. Initial and subsequent seeds can be output along with items such as RMS distance between vectors in a cluster. Most importantly, FASTCLUS produces a list with the cluster number that each

vector has been assigned to. Also useful, is a table showing the number of observations assigned to each cluster. Processing the list produced is then a simple task of collecting the vectors from the data set into their clusters and calculate scores for them (see chapter 6).

A brief, but informative description of FASTCLUS and many other software packages can be found in Romesburg [8].

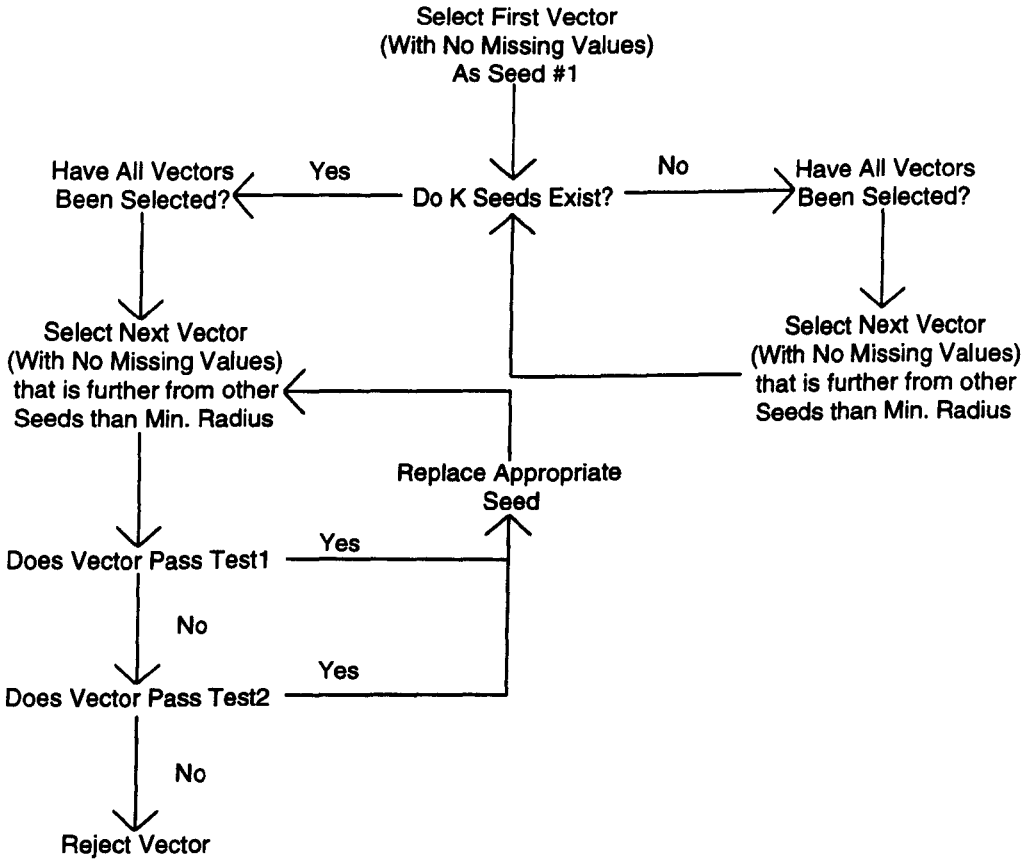


Figure 3.4 Flow Chart for seed selection

3.5 References

[1] Gower J C; "A comparison of some methods of cluster analysis"; Biometrics, Vol 23, 1967, pp. 623-637.

[2] Milligan G W; "An examination of the effect of six types of error perturbation on fifteen clustering algorithms"; Psychometrika, Vol 43, 3, 1980, pp.325-342.

[3] Kauffman L and Rousseeuw P; "Finding groups in data"; Wiley, New York, 1990, pp.22-27.

[4] Kauffman L and Rousseeuw P; "Finding Groups in Data"; Wiley, New York, 1990, pp.280-311.

[5] SAS/ STAT Users Guide, Vol. 1, Ver. 6, Forth Ed; SAS Institute INC, SAS Campus Drive, Cary NC 27513; pp. 823 - 850.

[6] MacQueen J B; "Some methods for classification and analysis of multivariate observations"; Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, pp.281-297.

[7] Hartigan J A; "Clustering Algorithms"; New York, John Wiley& Sons Inc, 1975, pp. 74-87.

[8] Romesburg H; "Cluster Analysis for Researchers"; Lifetime Learning Pub, 1984, pp. 295-296.

Chapter 4
Self-Organising Maps and Memory Arrays

4.1 Introduction

With the stated aim of this research being to locate clusters within a data set that have the maximum degree of commonality it is obvious that the class of neural networks pertinent to this problem is the unsupervised family. The Self-Organising Feature Map (SOM) [1,2] seems a logical starting point in terms of the artificial neural network analysis, the reasons for this being that the only knowledge possessed by the user is the dimensions of the data set. No knowledge is present of the existence of classes within the data, how many or if indeed any at all exist. Self-organising maps present as their output, a two dimensional surface upon which the data set will have been "ordered". This will give insight into the existence of clusters in the data and so training a SOM on the data seems to fulfil the requirements.

4.2 Self-Organising Maps

Having established that the technique is appropriate, the next requirement is to establish what data the model should be trained on. Two possibilities existed; the first being to create idealised data of the dimensionality required, and the second approach being to train the map in the expected manner on the real world data.
The first approach would have the advantage of creating a generic map that could be used in the analysis of further data of a similar nature, where as the second approach held the possibility of producing a map only capable of being used in the one analysis (if the data is unrepresentative of the problem space).
Both approaches were taken and initial analysis was carried out using the first approach.

4.2.1 Training on idealised data

For the purposes of these experiments a data set was created from the survey answers of a group of engineering management personnel in a local company (data supplied by TMCG[1]). Each question form was translated into a vector. The answers to each question were of the form of a five element scale. This was reduced to a binary set to simplify the problem and to increase statistical validity.

To aid interpretation of the final results it was attempted to divide the map surface by training the SOM so that specific areas of the map would represent low scores in certain sections of the questionnaire, with the ground between these zones representing more complex arrangements of low scores.

The forms were comprised of 28 questions in five sections, so the vectors contained 28 elements. In creating the idealised data there is a need to strike a balance between creating an even distribution covering the problem space and the data set being too large to sensibly handle on the computer system. A full data set could be created containing all the binary vectors, but 2^{28} vectors would be rather difficult to handle. For these reasons the idealised data was created by setting all the elements in a section low for a given vector. Table 4.1 shows an example of binary idealised data. It was hoped that training a SOM on data of this nature would create zones on the map surface that would represent low scores in set sections of the data form. Forms that had low scores in multiple sections would fall between zones on the surface, thus allowing an easy interpretation.

[1] The Management Consulting Group Ltd, Coventry.

	Section 1	Section 2	Section 3
Vector 1	000	111	111
Vector 2	111	000	111
Vector 3	111	111	000

Table 4.1 Idealised data for a form of nine questions split into three sections

Having trained the self organising map on the idealised data, the real data was presented and found to cluster on the map surface in groups defined by low overall sectional scores as had been anticipated. Figure 4.1 shows a hypothetical map surface for the data given in table 4.1

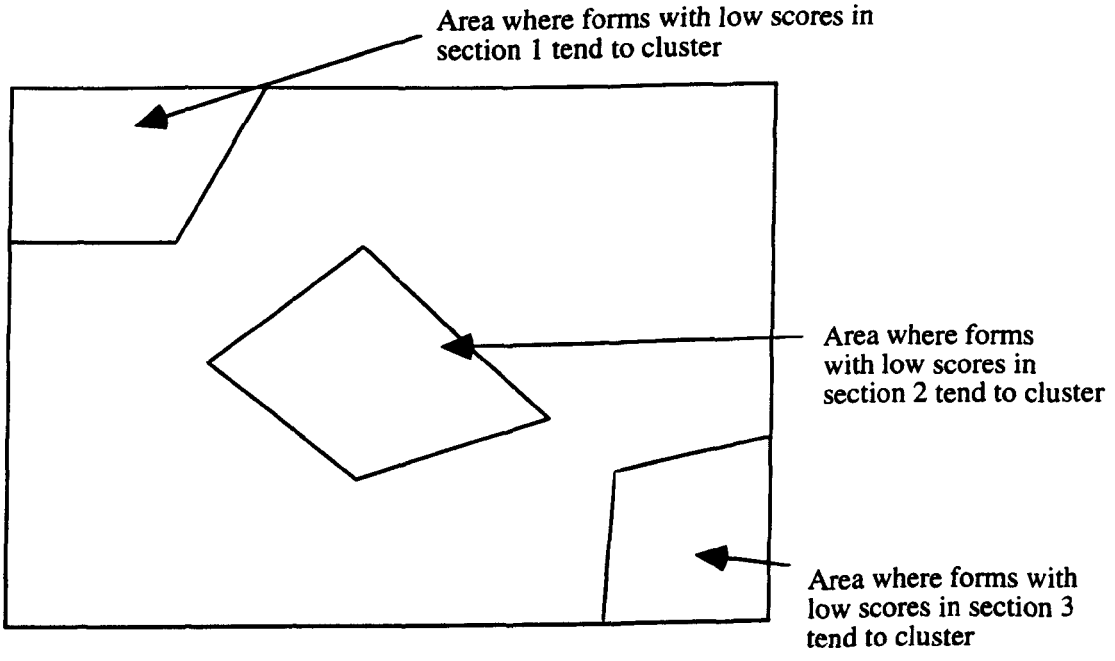


Figure 4.1 A map surface for the data given in table 4.1

In such a situation it is reasonable to assert that the self-organising map had separated the vectors into groups based on common qualities.

The success of this first method was tempered by a major problem, this being the interpretation of the results. The idealised data was generated to train the network on large areas of low score in each section of the vector (engineering survey form). This produced zones on the surface of the map that could be interpreted as low scores in a particular section. The problem was that most of the real data, when presented, resulted in a classification between the zones (as had been anticipated). If only two zones existed there was no problem, but if there were more than two zones, it was impossible to tell which zones contributed to the position of the vector on the map. It was observed that in all the cases investigated the largest peak on the map surface occurred in a position that lent to an easy interpretation. Most of the smaller peaks however, were more open to interpretation.

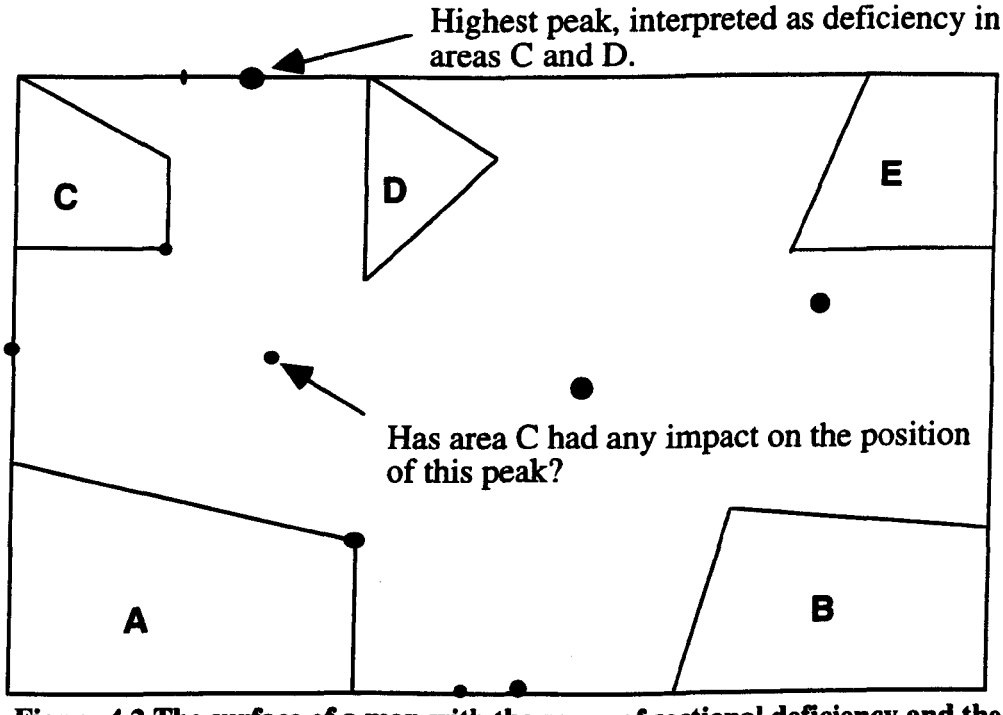


Figure 4.2 The surface of a map with the zones of sectional deficiency and the peaks marked (diameter of dot indicates height of peak).

The difficulty in deciding which zones on the map surface contributed to the position of the vector presented was caused by the fact that the distribution of the map nodes in the weight space followed the distribution of the training data in the input space. As the data is of a high dimensionality it can easily be seen that in order to produce a training set that would evenly distribute the nodes of the map in the data space, an unreasonably large training file would be needed. Paths between sections were traced (by entering artificial data) and found to be less than obvious and so assigning meaning to the position of a presented vector was in most cases impossible without inspecting the map weight vector. The zones were therefore playing no part in the interpretation of the data and there was no advantage had in training on the idealised data.

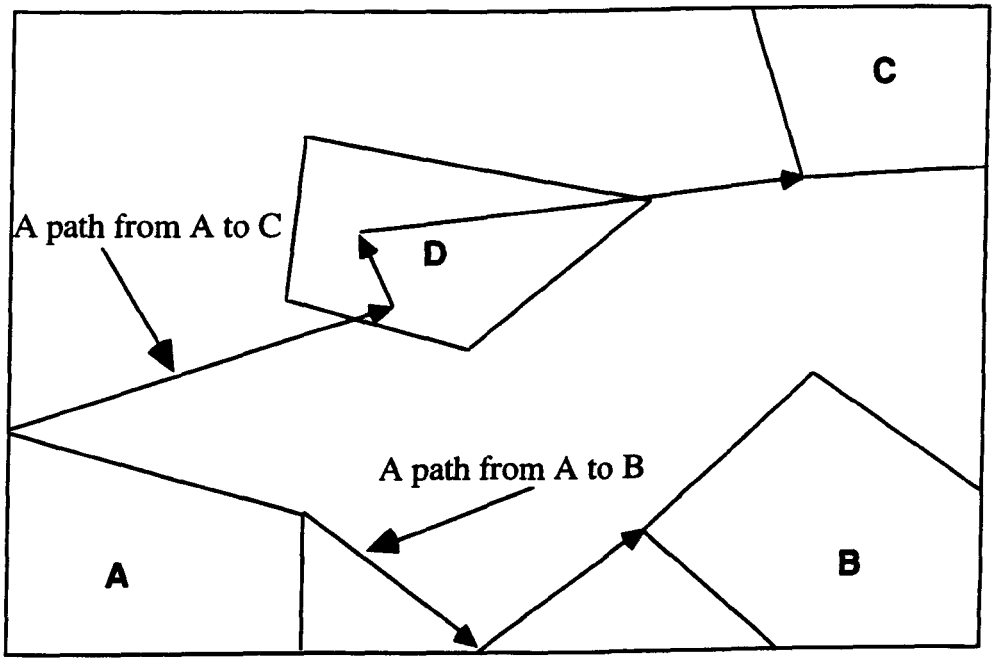


Figure 4.3 A map surface showing the non-linearity of paths between zones due to the fact that the map nodes are not evenly distributed in the data space.

The second approach of training the map on the real world data was then attempted. The map surfaces produced did show peaks, but the same problem was encountered again, in that most vectors fell in positions that lent to a difficult interpretation due to the distribution of the map nodes in the input space.

4.3 The Euclidean Memory Array

Since each element of the input vectors is a member of a small finite set, it seemed possible that weights could be mathematically derived, thus avoiding learning. Some work has been published on statistical methods for the production of weights [4] , but use of this type of technique limits the data to being representative of the possible input space.

In an attempt to overcome all the previous problems a vector quantisation algorithm [3] was proposed. The network consists of three layers; the input and output layers being vectors of summation nodes, and the hidden layer being an array.
The nodes in the input layer sum the values presented for each section of the form (for the purpose of this experiment, the input vector elements were restricted to the binary set {0,5}. This method was adopted in order to reduce the complexity of the problem, and thus make it easier to derive weights). The hidden array cells are fully connected to the input nodes via weighted links. The output nodes are also fully connected to the hidden array cells via weighted links.

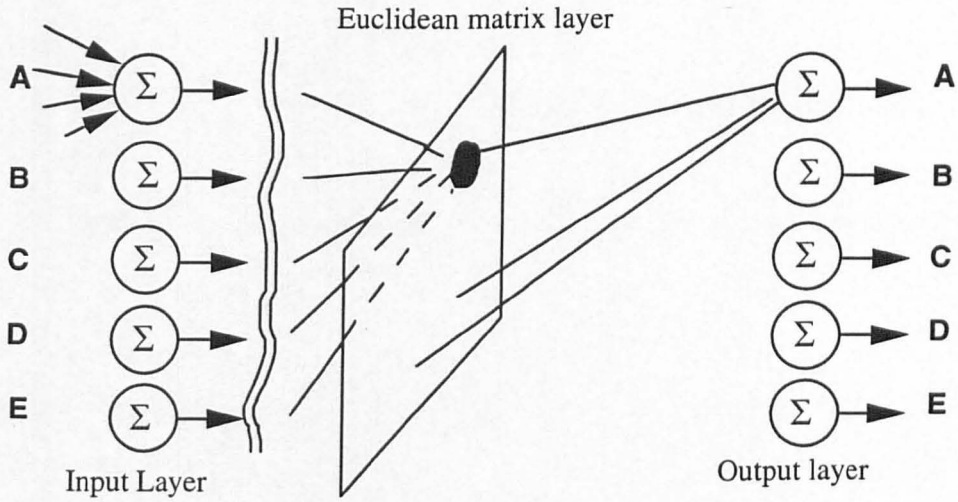


Figure 4.4 The Euclidean memory array

4.3.1 Data Propagation

Data from interview forms is processed by the Euclidean Memory Array (EMA) [5] as follows. The form is presented to the input nodes. The summation of each section is produced, and compared (using Euclidean distance, equation 4.1) to the weight vectors of the hidden layer. The closest match is found, and the corresponding cell has its value incremented (they are all set to zero initially). The values in the hidden layer cells are propagated via weighted connections to the output summation nodes.

In pseudo code:

(taking '*'= multiplication, comments in italics)

for (each hidden node) value = 0

Derive weights *(see section 4.3.2)*

(Data vector presented)

for (each input node) sum inputs

increment hidden node with '**min**(distance(I,W))'

for (each output node) output = **sum**(Weight_Vector * Hidden_Node_Values);

Where I is the output vector from the input nodes; W is the weight vector for the hidden matrix node.

4.3.2 Derivation of the weights

Input to hidden layer weights

To facilitate an easy interpretation of the network outputs it was necessary to have a range of weights that were evenly distributed through the possible input space.
As each input node took as its input the answers to six questions, each from the set {0,5} it is obvious that the maximum output from each node would be '30' (i.e. 6 inputs * 5 max. values). As there are five input nodes, each fully connected to the hidden layer, this obviously gives a possible weight space (input - hidden layer) of {0,0,0,0,0} to {30,30,30,30,30} with each element being a member of the finite set {0,5,10,15,20,25,30}. What was needed was a sensible degree of quantisation to cover the possible input space. Quantising to three discrete steps {0,15,30} gives a hidden array layer of 27*9 nodes. The full set of weight values was derived using a base three count from '00000' to '22222' replacing all the digits '1' with '15' and all the digits '2' with '30'. Using Euclidean distance comparison this obviously gives an unambiguous representation, there only ever being one weight vector as the closest match (this is essential if accurate results are to be obtained).

Hidden to output layer weights

There is one output node for each section of the data form. All the nodes in the output layer are fully connected to the nodes in the hidden layer. The weight connections modify the values calculated by the hidden layer in a manner that reflects the importance of the data. The importance of the data being output by a particular hidden layer node to a particular output layer node can be determined as follows. The input node that receives data from a particular section of the interview form will have a weighted connection to the node in the hidden layer. If the weight value (input to hidden layer) is '30' then the data is very important and the weight value (hidden to output layer) must reflect this. This can be achieved by assigning a weight value of '2' to amplify the data. If the weight value (input to hidden layer) is '15' then the data is moderately important and the connection weight (hidden to output layer) was set to '1' to reflect this. If the weight value (input to hidden layer) is zero then the data is of no importance to the output node and the weight value (hidden to output) is set to zero to reflect this.

4.3.3 Discussions on EMA

In order to ease interpretation of the results produced, additional output nodes were added to indicate the number of vectors that had high or medium scores for each section. The output from the network showed clearly where sectional global deficiencies were (if they existed). Using the sectional magnitudes in conjunction with the breakdown given by the additional nodes, sections that had low magnitudes but contained a high score for one question, could be pinpointed.

A second network was constructed to further process specified sections from the input data and perform the same analysis to give a breakdown to the level of individual questions. This network is obviously not able to generalise since the answer to any question was only a binary value.

Comparing results with a deterministic linear programming approach that compiled overall scores for each question by summing across the entire data set to produce a global picture showed that there was no difference. It was hoped that by inspecting the hidden array layer, insight could be gained into the internal clusters within the data set. This however turned out to be relatively unfruitful, as the picture produced was rather like a badly scaled graph. In not responding to the density patterns of the data in the input space, the model did not provide sufficient detail in the areas required. The insight into the internal clusters of the data set was minimal.

4.4 The vector memory array

In parallel with the design of EMA as an unsupervised technique, the Vector Memory Array method (VMA) [6] was developed as a supervised counterpart. A supervised technique was obviously of no value as far as the solution to the stated problem was concerned, but in the path of the research it was a natural progression between EMA and the interrogative memory structure that is reported in chapter 5. The principle of mathematically deriving the weights was furthered with the exploration of using the actual training data as the weight values.

The algorithm is based on vector quantisation techniques and is not restricted to operation in the binary space. One of its most distinctive features is the fact that it does not have a traditional propagated learning phase [7], and thus the speed with which the network produces results is very high.

VMA is a supervised learning paradigm utilising a four layer network (see figure 4.5). The input layer is fully connected to the first hidden array layer. The second hidden layer is a 1-D 'winner takes all' structure, selecting the smallest 'n'† Euclidean or non-Euclidean distances (calculated by the first hidden layer) to propagate through from the first hidden layer. The output layer is fully connected to the 'winner takes all structure'.

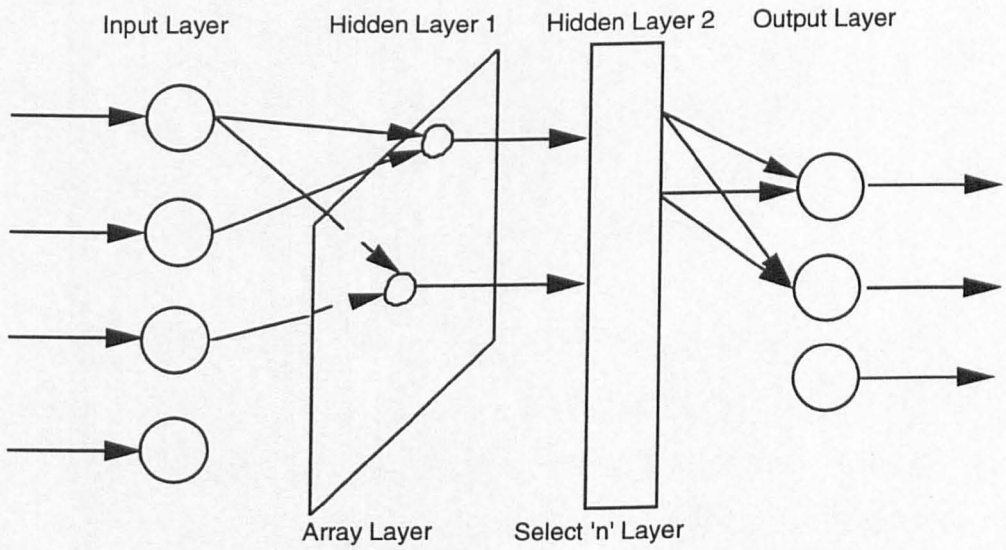


Figure 4.5 The Vector Memory Array

† Where 'n' is a user defined parameter, the Grouping index (see later).

4.4.1 Determination of weight values

The input layer contains the same number of nodes as the training vectors contain elements. The output layer contains the same number of nodes as there are classes, and hidden layer 1 contains the same number of nodes as there are training vectors.

Weight values are not so much learnt as loaded via the following stages. The first training vector is presented to the input nodes. The values are 'loaded' as the weights for the first cell in the hidden layer. The second training vector is then presented and it is used as the weights for the second cell in the hidden array layer and so on, until all the training vectors have been used. Each training vector consists of the data vector plus its target class as the last element. Table 4.2 shows an example of a five element vector.

	Element 1	Element 2	Element 3	Element 4	Element 5	Class
Vector 1	0.54	0.23	0.67	0.44	0.34	2

Table 4.2 A typical weight vector

The weights for the output layer are derived by inspecting the last element of the weight vector for the node they are connected to in the hidden array layer (via the 'winner takes all structure'). If the element contains the same number as the class the output node is to represent then the weight is set to '1' otherwise the weight is set to zero.

4.4.2 Functions computed by the nodes

The input nodes compute no function; they only transfer the value presented to them to their outputs. The cells in the hidden array layer calculate the distance between their weight vector and the pattern vector input, and output the calculated value. Four different distance metrics F have been used in the hidden array layer. The first is the Euclidean distance shown below as equation 4.1; the second is the city block style metric shown below as equation 4.2. equation 4.3 shows a variation of the city block style metric that has been found to produce increased accuracy in some situations and equation 4.4 shows the formula for the angle between two vectors that has also been used as a distance metric.

$$F = \sqrt{\sum_j (e_j - u_{ij})^2} \qquad (4.1)$$

The Euclidean distance metric

$$F = \sum_j \mathbf{abs}(e_j - u_{ij}) \qquad (4.2)$$

The City block style metric

$$F = \prod_j \mathbf{abs}(e_j - u_{ij}) \qquad (4.3)$$

The modified City block style metric

$$F = \mathbf{arccos}\left(\frac{E.U_i}{|E|\,|U_i|}\right) \qquad (4.4)$$

The "Angle between two vectors metric"

Where E is the input vector and e_j its jth element, U_i is the weight vector and u_{ij} its jth element with i dimensions and **abs** is the absolute value function.

4.4.3 Derivation of the output layer transfer function

The purpose of an individual output node is to present the user with a magnitude that represents two qualities for a specific class, and these are defined using the following two clauses:

Clause 1

The magnitude must reflect the closeness of the active inputs (from the 'Select n' hidden layer) to the input vector.

Clause 2

The magnitude must reflect the number of active inputs to the node (and thus reflect the number of vectors of the node's class that have been 'short listed' by the 'Select n' hidden layer).

As can be seen in Equation 4.5, the approach has been to sum the inverse of the distances and divide by the number of active inputs. This results in the average 'closeness' and satisfies the requirements of clause 1. A variable dependant on the number of active inputs is then introduced to satisfy the requirements of clause 2.

$$F = \frac{1}{N}\sum_{i=0}^{N-1}\frac{1}{x_i} + \mathbf{a}\ln(N) \qquad (4.5)$$

Where x_i is the ith active input to the node, a is a user defined variable and N is the number of active inputs the node receives.

4.4.4 The grouping index

At present the "winner takes all" structure propagates the 'n' smallest Euclidean distances from the hidden array layer to the output layer. A variation on this is to specify a minimum Euclidean distance rather than a group size. However when this approach was used it was discovered that the accuracy of classification was reduced very slightly by one or two percentage points. This could be accounted for by the number of decimal places used in setting a distance. This manner of specification was found to be much more difficult since the user was required to know what order of magnitude to use. In practice the specification of group size was used. The short calculation time for simulation enabled the grouping index to be varied through its full range and the best results taken.

4.4.5 Summary of operation

The operation of the network can be summarised as follows. A test vector is presented to the input layer. The nodes in the hidden array layer calculate the distance between their weight vectors (excluding the last class identifier element) and the input vector (which has no class identifier element). Each node in the hidden array layer outputs the distance calculated. The operation of the select 'n' layer is more transparent. The hidden array layer could be viewed as being fully connected to

the output layer. The Select 'n' layer allows the connections carrying the closest 'n' distances to be active and specifies all others as inactive. The weight vectors of the output layer then ensure that only relevant distances reach particular nodes, and the output nodes then use equation (4.5) to produce their outputs (the final output for the network) from the distances presented.

4.4.6 Plasticity/ Stability

The plasticity/ stability dilemma highlighted by Carpenter and Grossberg [8] is as follows; how can a network be adaptive and yet remain stable? Alternatively, how can a network learn new information without forgetting that previously learnt.
It is of obvious importance for any network once trained, to be able to accept further training to enable new or more accurate data to be incorporated into the "knowledge" base. This is not a problem with VMA, as new data can be loaded into the array at any time. New groups can be incorporated simply by adding an output node (with associated weight vector). The dilemma of old data being lost is not a problem either, since the data is still present in the array. If it is required that original data should be neglected, then it can simply be removed from the array structure.

4.4.7 Comparison With Alpaydin's "Grow And Learn" Algorithm

There are similarities between VMA and Alpaydin's 'Grow and Learn' Algorithm [9]. The first and most apparent similarity is the way in which vectors from the training set directly become the weight values between the input and the first hidden layer. The second similarity is in the second hidden layer. Alpaydin uses a 'winner

takes all' structure; VMA uses an expanded version that takes the 'n' closest vectors. The similarities however stop here. VMA does not use the waking and sleeping periods (although, some sort of pruning algorithm may well be a useful addition). The difference in behaviour between the two network paradigms is due to the 'winner takes all' structures and the output layers. In Alpaydin's paradigm the network propagates only the winner from the hidden layers to the output layer. The output layer then sums its inputs and outputs a '1' if the value is above a threshold. VMA's winner takes all structure outputs the 'n' smallest Euclidean distances. These values are then combined using equation 4.5.

4.4.8 Discussions On VMA

The main advantage of the Vector Memory Array method is that it is computationally very efficient since it does not involve a conventional learning phase and generates accurate results on the test data sets. The testing of VMA was primarily carried out on data taken from the Warwick Electronic nose [10]. This data has been analysed by many techniques [11,12,13] and so allowed good bench-marking. On these data sets VMA outperformed back propagation techniques both on speed and percentage accuracy. Results of this and other tests carried out can be found in chapter 8. The most obvious limitation is that the array in the first hidden layer stores all the training vectors. This obviously makes the technique cumbersome for problems with extremely large data sets. Some sort of pruning algorithm may prove a useful addition to remove unnecessary duplication and redundancy in the weight matrix. At the time of writing, memory is a relatively cheap commodity and

the quantities being used by VMA are not massive. VMA may well pay for its extravagant memory usage in a commercial environment by the computation time it saves. In testing, VMA reduced to tens of minutes problems that had taken days to compute by back propagation methods. Large amounts of memory are becoming increasingly standard on personal computers and it is currently recommended that Microsoft Word be run in 8MByte of RAM. This growth trend looks set to continue as prices continue to fall and developers add functionality to their software. Whilst the memory required may be prohibitive for small standalone systems, VMA is a plausible analysis tool for use on PCs.

4.5 References

[1] Kohonen T; "Self-organisation and Associative Memory"; Third Edition: Springer-Verlag, New York, 1989

[2] Dayhoff J; "Neural Network Architectures: An introduction"; Van Nostrand Reinhold: 1990, New York, pp. 163-191.

[3] Simpson P; "Artificial Neural Systems, Foundations, Paradigms, Applications, and Implementations"; Pergamon Press, New York, 1990, pp. 85-90.

[4] Kohonen T, Chrisley R and Barna G; "Statistical pattern recognition with neural networks"; Benchmarking studies: IEEE International conf. on neural networks, Vol 1-2, CH.165-VL.002: 1988, pp.61-68

[5] Larkin A B, Hines E L and Thomas S M; "The Euclidean Memory Array, A vector quantisation technique for the processing of data from interview forms"; Neural Computing and Applications (1994) 2:53-57 Springer-Verlag London Ltd.

[6] Larkin A B, Hines E L, Thomas S M and Gardner J W; "Supervised learning using the vector memory array method"; Proceedings of the Workshop on Neural Network Applications and Tools. Sept 13-14 1993, Liverpool England. Ed. Lisboa P, Taylor M. IEEE Computer Society Press, ISBN 0-8186-5845-2, pp. 6-10.

[7] Rumelhart D E, Hinton G E and Williams R J; "Learning internal representations by error propagation"; In (Rumelhart D E, McClelland J L) Parallel Distributed Processing Vol 1: The MIT Press, Cambridge, MA, 1986, pp.318-362.

[8] Carpenter G A and Grossberg S; "The ART of adaptive pattern recognition by a self-organising neural network"; IEEE Computer Magazine; March 1988; pp. 77-88.

[9] Alpaydin A; "Neural models of incremental supervised and unsupervised learning"; These n. 863, Department d'informatique, Ecole Polytechnique Feferale de Lausanne, Switzerland, 1990

[10] Gardner J W, Shurmer H V and Tan T; "Application of an electronic nose to the discrimination of coffees"; Sensors and Actuators B6 71 - 75, 1992.

[11] Gardner J W, Hines E L and Wilkinson M; "Application of artificial neural networks to an electronic olfactory system"; Meas. Sci. Technol. 1, 1990, pp. 446-451.

[12] Hines E L, Gianna C and Gardner J W; "Neural Network based electronic nose using constructive algorithms"; in "Neural Networks: Techniques and Applications", Lisboa and Taylor (Eds), Ellis Harwood (ISBN 0130621838), 1993.

[13] Hines E L, Gardner J W and Fekadu A A; "Genetic algorithm design of neural net based electronic nose"; International conference on neural networks and genetic algorithms, University of Innsbruck, Austria, 13-16 Apr, 1993.

Chapter 5
Clustering with Interrogative Memory Structures

5.1 Introduction

Developing further the idea of having mathematically derived weights led to the conception of the Interrogative Memory Structure (IMS) [1]. This paradigm follows the Vector Memory Array ideology of storing the training data as the weight values. The interrogative memory structure was the third major development in the pursuit of this research (following on from the Euclidean and vector memory arrays [2,3]) and proved to be relatively successful. The concept for the network came from the Jets and Sharks example in the book Parallel and Distributed Processing [4]. In this example a network holds information linking people with their jobs, education and age. The user interrogates the network by pulling nodes active, e.g. if the node for "high school" was pulled active, the nodes representing the people with high school education will become active and thus the output will be the people with the quality set by the user and their average qualities. It can be seen that a network of this type could be used for clustering. The exact nature of the algorithm used will be shown in the rest of this chapter. The paradigm showed capable of finding clusters within data sets and thus was capable of fulfilling the objectives laid out in the statement of intent for this research. This chapter gives a full treatment of the development of the IMS architecture and gives details of the topology and paradigm used to obtain the results shown in chapter 8.

As with self-organising maps [5], the topology is simplistic and the controlling algorithm makes use of this simplicity. Below, the IMS is developed through several stages. The paradigm can be represented as a network for implementation on parallel

architectures, or as in the case of many networks, can also be represented in an algorithmic form using a stored knowledge base.

5.2 The Architecture

The idea for the network came from the Jets and Sharks example in the book Parallel and Distributed Processing. To implement this concept for clustering, the network architecture they used was unsuitable as it used a propagated phase to arrive at the results and because of this would be unnecessarily slow. To solve the problem an architecture had to be created that would perform the required data mapping.
In any data set, the actual data items perform a mapping between a measuring device and a quantity being measured. In an electronic nose example [6], the data maps the measuring device (the sensors) to the quantity being measured (the coffee aroma or other smell). In the case of the survey of engineering firms, the data maps the questions on the forms (measuring device) to the companies being surveyed (the quantity being measured). With these thoughts in mind, a two layer network was created, the top layer representing the objects being measured, and the bottom layer representing the measuring devices. The weights between the two layers then became the data set being used. In the binary space being used for this thesis, a weight of value one corresponded to a positive correlation between two nodes and a value of zero to a lack of correlation.

As with self-organising maps, the architecture is really the network equivalent of a look up table, in this case put into a parallel architecture by creating a bi-directional network as can be seen in figure (5.1).

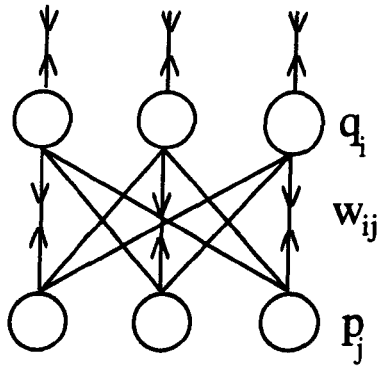


Figure 5.1 The topology used for an IMS (all connections are bi-directional)

5.3 Derivation of the Transfer Functions

All input and output is performed via the Q nodes, as shown in figure 1. The weights are loaded from a pre-processed data set. In the engineering survey application the P nodes can be thought of as representing companies, and the Q nodes can be thought of as representing questions. The weight values **W**, map the companies to the questions and are therefore the answers the companies have given to the questions. The network can be interrogated by pulling Q nodes active and then propagating the pattern through the weighted connections to the P nodes. The P nodes that receive an active input are pulled active. A second propagation stage is then entered into where the active P nodes propagate their output backwards to the Q nodes.

The overall effect is that the user selects a pattern of questions to present to the input of the layer and the network returns the companies containing the pattern of active questions as a sub-set of their answers. The output of the network is then, the average answers to the questions for all the companies whose vector contains the original user-selected question list as a subset of their active question answers (the active question answers being all the positive question answers or binary '1's in the vector).

5.3.1 Mathematical Derivation of transfer function

The user selects an active question vector where the elements are members of a binary set with '1' representing a positive question answer and "0" representing a negative question answer. More generally, the user selects the pattern of 'ones' in a vector that will be the minimum correlation between the returned vectors, i.e. if the user sets bit zero high, all the returned vectors will have at least the commonality of bit zero high.

$v_i \in \{0,1\}$ (an element in the user vector)

&

$w_{ij} \in \{0,1\}$ (an element in the weight matrix of question answers)

In the simplest model where nodes are restricted to binary operation and any P node becomes active for any positive correlation with the user vector the following holds:

$$p_j = \sum_{i=0}^{n} q_i w_{ij} \tag{5.1}$$

$$q_i = \mathrm{E}v_i + \overline{\mathrm{E}}\sum_{j=0}^{n} p_j w_{ij} \tag{5.2}$$

Where E is a synchronisation signal indicating the direction of data transfer. A more complex operation can be performed by only bringing P active if V is a subset of W (using V and Q interchangeably as Q passes V in the forward propagation phase). Mathematically,

if $\forall\ v_i = 1\ \exists\ w_{ij} = 1$ then $p_j = 1$

or

if $V \subset W_j$ then $p_j = 1$

To derive a suitable equation to carry out this function it is necessary to consider truth table 5.1.

q_i	w_{ij}	p_j
0	0	1
0	1	1
1	0	0
1	1	1

Table 5.1 Truth table for desired response of P.

Table 5.1 shows all possible combinations of weight vector bit w with input pattern vector bit q and the corresponding desired response for the p node.

From the above table, the activation function for the P nodes can be derived as shown in equation 5.3.

$$p_j = \overline{q_i} + q_i w_{ij} \tag{5.3}$$

Generalising equation 5.3 for the full interconnection gives equation 5.4,

$$p_j = \prod_{i=0}^{n} (\overline{q_i} + q_i w_{ij}) \tag{5.4}$$

A further addition can be made to force the Q layer to handle real numbers. This gives IMS significantly more scope and forms the basis of the completed system. Equation 5.5 shows the modification to the activation function.

$$q_i = \mathrm{E}v_i + \overline{\mathrm{E}}\mathrm{T} \tag{5.5}$$

Where T is computed as a real value (equation 5.6). The above is not strictly true as T is a real number and not in the binary space, but for the purposes of this example it is considered to be operated on in 2^n space. The essence is that if the network is in the first propagation stage, V is the output, otherwise the real value T is the output. This operation is rather like that of an analogue switch, with E being the digital control input and V and T being the switched variables.

$$\mathrm{T} = \frac{1}{\mathrm{n}} \sum_{j=0}^{n} w_{ij} p_j \tag{5.6}$$

Using the above structure and activation formulae allow a network to be created using a binary data set such as the engineering survey. The network can then be interrogated by a controlling algorithm. The nature of the controller will determine the nature of the insight on the data set.

5.4 The Controlling Paradigm

The aim was to produce a controller that would find clusters in the data. The controller descried below used the *real numbered* version of the network for reasons that will become obvious. The controller repeatedly interrogates the memory structure adding to clusters discovered until they either meet the criteria laid down by the user or the algorithm can find no further clusters. The routine starts by presenting a vector to the controller with only the least significant bit (LSB) set high. The network returns the average vector for all the weight vectors with the LSB set high. If in the returned vector any other bits are greater than the threshold value β, the controller then presents a vector containing the original LSB and also the other bits that were high in the returned vector. This process continues until no further bits in the return vector are greater than the value of β. The average vector returned and the active P nodes are then stored. The number of active P nodes is the number of vectors in the cluster and the weight vectors of those P nodes are the vectors in that cluster.

This process is then continued by presenting a fresh vector with a bit in the next place to the LSB set high and the same process entered into until no further bits in the returned vector are greater than the value of β. The start bit for the interrogation is rotated across each bit position in turn. Duplicate clusters are then removed from

the store. If the number and quality of the clusters found is not adequate the value of β is then decremented and the process started again. For each search with a certain β value the number and quality of clusters is stored so that the best value of β can be located. In the experimentation carried out β started with a value of one and was decremented by 0.1 until it reached zero.

The pseudo code for a controller is listed below, where β is as defined above.

```
repeat
{
β= 1
i = 0
repeat
        {
        repeat
                {
                pull q node i active
                count all Q nodes with activation levels above β
                if number  > 1 then set corresponding Q nodes high
                }until return vector is unchanged

        if number > minimum similarity then store information for group

        increment i

        }until i > number of nodes
```

decrement β

remove duplicate groups

}**until** (number of groups found > minimum number required) **or** (β =0)

If (β = 0) **and** (number of required groups has not been reached) **then** output best found

---

The need for β may not be obvious at first sight but will be explained in section 5.5.

5.5 Example of Use

The great advantage of the marriage of this controller with the interrogative memory structure in the application chosen for this study is that all clusters found have at least some common qualities that will run through the entire of the cluster.
Let us clarify with an example. Consider the six bit binary vectors below that can be separated into the two distinct groups vectors 1-4 and vectors 5-8. The commonalities are that vectors 1-4 have bits 3 and 5 set high where as vectors 5-8 have bits 1 and 2 set high.

v1	101000
v2	101001
v3	111000
v4	101100
v5	000110
v6	100110
v7	000111
v8	010110

starting the controller algorithm running from the least significant bit to the most significant bit the controller pulls bit zero high on the Q nodes and sets β to one.

Node pulled high = bit 0

presented vector

0	0	0	0	0	1

response vector

0.5	0	0.5	0.5	0.5	1

vectors in cluster = v7, v2

loop ends due to lack of bits above β

Node pulled high = bit 1

presented vector

0	0	0	0	1	0

response vector

0.25	0.25	0	1	1	0.25

vectors in cluster = v5, v6, v7, v8

Nodes pulled high = bits 1 & 2

presented vector

0	0	0	1	1	0

response vector

0.25	0.25	0	1	1	0.25

vectors in cluster = v5, v6, v7, v8

loop ends due to lack of bits above β

Node pulled high = bit 2

presented vector

0	0	0	1	0	0

response vector

0.4	0.2	0.2	1	0.8	0.2

vectors in cluster = v4, v5, v6, v7, v8

loop ends due to lack of bits above β

Node pulled high = bit 3

presented vector

0	0	1	0	0	0

response vector

1	0.25	1	0.25	0	0.25

vectors in cluster = v1, v2, v3, v4

Nodes pulled high = bits 3 & 5

presented vector

1	0	1	0	0	0

response vector

1	0.25	1	0.25	0	0.25

vectors in cluster = v1, v2, v3, v4

loop ends due to lack of bits above β

Node pulled high = bit 4

presented vector

0	1	0	0	0	0

response vector

0.5	1	0.5	0.5	0.5	0

vectors in cluster = v3, v8

loop ends due to lack of bits above β

Node pulled high = bit 5

presented vector

1	0	0	0	0	0

response vector

1	0.2	0.8	0.4	0.2	0.2

vectors in cluster = v1, v2, v3, v4, v6

loop ends due to lack of bits above β

If the algorithm has been set to find the clusters with the maximum similarity, then it can be seen above that it will return the two intended clusters. In a full implementation of the paradigm the controller would then decrement β and start again. Decrementing β to 0.9 will have no effect when it is decremented to 0.8 the search would be extended at two stages, as shown below.

Node pulled high = bit 2

presented vector

0	0	0	1	0	0

response vector

0.4	0.2	0.2	1	0.8	0.2

vectors in cluster = v4, v5, v6, v7, v8

Previously the controller terminated here, but as β = 0.8, the search now extends as follows.

Nodes pulled high = bits 1 & 2

presented vector

0	0	0	1	1	0

response vector

0.25	0.25	0	1	1	0.25

vectors in cluster = v5, v6, v7, v8

loop ends due to lack of bits above β

The search will also be extended in the following situation.

Node pulled high = bit 5

presented vector

1	0	0	0	0	0

response vector

1	0.2	0.8	0.4	0.2	0.2

vectors in cluster = v1, v2, v3, v4, v6

Previously the controller terminated here, but as $\beta = 0.8$, the search now extends as follows.

Nodes pulled high = bits 3 & 5

presented vector

1	0	1	0	0	0

response vector

1	0.25	1	0.25	0	0.25

vectors in cluster = v1, v2, v3, v4

loop ends due to lack of bits above β

As can be seen, the controller has not found any contradictory evidence, so the clusters returned will remain the same. As β is decremented further other extensions to the search will be attempted, but it can be seen that they will terminate without success.

When tested on the data sets described in chapter 7, the IMS was found to perform well. Clusters with sizeable similarity were located. The results of the tests run can be found in chapter 8.

5.6 References

[1] Larkin A B; "Clustering with interrogative memory structures"; Submitted to Neural Computing and Applications.

[2] Larkin A B, Hines E L and Thomas S M; "The Euclidean memory array, a vector quantisation technique for the processing of data from interview forms"; Neural Computing and Applications (1994) 2:53-57 Springer-Verlang London Ltd.

[3] Larkin A B, Hines E L, Thomas S M and Gardner J W; "Supervised learning using the vector memory array method"; Proceedings of the Workshop on Neural Network Applications and Tools. Sept 13-14 1993, Liverpool England. Ed. Lisboa P, Taylor M. IEEE Computer Society Press, ISBN 0-8186-5845-2, pp. 6-10.

[4] Rumelhart D E, Hinton G E and Williams R J; "Learning internal representations by error propagation"; In (Rumelhart D E, McClelland J L) Parallel Distributed Processing Vol 1; The MIT Press: 1986; pp. 25-31.

[5] Kohonen T; "Self-organisation and Associative Memory"; Third Edition: Springer-Verlag; 1989

[6] Gardner J W, Hines E L and Wilkinson M; "Application of artificial neural networks to an electronic olfactory system"; Meas. Sci. Technol. 1, 1990; pp. 446-451.

Chapter 6
The Theory of Clustering in a Binary Space

6.1 Introduction

At least two distinctly different approaches can be pursued with the use of clustering algorithms. The path chosen depends on the amount of prior knowledge held about the data. If it is known that the data contain say, three clusters, then the techniques applied can be judged on the degree to which they manage to separate out these classes. On the other hand, if there is no prior knowledge of the internal structure of the data, then the basis for the comparison of techniques becomes more complex. This body of research deals with the latter case and so with the work detailed in this thesis centring around a comparison of techniques that locate clusters within a data set, it is of obvious importance to have a firm definition of a cluster. Not only that, but it must also be known on what basis clusters can be compared, what is considered a good cluster and what is considered a bad cluster. In the following pages the definition used in this thesis is presented along with supporting theory. The metrics that have been used to compare clusters and techniques to derive the results in chapter 8 are shown.

6.2 What is a cluster?

An obvious assumption to make would be that a cluster consists of a group of points around a single point in a given space, the extreme case being that a cluster would be a single point and its nearest neighbours. Consider an 'n' dimensional binary vector V,

$V_i \in \{1,0\}$ where $0 \leq i \leq n$

In this work vectors are only considered in a binary space. The space is a subset of the Euclidean real space, but still defined by the same Hamel basis (i.e. the basis vectors are orthogonal and linearly independent). In three dimensions the basis would be as follows (1,0,0), (0,1,0) and (0,0,1).

It is intuitive that there will be n immediate neighbours of the minimum unit of distance away from V. To show this, consider the scenario where n=3. Taking an arbitrary vector,

V = (0,1,0)

Immediate neighbours existing will be,

(1,1,0)

(0,0,0)

(0,1,1)

Thus for n=3, there are three immediate neighbours.

There is only interest in correlations of positive responses from the survey forms, i.e. values of $v_i = 1$.

Using the above information a cluster can be defined as follows:

$\exists \mathbf{G}$ where $\mathbf{G} \subset \mathbf{W}_j$

$\exists v_i = 1$ and $\forall j, g_{ij} = 1$

Where **G** is the cluster, **W** is the data set and V is a vector in the data set. So, *there exists a subset of the data set such that there exists a particular bit position within*

that subset for which all vectors have a value of unity. That is to say that all the vectors in a cluster have at least one common quality, where "common quality" is defined as a bit position (dimension) equal to unity. The three dimensional vectors in table 6.1 have the common quality present in the Z plane.

X	Y	Z	
0	0	1	V1
1	0	1	V2
1	1	1	V3

Table 6.1 Three vectors with commonality in the Z dimension

Therefore a cluster is not necessarily only the collection of nearest neighbours to a point in space. In a data set it is quite possible for duplication of certain vectors to be present and this would constitute a single point cluster. The complete picture is then, built up of duplication and similarity. Therefore vectors in a cluster can either be identical or "similar".

6.2.1 Similarity in a binary space

From the above definition of a cluster, similarity between two vectors can be taken to be a single, or greater number of bit positions, that have an identical numeric value of "one" for all vectors. Therefore similar vectors are all in the same plane. Extending this argument, the largest cluster that can exist in a complete "n" dimensional binary space will contain all the vectors that define vertices of a particular plane. Numerically that would be 2^{n-1}. This statement omits the possibility of a vector being duplicated in a data set.

6.2.2 Duplication in a binary space

It is obvious that any vector can be duplicated any number of times (within the constraints of the size of the data set). The amount of duplication of individual vectors defines the probability distribution. Assuming an even distribution:

Total number of vectors = 2^n

The probability of a specific vector being included once in a data set = $1/2^n$

If **L** is defined as the length of a data set, the chance of a data set containing only one vector duplicated **L** times = $(1/2^n)^L$

In reality, it is extremely unlikely that a data set will have an even distribution. In the market research arena being discussed in this thesis, the actual distribution is affected by parameters such as the wording of questions and layout of the questionnaire. It can however be said that,

To locate a complete cluster, take a plane, locate all vertices and duplicates.

But will clusters found in this way be optimal? The problem can be extrapolated further, for as will be seen in the next section, sub-clusters can exist within a plane. The degree of sub-division necessary to locate optimal clusters is dependent on the nature of the distribution.

6.2.3 Existence of sub-clusters within a plane

For the sake of readability, this section is further sub-divided. The issue of sub-clusters within a plane is first tackled without the added complication of duplicate vectors being present in the data set, and then expanded to incorporate them.

6.2.3.1 Without Duplication

Just as the data set as a whole has its own distribution, so sub-distributions are present for sub-sets of the whole. Extracting all the vectors from a data set that are contained within a particular plane will give a cluster that fulfils the definition laid out above. The vectors of this cluster will be set in their own distribution and therefore, it is entirely possible that sub-clusters may exist. All of this is obviously dependent on the exact nature of the distribution. To demonstrate this principle , consider the complete set of vectors that form a cluster in the Z plane (with n=3) as shown in figure 6.1.

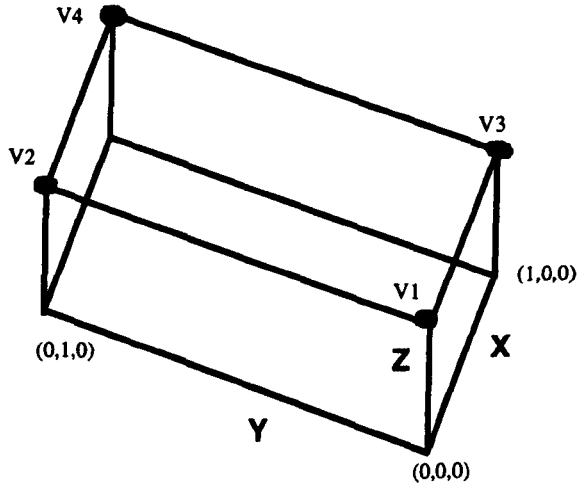


Figure 6.1 The complete set vectors that form a cluster in the Z plane

The marked vertices have the vectors shown in table 6.2.

x	y	z	
o	0	1	V1
o	1	1	V2
1	0	1	V3
1	1	1	V4

Table 6.2 The complete set vectors that form a cluster in the Z plane

Potential sub-clusters exist {V2,V4}, {V3,V4}.

The number of sub-clusters will increase with dimensionality. In fact with n=4, not only are sub-clusters present but sub-sub-clusters, as shown in table 6.3 for clusters in the l plane:

i	j	k	l	
0	0.	0	1	V1
0	0	1	1	V2
0	1	0	1	V3
0	1	1	1	V4
1	0	0	1	V5
1	0	1	1	V6
1	1	0	1	V7
1	1	1	1	V8

Table 6.3 A cluster in the L plane with sub-clusters present

Potential sub-clusters include

{V2,V4,V6,V8}

{V3,V4,V7,V8}

{V5,V6,V7,V8}

Potential sub-sub-clusters are:

{V4,V8}

{V3,V7}

etc..

The overall effect can be viewed as a tree diagram (figure 6.2).

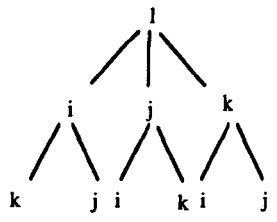


Figure 6.2 A cluster hierarchy tree diagram

The parent cluster has the commonality in terms of the **L** plane. A sub-cluster can be found by combining this commonality with one in the **K** plane. Further, a sub-sub-cluster could be located by introducing the **J** plane. Depending on the parameter to be optimised (i.e. size/ quality) the tree can be cut at any point to find the relevant clusters.

6.2.3.2 With Duplication

In a data set where duplication is present, the fundamental difference is that multiple instances of individual vertices exist. This does not change the fact that all members of a cluster will be in the same plane. However, branches of the tree may carry more weight due to duplication. Viable clusters could exist as multiple instances of a single vector and this would be an ideal cluster, although to some extent the definition of an ideal cluster is application dependant.

In the application under investigation, the ideal cluster would contain an optimisation of the maximum number of vectors with the maximum commonality. But just how can that be measured? The next section details how clusters can be scored against one another and follows on with generalising the principle to how techniques can be compared.

6.3 Cluster Metrics

Having given definition to a cluster, the next issue to be addressed is how the quality of a cluster can be measured. This definition will always be application dependent as the attributes that define a cluster are application dependent. Taking the definition previously stated,

$\exists \mathbf{G}$ where $\mathbf{G} \subset \mathbf{W}_j$

$\exists v_i = 1$ and $\forall j, g_{ij} = 1$

The crucial quality in this scenario is the commonality of at least one positive logic state. The ideal cluster would be some optimisation of maximum size against maximum commonality. With both these qualities being equally important, identical weighting can be placed on them. The task of producing a metric is then not a complicated one, and more easily explained with an example, taking the vectors below to be a cluster.

(0,0,1,0,1)

(0,1,1,0,1)

(0,0,1,1,1)

(1,0,1,0,1)

If the size of the data set is set arbitrarily at 20 vectors, the percentage of vectors taken from the whole by the cluster can be taken to be a metric of size, in this example calculated as follows:

4/20 = 0.2

A measurement of commonality can then be produced by taking the number of complete columns of unity one as a percentage of the possible, as shown in table 6.4.

0	0	1	0	1	**V1**
0	1	1	0	1	**V2**
0	0	1	1	1	**V3**
1	0	1	0	1	**V4**
0	**0**	**1**	**0**	**1**	**2/5 = 0.4**

Table 6.4 A measure of commonality taking complete columns

The score for the commonality is then 0.4. In the above example, there existed the possibility of five complete columns. Only two however shared the required commonality. A more general case of this could be taken by calculating the percentage commonality of each column instead, as shown in table 6.5.

0	0	1	0	1	**V1**
0	1	1	0	1	**V2**
0	0	1	1	1	**V3**
1	0	1	0	1	**V4**
1/4 = 0.25	**1/4 = 0.25**	**4/4 = 1**	**1/4 = 0.25**	**4/4 = 1**	**2.75/5 = 0.55**

Table 6.5 A measure of commonality using the percentage of '1's

The two metrics can then be combined to produce an overall score for the cluster. The method of combination can produce further insight into the quality of the cluster. The two methods used in this thesis are arithmetic and geometric mean. In the market research scenario, the geometric mean is of greater interest, as a cluster with no commonality, but large membership is of no significance.

6.4 Comparing Clustering Algorithm

Having defined a metric for clusters, it is a relatively simple step to define a measure for the output produced by a specific clustering technique. In essence, all that is required is a metric which combines the scores attributed to individual clusters produced by the algorithm. Two distinct elements are present then; the scores

themselves and the number of them (different techniques will find different numbers of clusters). There is a large class of algorithms that attempt to place every vector in the data set into a cluster. This can result in the production of some useful clusters surrounded in a bed of small useless clusters. Some techniques will even go as far as forming clusters containing only one vector. When dealing with large data sets, wading through this sea of unnecessary information can become rather tiresome and for this reason it is useful to position a threshold. Clusters with a number of vectors less than the threshold can then be ignored. In practice setting the threshold at ten percent of the size of the data set was found to be satisfactory (as no obvious problems were encountered). Not all techniques attempt to assign vectors to clusters in this manner and therefore seemingly produce less output. Using the threshold allows for the comparison of differing techniques on a level playing field, as only the meaningful clusters are being compared. Producing an overall metric for a technique can be as simple as summing the scores for the individual clusters. The exact nature of the combination of the scores to produce the final metric needs to be application dependant, since it depends on whether it would be considered useful to have a small number of large clusters or vice versa.

Chapter 7
The Data Sets

7.1 Introduction

The focus of this thesis is two fold; firstly the development and comparison of unsupervised neural network clustering algorithms with statistical routines; and the application of these neural computing techniques to the problem of finding internal clusters in binary market research data. The comparison of the neural and statistical algorithms was carried out using the application data sets, that is, data taken from market research surveys. The parametric distributions are not Gaussian, but are formed by responses to the questionnaires. It is useful when considering the results presented in this thesis to have some knowledge of the structure of the data sets.
In what follows the details of the unsupervised data sets are given followed by the details of the supervised data sets used for bench marking the Vector Memory Array [1].

7.2 Data Sets for Clustering

7.2.1 The Large market research data sets

The data used for the comparison of the clustering algorithms were binary market research data. The SAS manual [2] considers a data set to be "large" if the number of vectors exceeds one hundred. By that definition all four data sets used with the unsupervised techniques are large. Two of them contain almost one thousand vectors and the other pair around five hundred and fifty. The data originates from two extensive telephone surveys carried out on the engineering industry. Unfortunately

the Data Protection Act does not allow further details about the survey to be published. What now follows is an explanation of how the data was derived from the survey data sets and a graphical mathematical perspective of the data used.

7.2.2 From Interview Forms to Data Sets

The interview form used for the market research was not specifically designed to fit into the boundaries of this thesis. The data produced was not, on the whole, binary. It contained text fields, real integers and multiple choice answers. To produce a suitable data set which could be used for the purposes of the comparison required a substantial amount of pre-processing. The data provided by Bench-Mark Research[1] were from two surveys carried out. The first survey had been carried out on nine hundred and thirty three companies and contained forty main questions. The second survey was carried out on five hundred and forty seven companies and contained thirty main questions.

To obtain usable data from the two surveys, the questions with binary answers were located and extracted. This produced two data sets of seven and ten bits width, respectively. These data sets were then analysed by sight and columns that had an obviously high density removed, thus creating two further, narrower data sets, the latter two data sets having no apparent clusters. Thus four data sets were created. The dimensions are given in table 7.1.

[1] Bench-Mark Research, Swanley, Kent.

Data Set	A	B	C	D
Length (No. vectors)	547	547	933	933
Width (No. bits)	5	7	10	7

Table 7.1 The Dimensions of the Four Large Data Sets

The following figures show plan view images of the data sets. A white horizontal bar indicates the digit '1' and a black horizontal bar a '0'. Each row is one questionnaire, each column is one question. As can been seen, data set A was produced from set B by removing columns two and three. Likewise data set D was created from set C by removing columns one, two and seven.

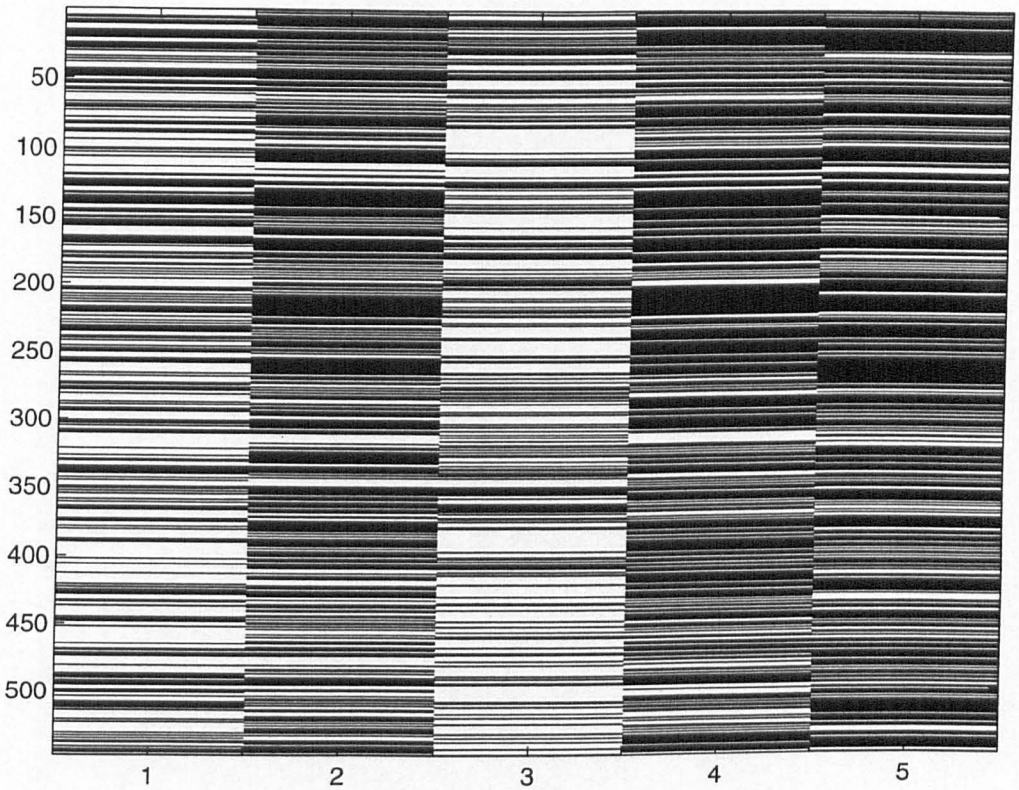


Figure 7.1 Data Set A

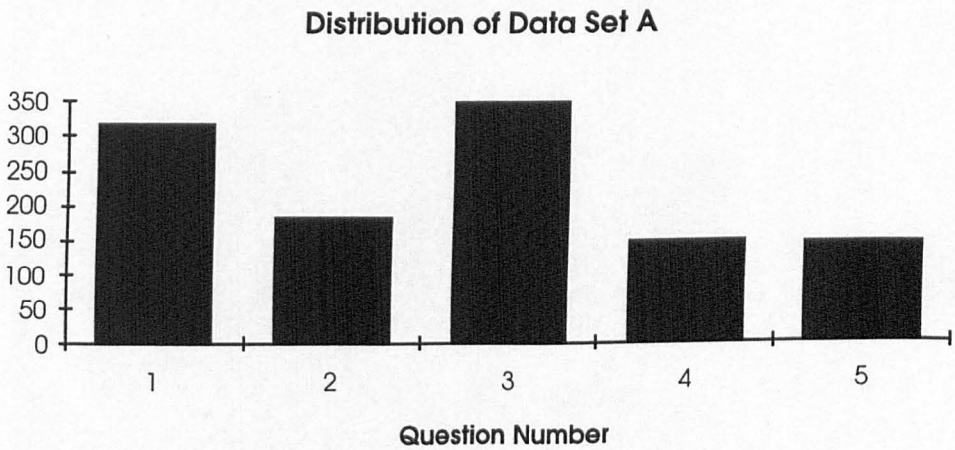


Figure 7.2 The Distribution of Data Set A

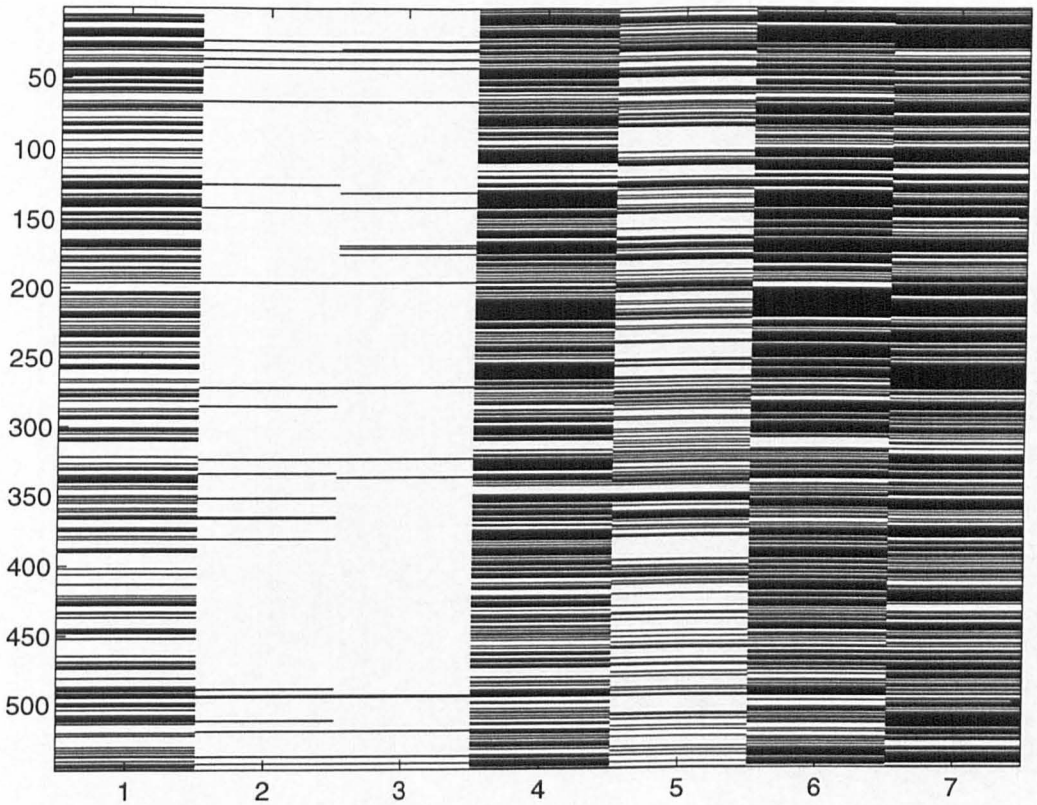


Figure 7.3 Data Set B

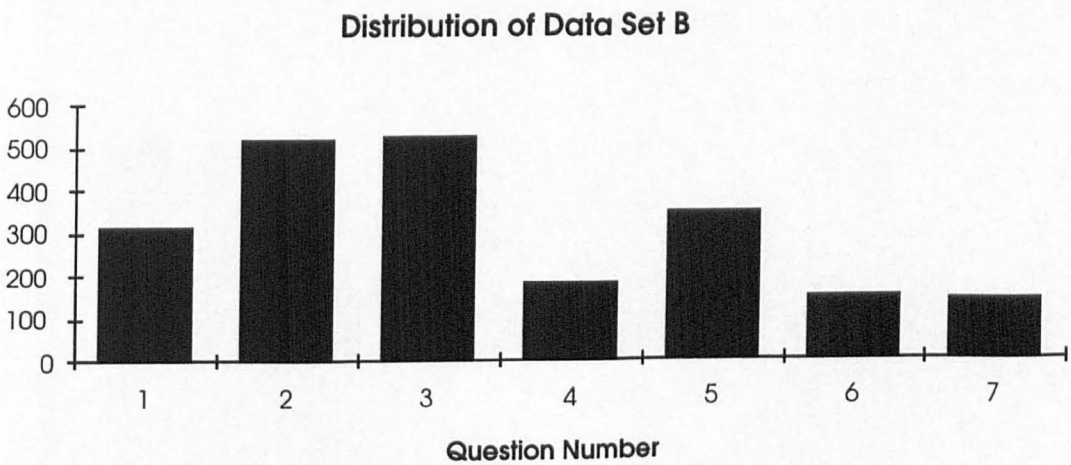


Figure 7.4 The Distribution of Data Set B

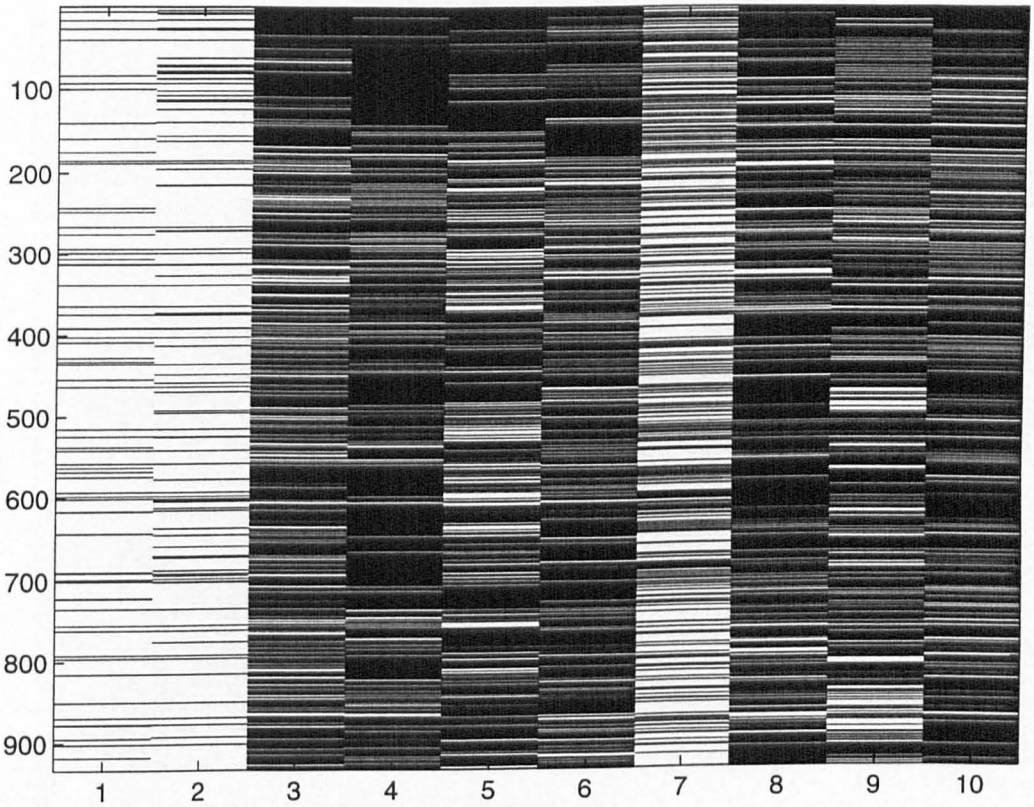


Figure 7.5 Data Set C

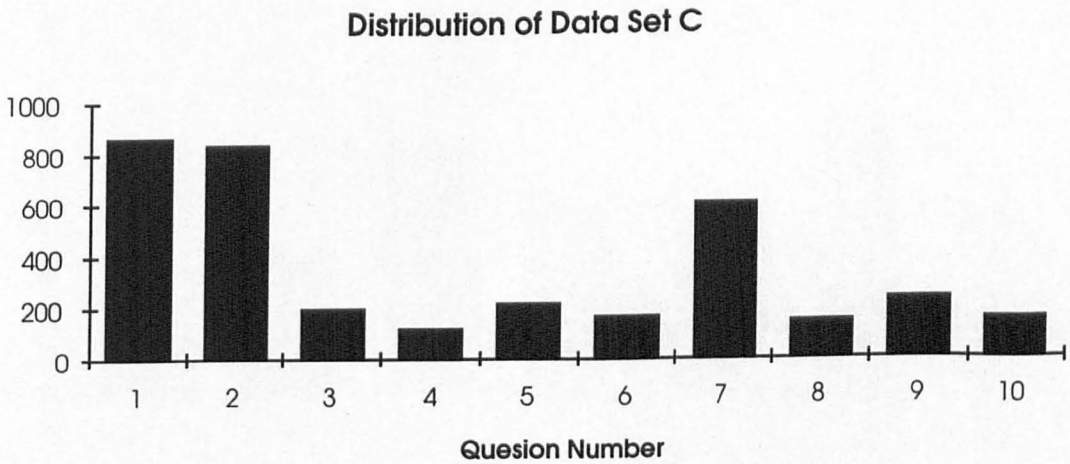


Figure 7.6 The Distribution of Data Set C

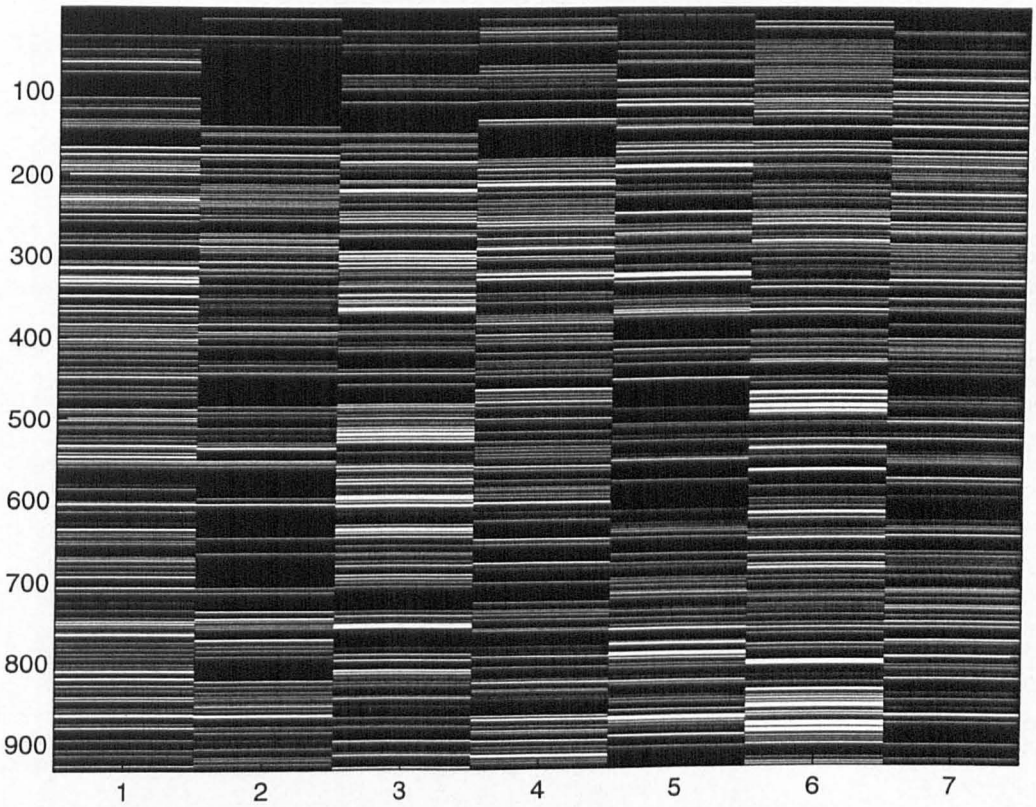


Figure 7.7 Data Set D

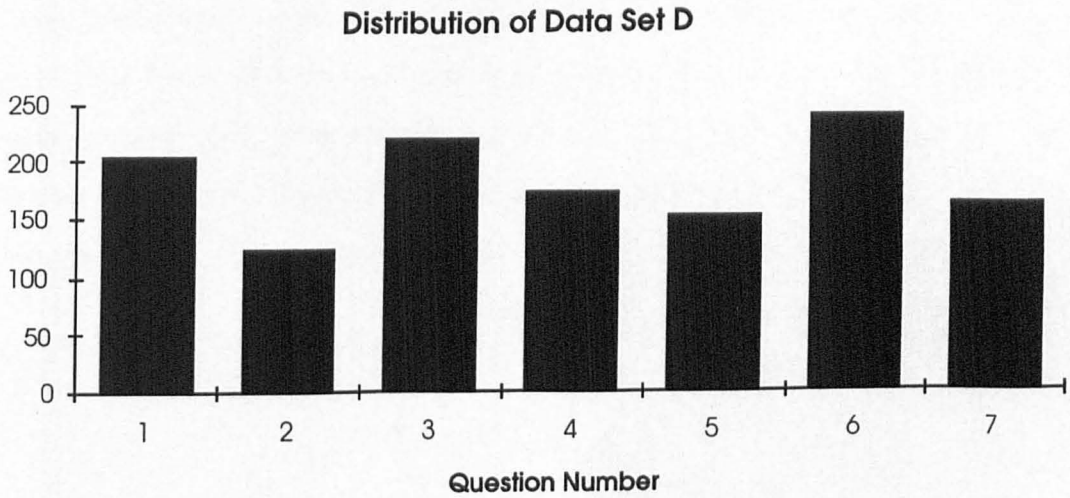


Figure 7.8 The Distribution of Data Set D

The removal of the columns from data sets B and C serves to flatten the distribution as the graphs accompanying the plan view plots show. A visual inspection of the plan view diagram for data set C would immediately lead the viewer to expect that on applying a suitable clustering algorithm to the data, the output would be groups of vectors with commonality in terms of questions 1, 2 and 7. The purpose therefore, in removing these questions from the data set, was to increase the difficulty of the problem. With the flatter distribution seen in the graph accompanying data set D, it is less obvious what commonality a clustering algorithm would find. The same rationale was used for data set B to produce set A. The removal of questions 2 and 3 can be seen to have flattened the distribution. It could be argued that questions 1 and 5 should also have been removed. The reason for not removing question 5 was due to the width of the original data set. Removing questions 2 and 3 reduced the data set to 5 bits wide; removing questions 1 and 5 would have reduced the width to 3 bits. This clearly would be too narrow.

Single point clusters are the easiest to identify. Groupings of this nature can be highlighted for visual inspection by placing the vectors in numerical order. Another useful indicator can be gained by removing all duplicate vectors and counting the number of vectors remaining. Table 7.2 below gives details of this metric for each data set.

	A	B	C	D
Initial no. of vectors	547	547	933	933
Width	5	7	10	7
Final no. of vectors	32	46	181	82

Table 7.2 Duplication within the Data Sets

From the table, it can be seen that in removing questions 1, 2 and 5 from data set C, the duplication has in fact been increased. This does not create a problem, since the aim was to have four data sets of varying degrees of difficulty. The number of unique vectors in a data set gives the maximum number of clusters present. The following diagrams give a plan view of the data sets after a numerical sort has been carried out. Some of the single point clusters can be seen quite clearly as adjacent vectors.

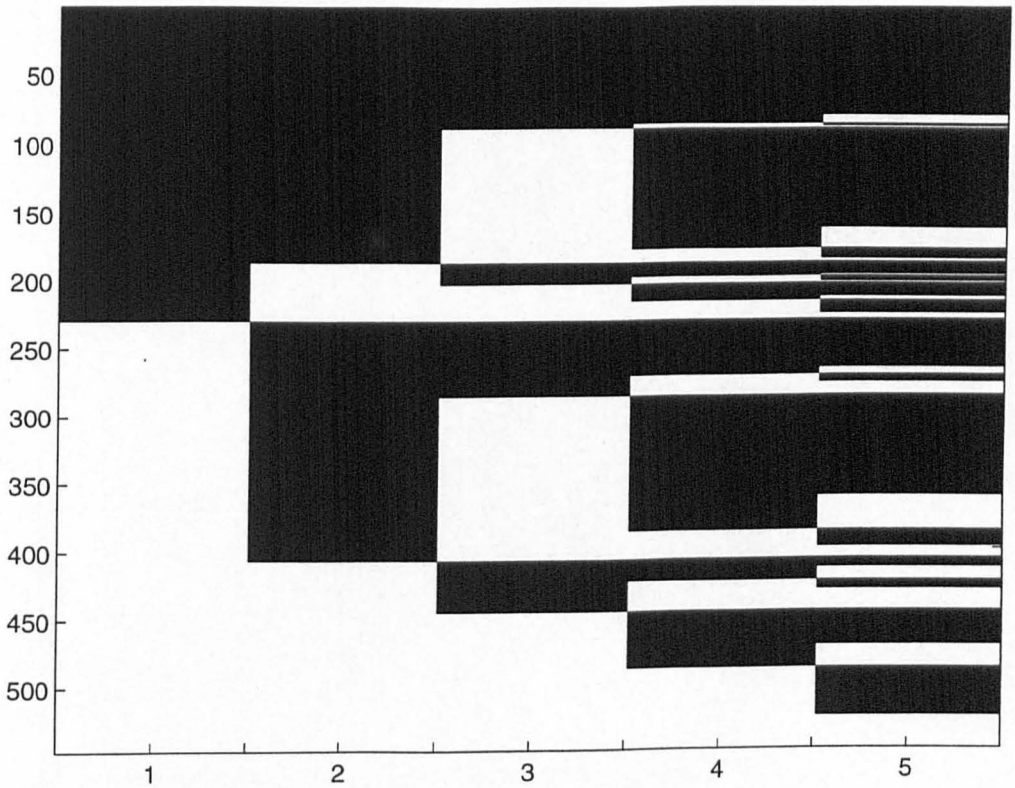


Figure 7.9 Data Set A in numerical order

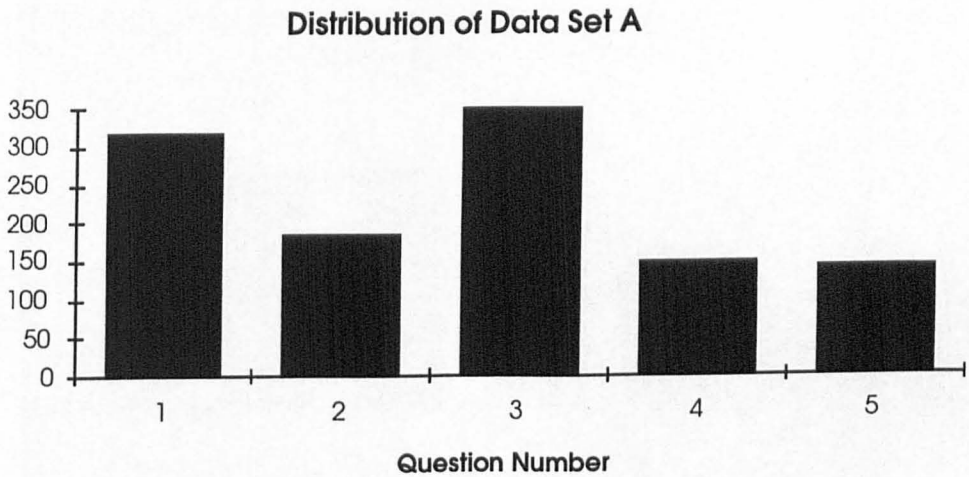


Figure 7.10 The Distribution of Data Set A

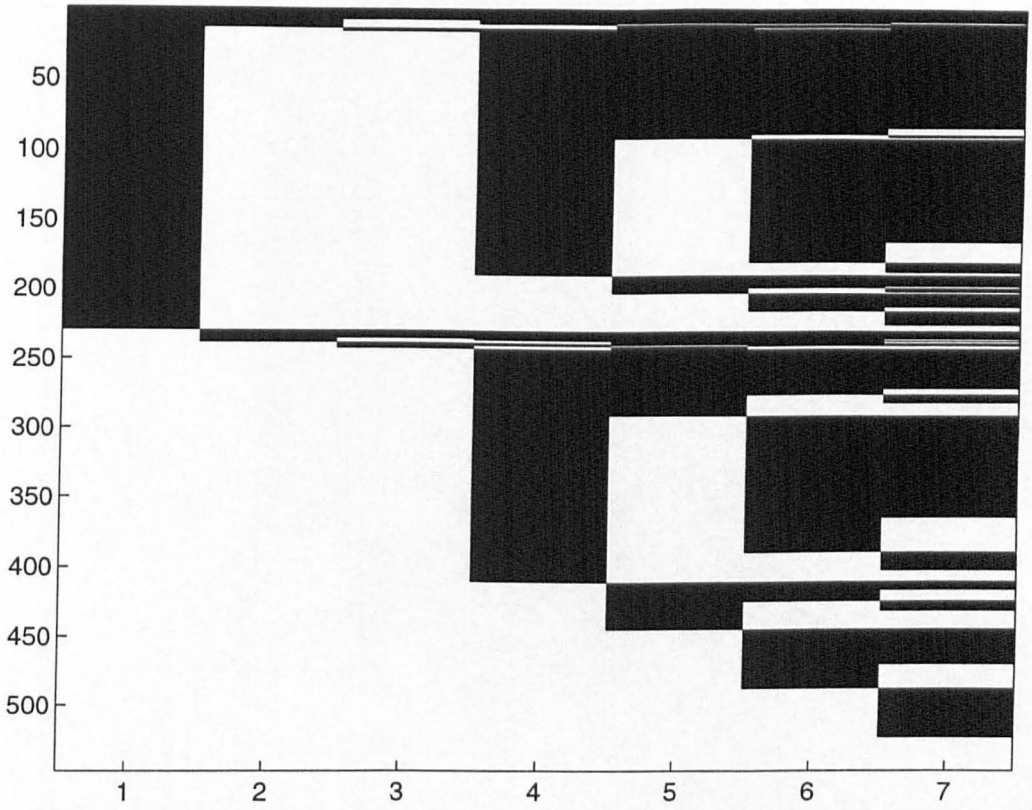


Figure 7.11 Data Set B in numerical order

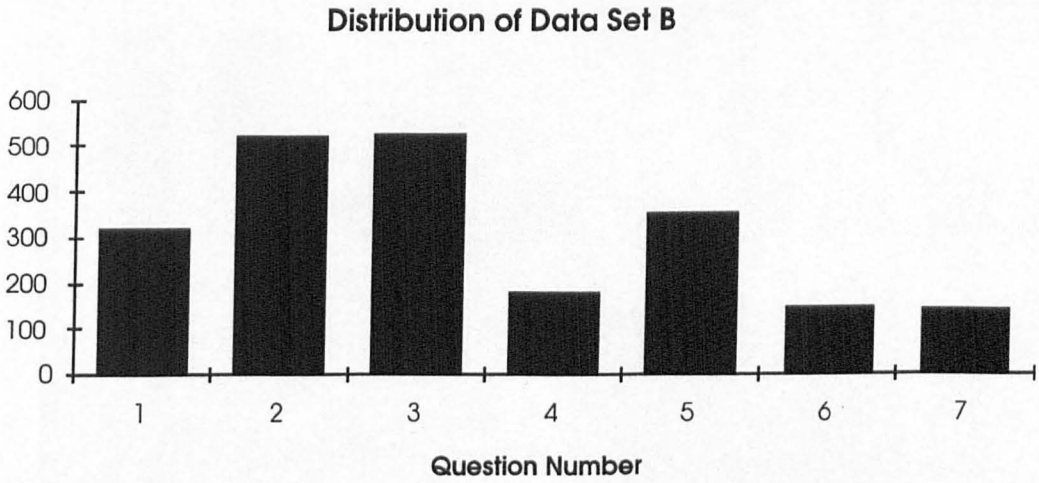


Figure 7.12 The Distribution of Data Set B

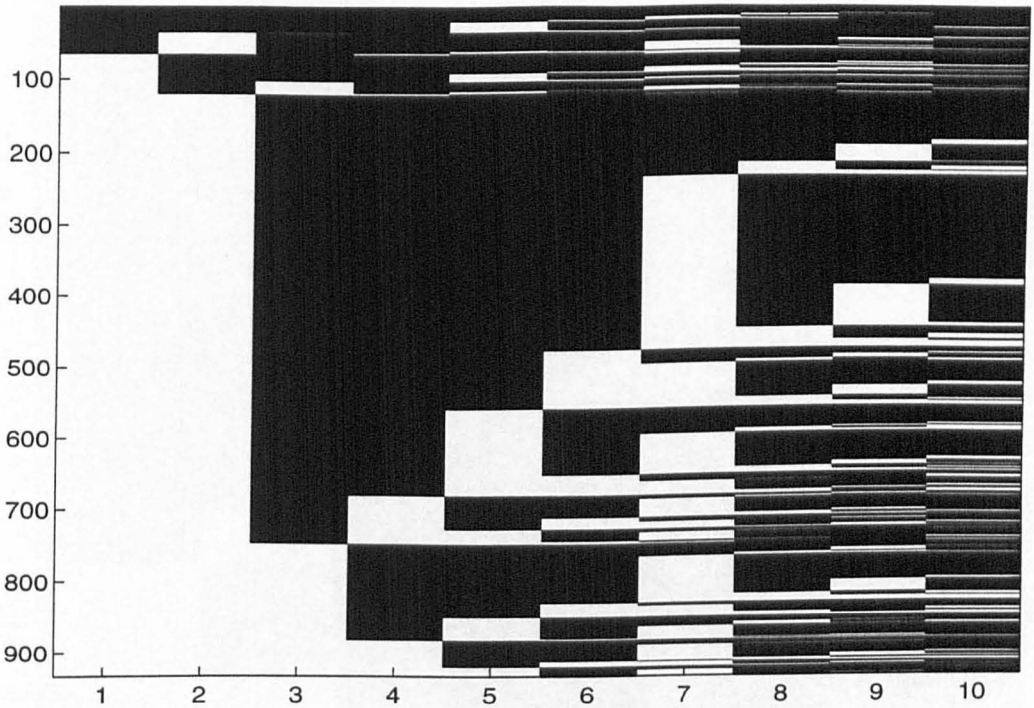


Figure 7.13 Data Set C in numerical order

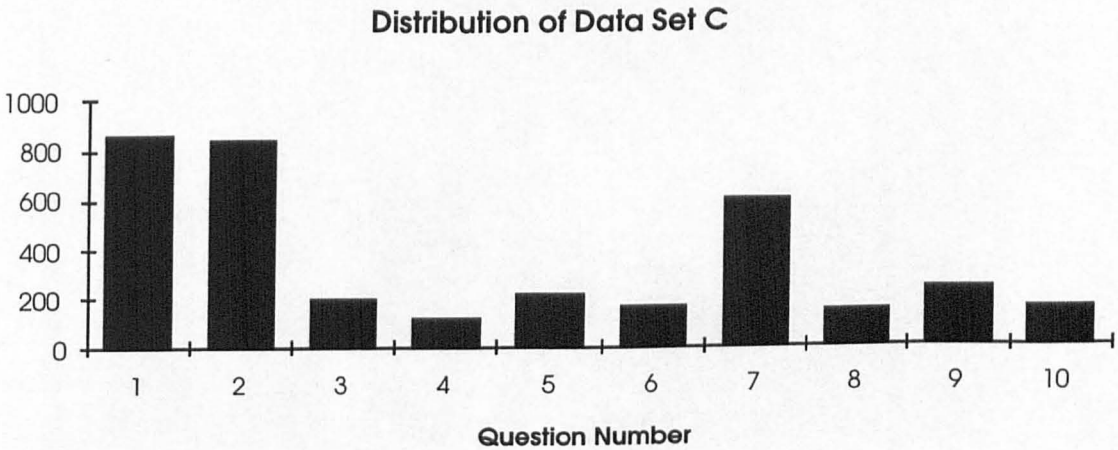


Figure 7.14 The Distribution of Data Set C

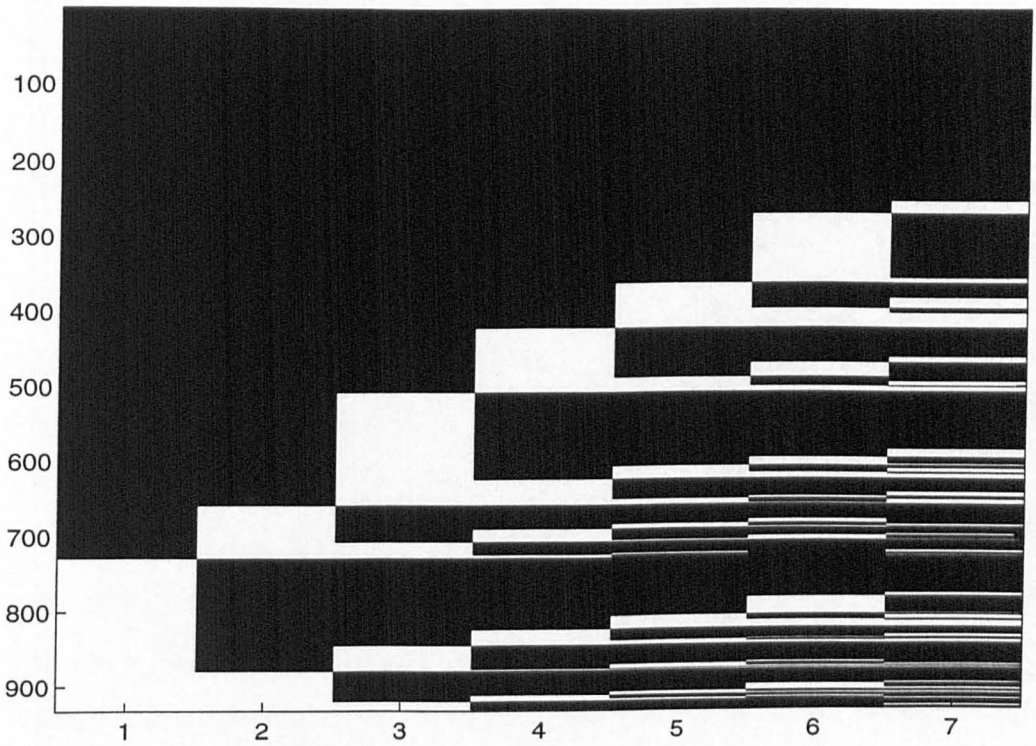


Figure 7.15 Data set D in numerical order

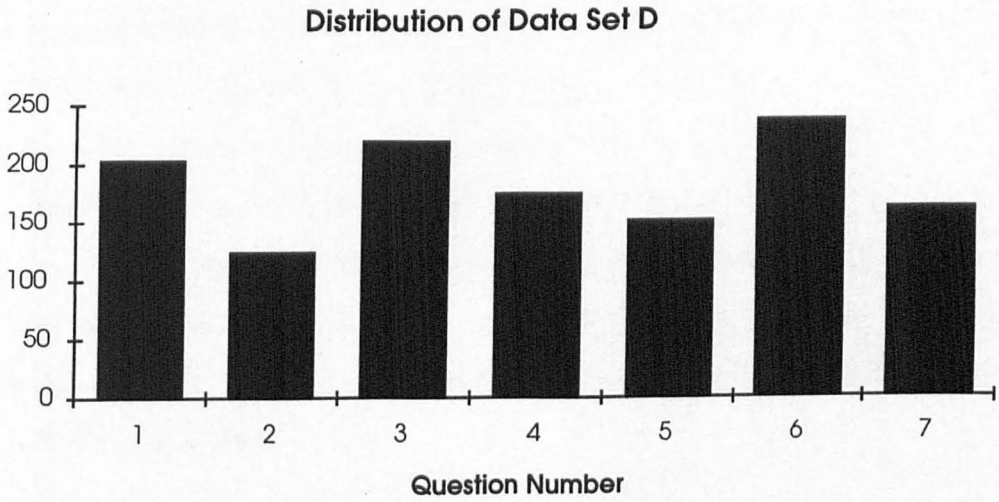


Figure 7.16 The Distribution of Data Set D

7.2.3 The Forgings 2000 data set

Forgings 2000 was a market research survey carried out by Bench Mark Research with Parallax Management Consulting Ltd[2]. The survey consisted of two questionnaires; the first was used to survey companies in the forgings industry itself and the second was used to survey their customers. The aim of the survey was to highlight any mismatch that might be present between the needs of the customers and the service provided by the industry. The survey was reasonably comprehensive for the industry in which it was carried out, but small by the scale of our previous data set. Each survey was carried out on 99 companies.

The aim of including clustering carried out on this data in the thesis is to demonstrate a practical application of the techniques that have been discussed and to compare them. Chapter one talked theoretically about the usefulness of clustering and Chapter six gave a discourse on the theory of cluster formation. The results from the processing of this data show practically what the previous chapters have discussed theoretically. To this end, the data set was chosen for its small size, this being advantageous when attempting to present the results in a written form suitable for inclusion in this thesis. A sub-section of the questions from one of the surveys was taken and processed using the Interrogative Memory Structure [3] technique developed in Chapter five. The results are not simply presented as numbers in terms of cluster metrics, but are then translated back to the survey and the implications of the results highlighted.

---

[2] Parallax Management Consulting, Coventry.

To keep the data set manageable and relevant, the information from just two full questions was encoded into binary for processing. The questions are as follows:

A) Please indicate your level of agreement with the following statements,

0 UNKNOWN

1 AGREE

2 DISAGREE

3 ALREADY UNDERTAKEN

1) We develop the design and specification for our forged products in house

2) We involve our forge supplier(s) in the design of our forged products at the earliest opportunity

3) We involve our supplier(s) of our forged products after we have developed the specification

4) We would like to involve our suppliers of forged products more

5) We buy off the shelf products which meet our specification/ requirements

6) Our purchasing trend is moving towards out sourcing

7) We look for suppliers who can provide us with all our manufacturing needs

8) We require a line side delivery service for forged parts

9) We adopt a policy of single sourcing for forged parts

10) Packaging/ labelling of forged parts is important to us and must meet our requirements

11) We prefer to buy forged products made in the UK

12) We prefer to buy forged products made in Europe

B) Please indicate your level of agreement with the following statements,

0 UNKNOWN

1 ALWAYS

2 SOMETIMES

3 NEVER

1) We expect delivery of forged products strictly to schedule

2) We expect the complete order to be delivered at the same time

3) It is acceptable to receive part orders

4) We would prefer an accurate indication of lead time

5) We would like suppliers to advise us of any problems with supply

6) We expect our suppliers to hold stocks of forgings

7) We require deliveries to be made on a particular day

There are nineteen questions, each of which can be answered with one of four responses, and the coding reflected this. A vector was formulated by taking each of the nineteen questions in turn and coding the response into a four bit binary number. The first four bits of the vector therefore represented the answer to question one, the second four, the answer to question two and so on.

The answers were coded into the four bit binary number; that is to say if the answer was a zero, bit zero was set high; if the answer was a one, bit one was set high and so on. Between each set of four bits, a "guard bit" was added which was always set to zero. This was to enable an easy visual inspection of the data set and results. The decoding process can be seen visually in the table 7.3.

	Question 1	Question 2	Question 3
Question Data	0	1	3
Decoded Segments	1 0 0 0	0 1 0 0	0 0 0 1
Final Vector		1 0 0 0 **0** 0 1 0 0 **0** 0 0 0 1	

Table 7.3 An example of the question decoding (guard bits in bold)

Table 7.3 shows that the four bit segments were decoded in reverse order. For a question answer of zero, the most significant bit was set to one. The implementation was carried out in this way to simplify the programming task. It does not affect results as the implementation was self-consistent.

The actual dimensions of the data set (including guard bits) are 99 vectors by 95 bits. Figure 7.17 shows a plan view of the data set. Figure 7.18 shows a graph of the distribution.

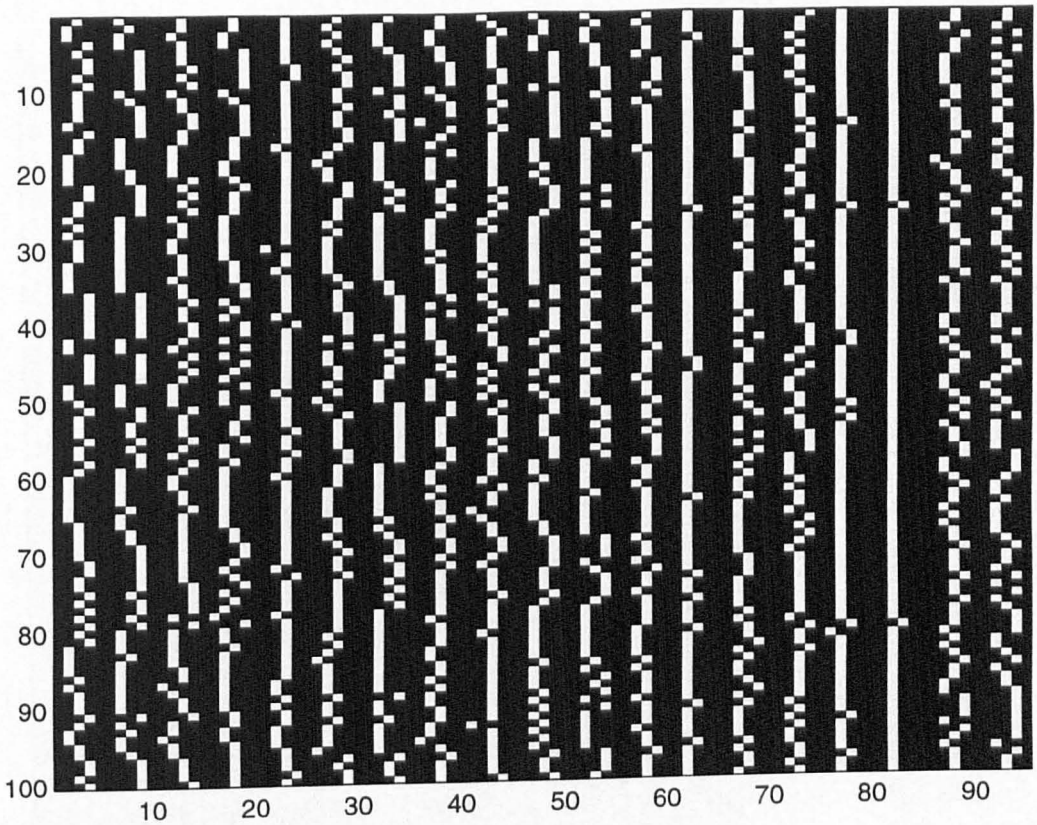


Figure 7.17. Plan view of the F2000 data set

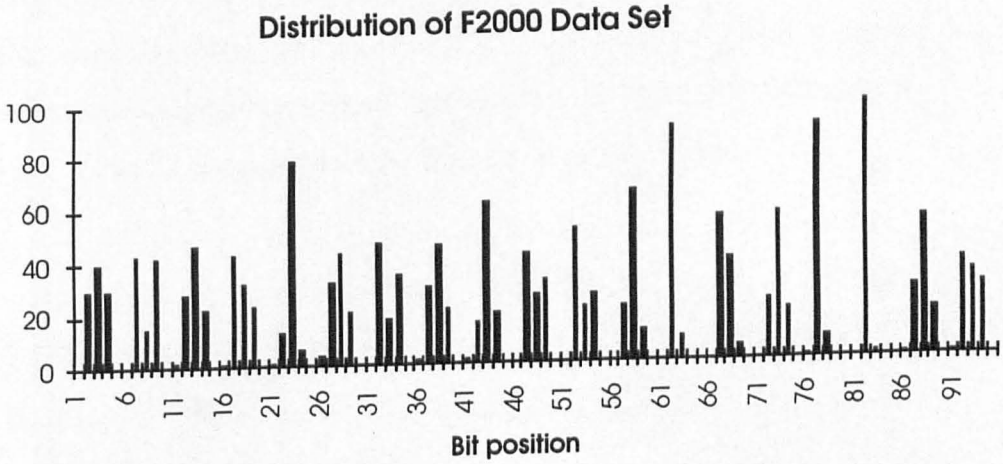


Figure 7.18 The Distribution of the F2000 Data Set

The guard bits can be clearly seen in figure 7.18. However it can also be seen that the distribution is reasonably flat, with a few obvious peeks that may result in clusters.

7.3 Data sets for Supervised Learning

Data suitable for bench-marking supervised classifying techniques were used in the development of the Vector Memory Array network (see Chapter 4). The two main data sets were obtained from the ongoing work into Electronic Noses at Warwick [4,5,6,7]. Both these data sets had been the subject of much analysis and were therefore considered suitable for the bench-marking of a new paradigm. The first data set was from the sampling of different alcohols using tin-oxide gas sensors and the second was from the sampling of different coffees. Mesh plots of both data sets can be viewed in figures 7.19 and 7.21. Normalised versions of the data sets were also used in the bench-marking process. The data were normalised by column, that is to say, the data produced by each individual sensor were normalised with respect to itself. Table 7.4 gives details of the dimensions of the data sets.

	Coffee	Alcohol
Length (Number of vectors)	89	40
Width	12	12
Number of Classes	3	5

Table 7.4 The Dimensions of the coffee and alcohol data sets

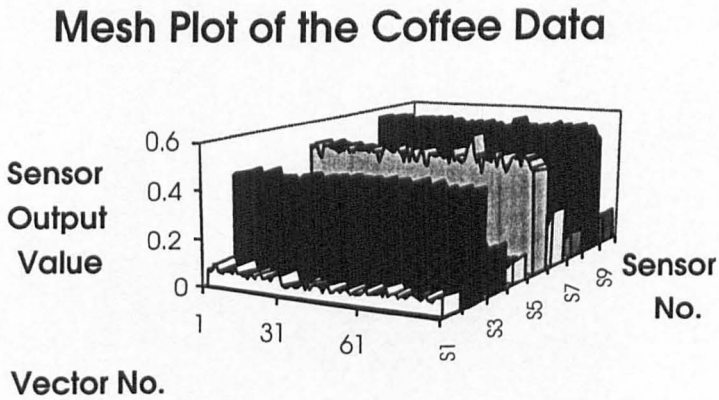


Figure 7.19 The coffee data set

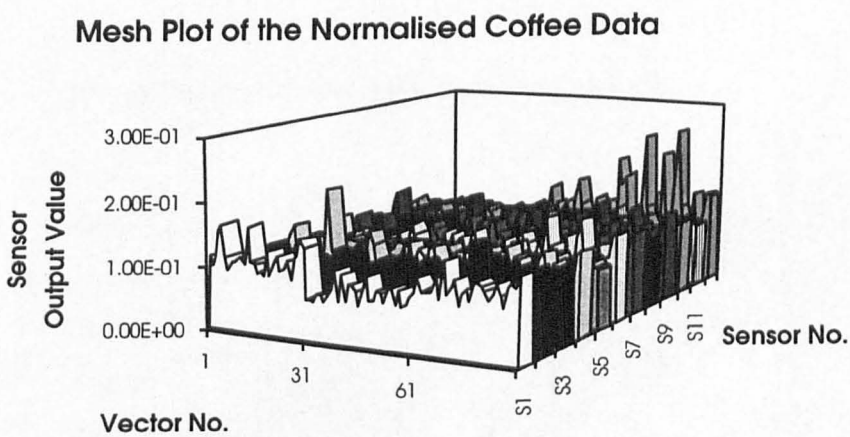


Figure 7.20 The normalised coffee data set

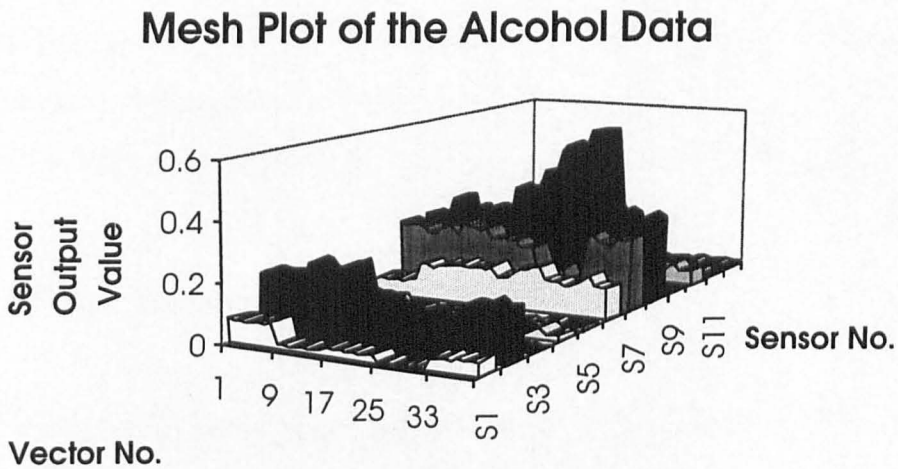


Figure 7.21 The alcohol data set

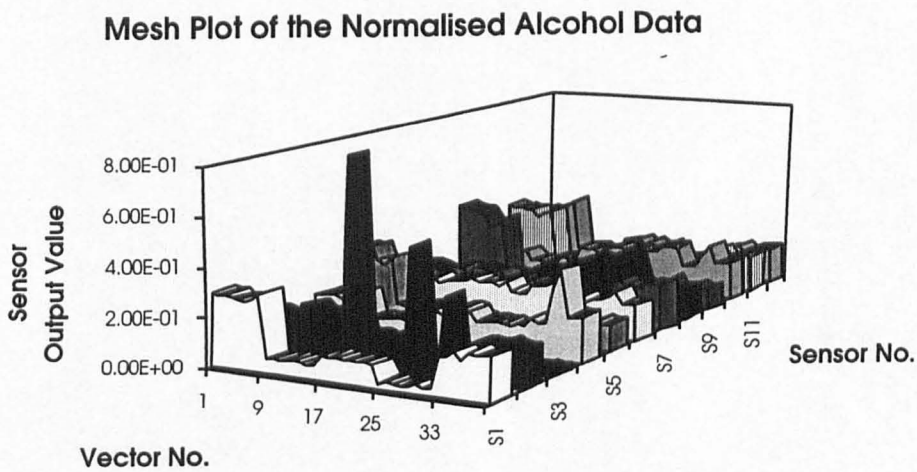


Figure 7.22 The normalised alcohol data set

It should be noticed that the five classes present in the alcohol data set can be visually identified. The three classes present in the coffee data can be seen to some extent, although the distinction is not as pronounced. Previous attempts to classify this data using both statistical and neural network techniques had found the alcohol data to be easily classified, where as the coffee data proved more difficult.

7.4 Other Data

The Vector Memory Array paradigm has been used in other applications [8] producing a similar quality of result to those presented in chapter 8. The author was not involved in this work and so the data sets and results are not presented. The reference to a published work is cited for the purpose of completeness.

7.5 References

[1] Larkin A B, Hines E L, Thomas S M and Gardner J W; "Supervised learning using the vector memory array method"; Proc. Neural Network Applications and Tools; 1993; Ed. PJG Lisboa & MJ Taylor; IEEE Comp. Soc. Press, pp. 6-10.

[2] SAS/ STAT Users Guide, Vol. 1, Ver. 6, Forth Ed; SAS Institute INC, SAS Campus Drive, Cary NC 27513; pp. 823-849.

[3] Larkin A B; "Clustering of binary data using the interrogative memory structure"; submitted to "Neural Computing Applications and Techniques", Springer Verlag.

[4] Gardner J W, Shurmer H V and Tan T; "Application of an electronic nose to the discrimination of coffees"; Sensors and Actuators B6 71 - 75, 1992.

[5] Gardner J W, Hines E L and Wilkinson M; "Application of artificial neural networks to an electronic olfactory system"; Meas. Sci. Technol. 1, 1990, pp. 446-451

[6] Hines E L, Gianna C and Gardner J W; "Neural Network based electronic nose using constructive algorithms"; in "Neural Networks: Techniques and Applications", Lisboa and Taylor (Eds), Ellis Harwood, 1993.

[7] Hines E L, Gardner J W and Fekadu A A; "Genetic algorithm design of neural net based electronic nose"; Int. conf. on neural networks and genetic algorithms, Uni. of Innsbruck, Austria, 1993.

[8] Craven M A, Hines E L, Gardner J W, et al. "Application of an artificial neural network based electronic nose to the classification of bacteria"; 3rd Euro Cong. on Intelligent Techniques and soft computing, Aachen Germany, 1994.

Chapter 8
Results

8.1 Introduction

This chapter presents the results of the analyses. The output of the clustering algorithms is presented along with attributed scores and ranking tables. The experimental techniques used are explained, showing the parameter values used for each procedure. The output produced by the Interrogative Memory Structure algorithm on the forgings data is also presented. This output is then translated back into reality (using the original questionnaire) to show a practical application of the techniques.

Results are also presented from the bench-marking of the Vector Memory Array paradigm on the coffee and alcohol data sets.

8.2 Unsupervised Analyses

8.2.1 The Analysis of the large market research data sets

This set of analyses were run to compare the performance of several different types of clustering algorithms with the paradigm developed in this thesis - the Interrogative Memory Structure. The data sets used to facilitate this comparison can be seen in Chapter 7; they are four data sets derived from a large survey carried out on the engineering industry. The data sets have varying distributions and differing internal structures. The aim of each analysis was to produce clusters that were a compromise between a maximisation of both size and commonality. The methodology used for

scoring the individual clusters and the algorithms as a whole can be viewed in Chapter 6.

8.2.2 Experimental method

The different clustering algorithms differ greatly in the approach taken to finding clusters. Without exception, however, they all require the user to set key parameters before they will efficiently find clusters. As the number of internal clusters in a real data set is usually unknown, the optimal setting of these parameters is a difficult task. In all instances several analyses have been run for a particular technique with a specific data set. The parameter values have been varied through their reasonable paths. In some cases literature has guided the settings and this will be indicated as appropriate.

8.2.3 Parameter values

Many analyses were run with many parameter settings. It was found, however, that the results produced by the parameters settings (listed below) gave a good representation of the output states attained. For the purposes of conciseness other settings that produced similar results are not listed. The settings used are laid out below by technique.

8.2.3.1 Self-organising Map

The software used for these analyses was Kohonen's own software for the Self-Organising Feature Map (SOM) downloaded via INTERNET and run on a UNIX system. The algorithm starts by randomly initialising the weight values. This starting point obviously has some bearing on the ability of the network to locate 'good' clusters. To this end, the simulator has a built-in feature which will run as many different analyses as the user requires (taking a random starting point) and present the best results, where "best" is defined in terms of a low quantisation error. In each case the number of random trials was set to ten. The other parameters to be specified were the topology (either hexagonal or rectangular), the neighbourhood type (bubble or Gaussian), the map dimensions, the training length of the first phase, the training rate of the first phase, the initial radius of the first phase, the training length of the second phase, the training rate of the second phase and the initial radius of the second phase. The manual accompanying the software gives guidance as to how the parameters should be set.

After familiarisation with the software, seven tests were run. The parameter values used are presented in table 8.1. The advice given on parameter settings in the SOM manual states that it is easier to visualise the output of a hexagonal topology due to the fact that all the immediate neighbours of a node have an ordered relationship with it. Thus hexagonal topology was used throughout these experiments. The manual also states that a rectangular dimensioned map (taking into account the probability distribution of the data) has advantages in stabilising the output and that the bubble neighbourhood function is the more reliably implemented one. Parameter values were given in examples for the use of the bubble function and so these were

used as a starting point. Test 1 used the suggested parameters. As table 8.1 shows, the others tests were variations on this.

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7
No. of random trials	10	10	10	10	10	10	10
Topology	Hexa	Hexa	Hexa	Hexa	Hexa	Hexa	Hexa
Neighbourhood	Bubble	Bubble	Bubble	Bubble	Bubble	Bubble	Bubble
X dimension	5	7	4	5	5	5	5
Y dimension	4	6	3	4	4	4	4
First phase training length	1000	1000	1000	1000	1000	1000	1000
First phase learning rate	0.05	0.05	0.05	0.05	0.05	0.5	0.005
First phase initial radius	5	7	4	5	5	5	5
Second phase training length	100000	100000	100000	500000	300000	100000	100000
Second phase learning rate	0.02	0.02	0.02	0.02	0.02	0.2	0.002
Second phase training length	3	3	2	3	3	3	3

Table 8.1. The SOM parameter values used for data set A.

8.2.3.2 Adaptive Resonance Theory

The ART1 analyses were run using the software packages NeuralWorks Professional II Plus. The parameters that required setting were: the vigilance, the number of input nodes, the number of output nodes and the number of training iterations.
The number of input nodes has to be set to the width of the data set. The number of output nodes is effectively the maximum number of clusters ART is allowed to generate. This was varied as table 8.2 shows. The suggested value of vigilance parameter is 0.8. This was taken as a starting point and also varied as shown in table 8.2. The number of iterations was found experimentally and then varied as shown in table 8.2.

	Test 1	Test 2	Test 3	Test 4	Test 5
No. of nodes I/P layer	*	*	*	*	*
No. of nodes F2 layer	10	10	10	10	10
Iterations	10,000	10,000	10,000	10,000	10,000
Vigilance	0.1	0.4	0.6	0.8	0.9

	Test 6	Test 7	Test 8	Test 9	Test 10
No. of nodes I/P layer	*	*	*	*	*
No. of nodes F2 layer	20	40	60	100	100
Iterations	10,000	10,000	10,000	10,000	100,000
Vigilance	0.8	0.8	0.8	0.8	0.8

Table 8.2. Parameter settings for the ART1 analysis (* No. = width of data set.)

8.2.3.3 FASTCLUS

The FASTCLUS algorithm requires only one of two parameters to be specified. The user can either set the minimum distance between clusters or the maximum number of clusters. As setting the distance requires a knowledge of the internal structure of the data set, the only parameter used was the maximum number of clusters. This was varied as shown in table 8.3.

	Test 1	Test 2	Test 3	Test 4	Test 5
Max. No. of Clusters	10	15	20	25	30

	Test 6	Test 7	Test 8
Max. No. of Clusters	40	50	100

Table 8.3. The parameter setting for the FASTCLUS tests

8.2.3.4 Interrogative Memory Structure

No parameter settings are explicitly required for an IMS analysis. The dimensions of the data set must be supplied and in the simulator written for the purpose, the minimum number of clusters to be satisfied with can be specified along with minimum commonalities. These parameters were set so as to allow the simulator to carry out a full search for all clusters present. For each data set, four analyses were run, each using a different method of scoring the clusters. These techniques are

described in Chapter 6. The four techniques sub-divide into two that only take account of complete commonality in a cluster and two that score whatever commonality exists. These two sub-sections then both have either an arithmetic or geometric method of reaching their final score. Full details of these techniques are presented in Chapter 6.

8.3 Results Tables

Following the running of each analysis, the clusters produced were measured using the metrics described in Chapter 6 and results tables formed. From these metrics, overall scores for each analysis were produced and entered into summary tables. From these summary tables the best scores for each clustering algorithm (relative to a specific data set) were taken and overall results tables generated, one for each data set. These overall results tables were then used to compile an overall ranking table showing the relative performance of each algorithm.

To present all the results tables in this thesis would make the work unreadable. For this reason examples have been selected to show the process. The overall results tables are then presented followed by the overall ranking table.

Table 8.4 shows the initial scores obtained by the cluster output from the Self-Organising Map analysis run on data set A.

SOM				% Coverage		% Full Bars		
Ques	People	Full Bars	% Full Bars	Sum Totals	Prod Totals	Sum Totals	Prod Totals	Label
0.26	0.20	1.00	0.20	0.23	0.23	0.20	0.20	1\107
0.72	0.11	0.00	0.00	0.41	0.28	0.05	0.00	1\58
0.40	0.14	2.00	0.40	0.27	0.23	0.27	0.23	1\74
0.04	0.18	0.00	0.00	0.11	0.09	0.09	0.00	1\97
0.20	0.14	1.00	0.20	0.17	0.16	0.17	0.16	2\74
0.40	0.14	2.00	0.40	0.27	0.23	0.27	0.23	2\74b
0.01	0.15	0.00	0.00	0.08	0.03	0.07	0.00	2\80
0.27	0.20	1.00	0.20	0.23	0.23	0.20	0.20	3\109
0.72	0.11	0.00	0.00	0.41	0.28	0.05	0.00	3\58
0.40	0.14	2.00	0.40	0.27	0.23	0.27	0.23	3\74
0.80	0.17	1.00	0.20	0.48	0.37	0.18	0.18	3\91
0.04	0.18	0.00	0.00	0.11	0.09	0.09	0.00	3\97
0.72	0.11	0.00	0.00	0.41	0.28	0.05	0.00	4\58
0.40	0.14	2.00	0.40	0.27	0.23	0.27	0.23	4\74
0.20	0.14	1.00	0.20	0.17	0.16	0.17	0.16	4\74b
0.04	0.18	0.00	0.00	0.11	0.09	0.09	0.00	4\97
0.26	0.20	1.00	0.20	0.23	0.23	0.20	0.20	5\107
0.70	0.13	0.00	0.00	0.41	0.30	0.06	0.00	5\71
0.40	0.14	2.00	0.40	0.27	0.23	0.27	0.23	5\74
0.04	0.18	0.00	0.00	0.11	0.09	0.09	0.00	5\97

0.26.	0.20	1.00	0.20	0.23	0.23	0.20	0.20	6\107
0.84	0.11	2.00	0.40	0.47	0.30	0.25	0.21	6\58
0.43	0.16	2.00	0.40	0.30	0.26	0.28	0.25	6\87
0.04	0.18	0.00	0.00	0.11	0.08	0.09	0.00	6\96
0.40	0.14	2.00	0.40	0.27	0.23	0.27	0.23	7\74
0.84	0.14	1.00	0.20	0.49	0.34	0.17	0.17	7\75
0.03	0.17	0.00	0.00	0.10	0.07	0.08	0.00	7\90
0.24	0.17	1.00	0.20	0.20	0.20	0.18	0.18	7\92

Table 8.4. Initial scores obtained from SOM analysis on data set A.

The meaning of the columns is as follows:

Ques: The number of bits set at one in the cluster, as a percentage of the possible.

People The number of vectors in the cluster as a percentage of the possible.

Full Bars The number of columns containing only bits set to one.

%Full Bars The number of columns containing only bits set to one, as a percentage of the possible.

%Coverage

Sum Totals The total score calculated by the arithmetic method for the percentage coverage measurement (i.e. density of bits set to one).

Prod Totals The total score calculated by the geometric method for the percentage coverage measurement (i.e. density of bits set to one).

%Full Bars

Sum Totals The total score calculated by the arithmetic method for the percentage full bars method (i.e. percentage of columns with all bits set to one).

Prod Totals The total score calculated by the geometric method for the percentage full bars method (i.e. percentage of columns with all bits set to one).

Label Analysis number\ cluster reference number.

Combining the results in table 8.4 produces table 8.5.

SOM	Combined Group Totals			
% Coverage		% Full Bars		Label
Sum	Product	Sum	Product	
1.01	***0.84***	***0.55***	***0.52***	***1***
0.24	0.23	0.14	0.13	2
1.08	**0.96**	**0.54**	**0.44**	**3**
0.79	0.68	0.41	0.40	4
0.76	0.63	0.34	0.28	5
0.66	0.58	0.34	0.28	6
0.70	0.63	0.41	0.35	7

Table 8.5. Combined group totals for SOM analysis on data set A.

The titles of the columns in table 8.5 are as for table 8.4. The label refers to the analysis number. The row in bold indicates the analysis with the highest overall score; the column in bold-italic is the second place analysis.

Tables 8.6 show the overall results for each of the four different scoring methods (see Chapter 6). Tables 8.7 are the overall ranking tables compiled from all the analyses carried out on the four data sets.

(a)

% Full Bars Product				
ART	SOM	IMS	FASTCLUS	Data Set
0.87	0.62	0.78	0.85	A
1.84	1.03	0.71	0.47	B
0.50	0.44	0.70	0.58	C
0.59	0.40	0.17	0.00	D

(b)

% Full Bars Sum				
ART	SOM	IMS	FASTCLUS	Data Set
0.93	0.79	1.08	1.05	A
2.09	1.32	0.74	0.84	B
0.60	0.54	0.73	0.58	C
0.60	0.59	0.19	0.35	D

(c)

% Coverage Product				
ART	SOM	IMS	FASTCLUS	Data Sets
1.10	1.19	0.83	1.09	A
1.92	1.64	0.83	1.04	B
0.98	0.96	0.84	0.88	C
0.80	0.68	0.22	0.41	D

(d)

% Coverage Sum				
ART	SOM	IMS	FASTCLUS	Data Sets
1.32	1.51	1.19	1.43	A
2.21	2.26	0.85	1.33	B
1.01	1.08	0.92	1.00	C
0.83	0.84	0.30	0.53	D

Tables 8.6 The overall results tables

(a)

Overall Ranking %Full Bars			
By Product		By Sum	
ART	1	ART	1
IMS	2	IMS	2
FAST	3	FAST	3
SOM	3	SOM	4

(b)

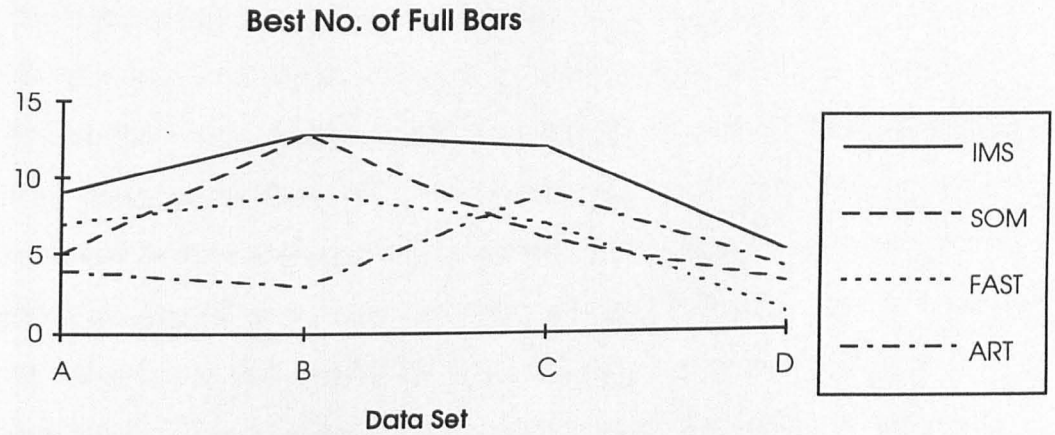
Overall Ranking %Coverage			
By Product		By Sum	
ART	1	SOM	1
SOM	2	ART	2
FAST	3	FAST	3
IMS	4	IMS	4

(c)

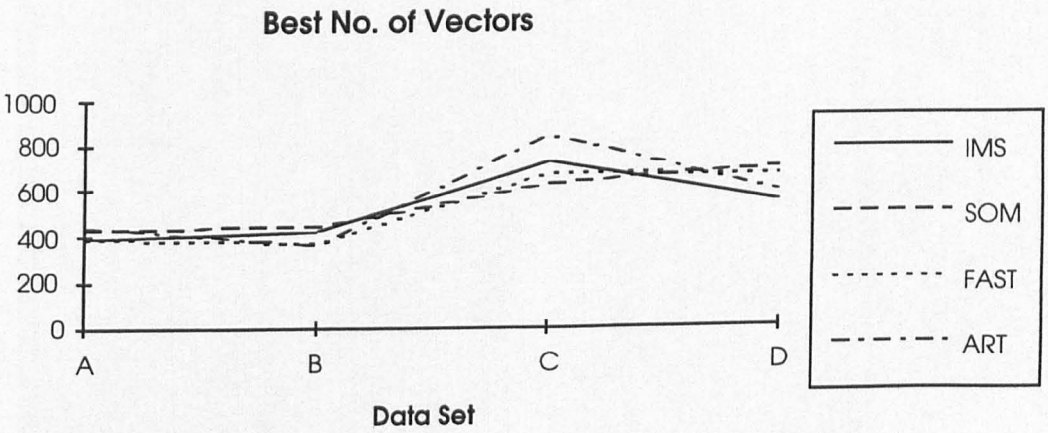
Overall Ranking for each Data Set							
A		B		C		D	
ART	1=	ART	1	ART	1	ART	1
SOM	1=	SOM	2	IMS	2	SOM	2
FAST	2	FAST	3	SOM	3	FAST	3
IMS	3	IMS	4	FAST	3	IMS	4

Tables 8.7 Overall ranking for the four methods of analysis on the four data sets

(a)



(b)



Figures 8.1 Graphs comparing the best clusters found by each technique

8.4 Comments on results

Within the confines of the experiments carried out, it can be seen that ART1 has consistently performed well. Only when the clusters were scored using the more general "%coverage by sum" did ART1 slip into second place. SOM also performed well, gaining its best results in the "%coverage" scores. In reality there was not a significant difference between the clusters found by the differing techniques. The results in tables 8.7 show very mixed results for IMS. The graphs in figures 8.1 show the reason for this. IMS consistently produced clusters containing the highest amounts of commonality, but did this to the detriment of the number of vectors in the clusters. This is also reflected in the fact that IMS scores well in the "%Full Bars" tables, but not in the "%Coverage" tables.

8.5 The Forgings 2000 Analysis

The Forgings 2000 data set was acquired from a survey of the forgings industry. Details of the internal structure of the data set can be found in Chapter 7. The purpose of running an analysis on this data was to show the problem area of this work in a practical light. A small data set was deliberately chosen to enable the output generated to be presented in written form. As the purpose of running the analysis was to find maximum commonality, the IMS algorithm was used with the "full bar product" optimisation (see Chapter 6). In the following section, the results are presented. The nature of the clusters output are shown. A "real world" interpretation to these results is then given from the original question sheet.

8.5.1 Results

Three clusters where generated from the analysis. Table 8.8 gives their qualities.

	No. of Vectors	No. of Full Bars
Cluster 1	13	7
Cluster 2	16	7
Cluster 3	11	5

Table 8.8 The Clusters found by IMS in the Forgings 2000 data set

Table 8.9 shows how the commonality in terms of columns in a cluster decodes into common question answers.

Question No.	Statement	Cluster No.	No. of companies
1/2	Already	1	13
1/6	Already		
1/7	Already		
1/10	Already		
2/1	Always		
2/4	Always		
2/5	Always		
1/2	Agree	2	16
1/7	Agree		
1/8	Agree		
1/11	Agree		
2/1	Always		
2/4	Always		
2/5	Always		
1/5	Disagree	3	11
1/10	Disagree		
1/12	Disagree		
2/4	Always		
2/5	Always		

Table 8.9 The decoding of cluster output into question answers

To take the largest cluster as an example, the information in Table 8.9 can now be applied to the actual questions as shown in Chapter 7. So the group of sixteen companies had commonality in the fact that they agree with the following statements:

2) We involve our forge supplier(s) in the design of our forged products at the earliest opportunity

7) We look for suppliers who can provide us with all our manufacturing needs

8) We require a line side delivery service for forged parts

11) We prefer to buy forged products made in the UK

and always observe the following:

1) We expect delivery of forged products strictly to schedule

4) We would prefer an accurate indication of lead time

5) We would like suppliers to advise us of any problems with supply

The same decoding could be carried out for the other two clusters generated. Had we taken the smallest cluster from table 8.9, the group of eleven companies would have disagreed with three statements and always observed a further two. This information can then be used to produce the required competitive advantage.

8.6 Supervised Analysis Results and Comments

During the development of the IMS paradigm, the Vector Memory Array method of supervised classification was discovered. Details of the operation of VMA can be found in Chapter 4.

To facilitate the bench marking of VMA, it was decided to use data from the Warwick Electronic Nose consisting of twelve Figaro gas sensors. Table 8.10 shows the sensor array used. This data has been analysed using a number of different networks. The number of distinct groups and the overlap between them was known. Two data sets are considered here, the first being from the analysis of five alcohols. This data was known to contain five distinct well-separated groups representing five simple odours. The application of back propagation network paradigms to this data has produced 100% accuracy. The second data set was obtained from the analysis of the complex odours from commercial coffees. This data contained three distinct groups, known not to be well separated. Back propagation techniques on this data have failed to converge; some self-organising techniques have achieved high separation percentages but unreliably. Alpaydin's Constructive Learning Algorithm achieved a very low accuracy. Details of the structure of the data sets are given in Chapter 7.

No.	1	2	3	4	5	6	7	8	9	10	11	12
TGS	813	831	815	816	823	800	842	882	881	883	825	821

Table 8.10 Types of commercial Taguchi gas sensors (TGS) used in Electronic Nose

Using VMA (with any of the metrics) to analyse the alcohol data, 100% classification accuracy was achieved in a few seconds running on a Sun SPARC IPC workstation. All other techniques had taken at least tens of minutes.

On analysing the coffee data VMA achieved a maximum of 92% accuracy (using the city block metric) also in a few seconds. Back propagation techniques had failed to converge on this data, while linear discriminant function analysis produced a success rate of 82% (for references see chapter 7). The results obtained for the analysis of the coffee data using two metrics can be seen in figures 8.2 & 8.3.

In the figures 8.2, 8.3 the parameter 'a' has been set to either exclude, include or amplify the effect of the 'log term' (equation 8.1) . The effect that the 'log term' has depends on the nature of the data set, but it can be seen that in both cases the highest accuracy was achieved with the lowest group sizes and with the 'log term' either absent or unamplified. The modified city block metric and the 'angle between two vectors' metric produced similar results but the accuracy was reduced in both cases.

$$F = a\ln(N) + \frac{1}{N}\sum_{i=0}^{N-1}\frac{1}{x_i} \tag{8.1}$$

Although the traces in figures 8.2, 8.3 follow the same general trend (as group size increases, accuracy falls), it can be seen that the effect of the 'log term' changes with group size. The overall effect of amplifying the 'log term' (a=10) is to reduce accuracy with larger group sizes. The inclusion of the an unamplified 'log term' (a=1) gives slightly improved overall accuracy. As group size increases, each new vector included can alter the classification accuracy if it changes the classification of any of the outlying vectors. This combined with the effect of the 'log term' leads to the trace crossovers seen in figures 8.2, 8.3.

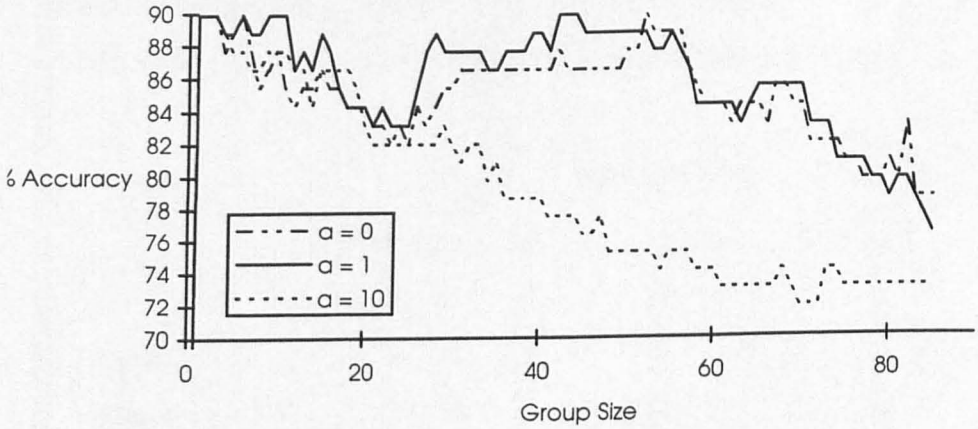


Figure 8.2. A plot of group size against percentage accuracy using the Euclidean distance metric for the original coffee data

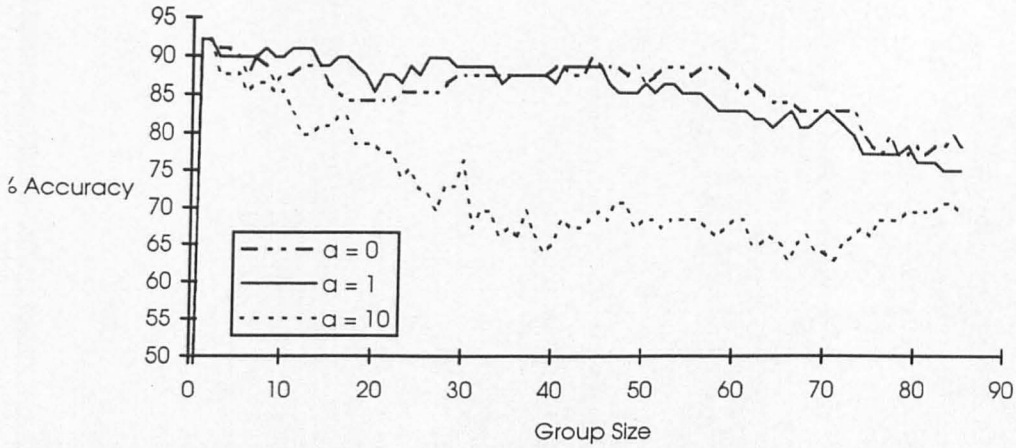


Figure 8.3. A plot of group size against percentage accuracy using the city block style distance metric for the original coffee data

Tables 8.11 and 8.12 show confusion matrices for analysis carried out on the coffee and alcohol data sets. In these tables the difference between the quality of the results obtained can be clearly seen: perfect results being produced for the alcohol data, whilst the more complicated coffee data has been classified less accurately.

		Desired Class				
		1	2	3	4	5
	1	8	0	0	0	0
Actual	2	0	8	0	0	0
Class	3	0	0	8	0	0
	4	0	0	0	8	0
	5	0	0	0	0	8

Table 8.11 A confusion matrix produced using the Euclidean distance metric on the original alcohol data.

		Desired Class		
		1	2	3
Actual	1	28	1	1
Class	2	1	29	0
	3	2	2	25

Table 8.12. A confusion matrix produced using the city block style metric on the original coffee data.

Chapter 9
Conclusions and Discussions

9.1 The Research Objective

The objective of this research was to locate clusters in market research interview data that contained definite commonality. As the problem was restricted to binary data sets, commonality could be defined in terms of all the vectors within a cluster having a logic state of '1' for a specific bit position.

9.2 The Development of this Body of Research

The two schools of mathematics suitable for pursuing a solution to this problem are artificial neural networks and statistical clustering algorithms.

Work by the author progressed through three phases of development once the problem had been clearly defined (see Chapter 6). Kohonen's self-organising maps (SOM) [1] (a neural network technique) plots a multidimensional data set onto a two dimensional surface, giving an "ordering" to the data in the process. This technique seemed theoretically capable of meeting the objective. The problem encountered was that the map surface produced follows the distribution of the data set (this is intended); this made interpretation of positions on the map surface difficult. To solve this problem the Euclidean Memory Array (EMA) [2] was developed. EMA mathematically derived its weights to avoid the above problem. This approach proved relatively unproductive since the output in some cases then became like a badly scaled graph. This approach to deriving weights led to the development of the Vector Memory Array (VMA) [3], a supervised technique that although unsuitable for meeting the research objective has compared favourably in trials against other

algorithms. VMA used the actual data set as the weight values for the network. Further development to the use of the main principle of VMA (using the data set as weight values) in an unsupervised technique led to the conception of Interrogative Memory Structures (IMS) [4]. As an unsupervised technique, IMS is capable of meeting the research objective. IMS has been tested against SOM, Adaptive Resonance Theory (ART1, a neural network paradigm developed by Grossberg [5]: see Chapter 2) and FASTCLUS [6] (a statistical Euclidean distance-based clustering algorithm, developed by the SAS Institute).

9.3 Comparison of Techniques

Tables 9.1 and 9.2 give qualitative comparisons of VMA with Back Propagation (BP) [7], and IMS with SOM, ART1 and FASTCLUS. It would have been more desirable to be able to give quantitative comparisons in both cases but this is not feasible due to the complexity of the issues. The performance of the paradigms depends on the size and type of complexity of the data sets for criteria such as memory usage, speed and accuracy, and so there are no easy overall qualitative measures that can be used.

Table 9.1 shows a comparison of VMA with BP. From the description given of VMA in Chapter 4, it can be seen that the technique stores the entire data set in memory and is therefore rather wasteful. In the trials carried out (see Chapter 8) VMA performed with extreme speed of operation, classifying data sets in seconds that took BP tens of minutes. The classification accuracy was also very high,

performing at least on a par with BP and on one data set producing results of greater than 90% accuracy when BP was often failing to converge on any solution. Given that the trend in modern computing is for ever increasing memory, the memory utilisation of VMA should not prove a problem.

VMA	BP
Very fast	Slow
High accuracy	Unpredictable/ unreliable accuracy
Easy to use	Difficult to know when to stop training
Can easily incorporate new data	Cannot easily incorporate new data
Can use large amounts of memory if large data set is used.	Memory efficient with large data sets.

Table 9.1 A comparison of VMA with BP.

Tables 9.2 shows a comparison of IMS with SOM, ART1 and FASTCLUS in terms of:

(a) Ease of use

(b) Ease of understanding

(c) Ease of interpretation

(d) Memory usage

(e) Speed of operation

(f) Use of optimisation

(g) Ability to handle binary/ non binary

As with VMA, IMS stores the whole data set, and can therefore be said to be wasteful in terms of memory usage. In this comparison, IMS and ART1 are specifically designed for use on binary data sets. The main advantages of IMS were in terms of its ease of use (only equalled by FASTCLUS), speed of response (again only equalled by FASTCLUS) and ability to tailor the optimisation used to the problem. IMS consistently produced clusters containing the highest degree of commonality. When general metrics (see Chapter 6) were used to compare the techniques, ART1 gave the best performance.

(a)

IMS
(a) Easy to use
(b) Easy to understand
(c) Easy to interpret results
(d) Stores entire data set in memory
(e) Fast
(f) Only Binary
(g) Can select optimisation

(b)

SOM
(a) Moderate ease of use (several parameters require setting, by trial and error)
(b) Easy to understand
(c) Difficult to interpret results due to distribution of map nodes in data space
(d) User selects how much memory is used by setting size of map layer

(e) Slow (uses iterative learning rule)
(f) Not limited to binary
(g) No optimisation possible (vectors ordered by Euclidean distance)

(c)

ART1
(a) Moderate ease of use (some parameters require setting by trial and error)
(b) Difficult to understand (operation is mathematically complex)
(c) Easy to interpret results
(d) User selects how much memory is used by setting number of nodes in F2 layer
(e) Slow (due to resonant period required to change weight values)
(f) Limited to binary (but non-binary version exists, ART2)
(g) Optimisation through adjustment of vigilance threshold (by trial and error)

(d)

FASTCLUS
(a) Easy to use (max. number of clusters requires setting by trial and error)
(b) Easy to understand
(c) Easy to interpret results
(d) Moderate memory usage (dependant on size of data set)
(e) Fast
(f) Not limited to binary
(g) No optimisation available (only adjustable parameter is max. number of seeds)

Tables 9.2 A comparison of IMS, SOM, ART1 and FASTCLUS

9.3.1 Comments

To aid clarity, comments on VMA (a supervised technique) and IMS (an unsupervised technique) are addressed separately. It can be seen however that the advantages of the techniques in their individual scope of operation are the same.

VMA

With adaptation to the VMA algorithm to remove redundant vectors from the matrix layer, its memory usage should decrease. This will have the effect of further speeding computation time as the network will be smaller. The main advantages of VMA are:

- Ease of use
- Speed
- Accuracy

IMS

IMS has consistently produced clusters with the highest degree of commonality. Recent work has shown that IMS can successfully be used in the area of manufacturing systems Group Technology to produce the part families. The main benefits of IMS in the application area of market research is:

- Ease of use
- Speed
- Degree of commonality in clusters located

9.4 Future Work

VMA and IMS have both proved to be extremely successful. A number of easy improvements could bring a further boost to the performance of both algorithms. The addition of a pruning algorithm to remove redundant vectors from the arrays could decrease computation time for both techniques. This would have the added benefit of reducing the amount of memory used in each case. Further exploration of transfer functions for the output nodes on VMA and the optimisation strategies for IMS could yield increases in terms of accuracy.

The analysis of market research data could be taken further by modifying IMS to cope with non-binary representations. A coding system for representing the interview forms in a non-binary format could also be investigated.
The testing of both IMS and VMA in other application areas would also yield useful information about their performance.

9.5 References

[1] Kohonen T: "Self-organisation and associative memory"; Third Edition: Springer-Verlag, New York, 1989.

[2] Larkin A B, Hines E L and Thomas S M; "The Euclidean memory array, a vector quantisation technique for the processing of data from interview forms"; Neural Computing and Applications (1994) 2:53-57 Springer-Verlag London Ltd.

[3] Larkin A B, Hines E L, Thomas S M and Gardner J W; "Supervised learning using the vector memory array method"; Proceedings of the Workshop on Neural Network Applications and Tools. Sept 13-14 1993, Liverpool England. Ed. Lisboa P, Taylor M. IEEE Computer Society Press, ISBN 0-8186-5845-2, pp. 6-10.

[4] Larkin A B; "Clustering with interrogative memory structures"; Submitted to Neural Computing and Applications; Springer-Verlang.

[5] Carpenter G A and Grossberg S; "The ART of adaptive pattern recognition by a self-organising neural network"; IEEE Computer Magazine; March 1988; pp. 77-88.

[6] SAS/ STAT Users Guide, Vol. 1, Ver. 6, Forth Ed; SAS Institute INC, SAS Campus Drive, Cary NC 27513; pp. 823 - 850.

[7] Rumelhart D E, Hinton G E and Williams R J; "Learning internal representations by error propagation"; In (Rumelhart D E, McClelland J L) Parallel Distributed Processing Vol 1: The MIT Press, Cambridge, MA, 1986, pp.318-362.

PAGES
NOT SCANNED
AT THE REQUEST OF
THE UNIVERSITY

SEE ORIGINAL COPY
OF THE THESIS FOR
THIS MATERIAL