C321

MONASH UNIVERSITY
THESIS ACCEPTED IN SATISFACTION OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
ON..... 4 October 2002 .....
.....[redacted].....
Sec. Research Graduate School Committee

Data Mining with Structure Adapting Neural Networks

by

Lakpriya Damminda Alahakoon BSc.(Hons)

A thesis submitted in fullfilment of the requirements for

the degree of Doctor of Philosophy

School of Computer Science and Software Engineering

Monash University

March 2000

To

Amma & Thaththa

Abstract

A new generation of techniques and tools are emerging to facilitate intelligent and automated analysis of large volumes of data and discovering critical *patterns* of useful knowledge. Artificial neural networks are one of the main techniques used in the quest for developing such *intelligent* data analysis and management tools. Current artificial neural networks face a major restriction due to their *pre-defined* network structures which are *fixed* through the life cycle. The breaking of this barrier by models with adaptable structure, can be considered as a major contribution towards achieving truly intelligent artificial systems.

Kohonen's Self Organising Map (SOM) is a neural network widely used as an exploratory data analysis and mining tool. Since the SOM is expected to produce topology preserving maps of the input data, which are highly dependent on the initial structure, it suffers from the pre-defined fixed structure limitation even more than the other neural network models. In this thesis, the main contribution is the development of a new neural network model which preserves the advantages of the SOM, while enhancing its usability by an incrementally adapting it's architecture.

The new model, called the Growing Self Organising Map (GSOM), is enriched with a number of novel features resulting in a mapping which captures the topol-

ogy of the data, not only by the inter and intra cluster distances, but also by the shape of the network. The new features result in reducing the possibility of twisted maps and achieves convergence with localised self organisation. The localised processing and the optimised shape helps in generating representative maps with smaller number of nodes. The GSOM is also flexible in use, since it provides a data mining analyst with the ability to control the spread of a map from an abstract level to a detailed level of analysis as required.

The GSOM has been further extended using its incrementally adapting nature. An automated cluster identification method is proposed by developing a data skeleton from the GSOM. This method extends the conventional visual cluster identification from feature maps, and the use of a data skeleton, in place of the the complete map, can result in faster processing. The control of the spread of map, combined with the automated cluster identification, is used to develop a method of hierarchical clustering of feature maps for data mining purposes.

The thesis also proposes a conceptual model for the GSOM. The conceptual model facilitates the extraction of rules from the GSOM clusters thus extending the traditional use of feature maps from a visualisation technique to a more useful data mining tool. A method is also proposed for identification and monitoring change in data by comparing feature maps using the conceptual model.

Declaration

This thesis contains no material that has been accepted for the award of any other degree or diploma in any university or other institution. To the best of my knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

L. D. Alahakoon

Acknowledgments

I am grateful to my principal supervisor Professor Bala Srinivasan for his able guidance and intellectual support throughout my research work. I am also grateful to my second supervisor Dr. Saman Halgamuge (University of Melbourne) for the valuable advice, guidance, moral support and friendship during this period. They were both available whenever I needed advice and helped to maintain the research work on a steady course over the years.

I thank Dr. Arkady Zaslavsky, Postgraduate coordinator, School of Computer Science and Software Engineering, for providing financial support during difficult times. I also take this opportunity to thank Ms. Shonali Krishnaswamy for the friendship during these years we shared an office, and also my colleagues and friends, Mr. Monzur Rahman, Dr. Phu Dung Lee, Mr. Campbell Wilson, Dr. Maria Indrawan, Mr. Santosh Kulakarni, Mr. Robert Redpath and Mr. Salah Mohammed. They provided a friendly and helpful atmosphere where I managed to *fit in*, and made it easier to get used to the new environment. I thank Mr. Duke Fonias for the excellent technical support and help throughout my PhD candidature.

Finally, I am grateful to my wife Oshadhi, and daughters Navya and Laksha for their love, kindness and tolerance during the *PhD years*.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Exploratory Data Analysis for Data Mining

During the last decade, there has been an explosive growth in the ability to both generate and collect data in electronic format. Advances in scientific data collection technology, the widespread use of bar codes in commercial products, computerisation of many businesses and government transactions has generated a massive amount of data. Advances in data storage technology has produced faster, higher capacity and cheaper storage devices. This expansion and advancement in technology has resulted in better database management systems and data warehousing systems which have transformed the data deluge into massive data stores [Fay96b], [Inm97], [Inm98]. Although advances in computer networking has enabled large number of users to access these vast data resources, corresponding

advances in computational techniques to analyse the accumulated data has not taken place [Che96], [Mat93], [Fay96a].

Raw data is a valuable organisational resource as it forms the basis for higher level information being used for strategic decision making. In scientific activities, data may represent observations collected about some phenomena being studied. In business, raw data captures information about the markets, competitors and customers. In manufacturing such data may represent performance and optimisation opportunities. Such information is useful for decision support, or for exploration and better understanding of the underlying phenomena which resulted in generating the data. Traditionally data analysis was performed manually using statistical techniques. Such approaches have become unusable and obsolete as the volume and dimensionality of the data increased. The situation is further complicated by the rapid growth and change occurring in the data. Hence tools that can at least partially automate the analysis task have become a necessity. The field of data mining (DM) and knowledge discovery (KD) has emerged as an answer to this need and looks at new approaches, techniques and solutions for the problems in anlysing large databases [Kno96], [Kei96].

There are different approaches that can be employed for data mining. Techniques that have been popularly used are classification, clustering, estimation, prediction, market basket analysis and description [Ber97b]. A number of tools such as

statistics, decision trees, neural networks, genetic algorithms have been used to implement these techniques [Ber97c], [Ken98], [Ber97b]. In most cases a combination of such tools and techniques will be needed for achieving the goals of data mining [Wes98].

During a data mining operation, irrespective of techniques and tools being used, it is generally a good practice to initially obtain an unbiased view of the data. Unlike situations where an analyst may use standard mathematical and statistical analysis to test predefined hypothesis, data mining is most useful in exploratory analysis scenarios where an *interesting* outcome is not predefined [Wes98]. Therefore data mining can be carried out as an iterative process within which progress is defined and directed by *discovery* of interesting patterns. The analyst can start by obtaining an overall picture of the available data. This overall understanding can then be made use of to perform better directed and planned modeling and analysis of subsets of the data. This allows the initial knowledge obtained about the data to support better planning and utilisation of one or more data mining tools and techniques. Therefore, whatever form the analysis takes, the key is in adopting a flexible approach that will allow the analyst to make unexpected discoveries beyond the bounds of the established expectations within the problem domain [Wes98].

To achieve such flexibility and independence from pre-conceived bias on the property of the data, we propose that the data mining task be conducted in two main steps.

1. Initial exploratory analysis phase

2. Secondary directed analysis phase.

These steps can be used a number of times with different order of precedence, but the data mining task should begin with an initial exploratory analysis. Even when well defined goals and targets exist, such exploratory analysis may provide unexpected outcomes which can result in the change or modifications to the initial plan. Due to the importance attached to the initial exploratory analysis, this research has been focussed on the development of better and improved methods for such analysis.

Currently the popular methods used for exploratory data analysis are visualisation techniques, clustering techniques and unsupervised learning neural network models. Visualisation techniques have recently become popular and a large number of commercial software has appeared [Wes98]. Clustering techniques have been traditionally used and are still being improved and enhanced [Fis95], [Wan97], [Zha97]. Unsupervised neural network models have been used in the past for pattern recognition and clustering applications and recently have been recognised for their usefulness in data mining applications [Deb98], [Law99]. The

Kohonen Feature Map or the Self Organising Map (SOM), is currently the most widely used unsupervised neural network model and has been used for different applications ranging from image processing [Suk95], [Ame98], speech recognition [Kan91], [Sch93], engineering [Rit93], [koh96b], pattern classification and recognition [Bim96], [Jou95] and recently for data mining [Deb98], [Ult92]. The SOM generates mappings from high dimensional input space to lower dimensional (normally two) topological structures. These mappings not only produce a low dimensional clustering of the input space, but preserves the topology of the input data structure. That is, the neighbourhood relationship in the input data is preserved. Also it has the property that regions of high density or population corresponds to large parts (in terms of size) of the topological structure. Therefore the output mapping of the SOM provides dimensionality reduced visualisation of clusters. The relative *positions* and *spread* of the clusters provide the inter and intra cluster attribute information which can be used to *understand* the data. These properties have made the SOM an attractive tool of exploratory data analysis and mining.

Similar to the other traditional neural network models, the *structure* of the SOM has to be defined at the start of training. It has been shown that a predefined structure and size of the SOM has limitations on the resulting mapping [Fri94], [Lee91]. If we consider the SOM as a lattice of nodes, the degree of topology preservation depends on the choice of lattice structure [Vil97]. Therefore it may

only be at the end of training that the analyst realises that a different shape or number of elements would have been more appropriate for the given data set. As such the user has to try different lattice structures and determine by trial and error which lattice yields the highest degree of topology preservation. Although applicable to most situations, this *structure limitation* is even more relevant to data mining applications since the user is not aware of the structure in the data, which makes it difficult to choose an appropriate lattice structure in advance. The pre-definition of structure *forces* user bias onto the mapping and therefore limits the advantages for exploratory data mining.

A solution to this dilemma is to determine the shape as well as size of the network during training in an incremental fashion. Although the fixed structure constraint applies to all traditional neural networks, lattice structure neural networks suffer more from this limitation (discussed in section 1.2 and in chapter 5 in more detail), and also has the potential to gain more advantages from incrementally generating architectures. As such the incremental structure adaptation in self organising neural networks is the main focus of this research work.

In section 1.2 we discuss the fixed structure limitation as a problem affecting most neural networks which restrict their ability for intelligent behaviour. This discussion is then used to highlight the proposed work on structure adapting self

organising maps as one branch of the more general problem of fixed structure affecting most current neural network models.

1.2 Structure Adapting Neural Networks for Developing Enhanced Intelligent Systems

Neural networks as a field of scientific research can be traced back to the 1940's. During the early days *the holy grail* of neural network researchers was the development of an artificial model which can simulate the functionality of the human brain. After the initial euphoria died down, the goals have been re-specified, and the main aim of the current neural network community is the development of models which can surpass the functionality of traditional von Neumann computers with the ability to learn. Such *learning* is expected to provide the neural network with *intelligence* which can be harnessed for the benefit of human beings. One way of measuring the intelligence of an artificial (non-human) system is the amount and type of useful behaviour it can generate without human intervention. This can be further interpreted as, the lesser the human intervention required by an artificial system, the more it can be considered intelligent.

Current (traditional) artificial neural network models allow the networks to adjust their behaviour by changing the interconnection weights. The number of neurons and the structural relationships among the neurons have to be pre-specified by the

neural network designers, and once the structure is designed, it is fixed throughout the life cycle of the system. Therefore the learning ability of the system and as such it's intelligence is constrained by the network structure, which is pre-defined. Due to this constraint the network designer faces the difficult, sometimes impossible task of figuring out the optimum structure of the network for a given application. A solution to this problem is to let the neural network decide it's own structure, according to the *needs* of the input data. This leads to the concept of adaptable structure neural networks where the constraint of pre-specified fixed structure is removed. Such adaptable structure neural networks requires less human intervention and therefore can be called *more intelligent* compared to their human specified fixed structure counterparts.

After the early days of neural network research, the development of the field was restricted by the *single layer constraint* as highlighted by Minsky and Papert [Min69]. A revival of neural network research occurred due to breaking of the single layer barrier by developing *multi layer learning algorithms*. We consider the constraint of fixed pre-definable structures as a similar barrier which needs to be broken for the research to progress further. Therefore the concept of structurally adaptable neural networks can be considered as a major leap towards achieving the final goal of neural network researchers - *viz. the development of truly intelligent artificial systems.*

1.3 Motivation for the Thesis

The Self Organsiing Map (SOM) is a neural network model that is capable of projecting high dimensional input data onto a low dimensional (typically 2 dimensional) array (or map). This non-linear projection produces a two dimensional feature map that can be useful in detecting and analysing features in the input data in which specific classes or outcomes are not known a priori, and hence training is done in unsupervised mode. In data mining applications, the SOM can be used to understand the structure of the input data, and in particular, to identify clusters of input records that have similar characteristics in the high dimensional input space. An important characteristic of the SOM is the capability to produce a structured ordering of the input data vectors called *topology preservation*. Similar vectors in the input space are mapped to neighbouring nodes in the trained feature map. Such topology preservation is particularly useful in data analysis and mining applications where the relative positions of the clusters contain information about their relationships.

The main problem that can be identified when using the SOM as a data mining tool is the need for pre-defining the network structure and the fixed nature of the network during its life cycle. The predefined fixed structure problem of the SOM can be described in detail as follows.

1. One of the main reasons for using the SOM as a data mining tool is that it is an unsupervised technique and as such does not require *knowledge* about the data for training. However, the need to define the *proper* size and shape of the SOM at the outset implies the need to understand the data, which limits the advantage of unsupervised learning. Further, having to *fix* the network structure in advance results in the development of non-optimal SOMs where the ideal cluster structure may not be visible.

2. To achieve proper topology preservation, the SOM has to be initially built with appropriate length and width (in a two dimensional map) to match the input data. When the network is not built to the ideal shape, a distorted view of the topological relationships may occur, thus restricting its usage as a data mining visualisation tool. Such a distorted picture can also provide false information resulting in suboptimal or even wrong conclusions on a data set.

3. When it is required to map an *unknown* data, the network designer may have to build a large SOM which can accommodate any input set of the application domain. Such networks usually tend to be larger than required and result in slow processing time thereby wasting system resources.

The need and timeliness of data mining is highlighted in section 1.1 and the need for structure adapting neural networks is discussed in section 1.2. Therefore the work towards the development of an adaptable SOM structure can be considered as

1. The development of a novel neural network model which preserves the existing usefulness and advantages of the SOM, but removes the major limitation of pre-defined fixed structure. Such a model has advantages for all applications of the SOM, and will have special advantages for data mining.

2. In the wider interest of the neural network community, the work towards breaking the fixed structure barrier, thus a step in the direction of achieving truly intelligent artificial systems.

1.4 Objectives of the Thesis

The development of structure adapting neural networks is an area of research which has high potential for producing better intelligent systems [Qui98]. Several developments in the area of supervised adaptable neural network structures have been reported using derivatives of the back propagation algorithm. The main categorisations of such work are, node removal methods from networks [Moz89], [Bro94], [Sic91] and adding nodes and new connections between nodes [Han90], [Han95], [Ash95a], [Hal97] and the adding and pruning of nodes [Bar94]. A formalisation of the concept of structure adapting neural networks is developed by

Lee [Lee91]. Although these models have been developed and improved during the past decade, no major real life applications have yet been reported.

On the contrary, the developments in the area of unsupervised models have been mainly limited to attempts at extending the SOM [Fri94], [Lee91], [Mar94], [Bla95]. Similar to the supervised models, none of these work has been applied to *real life* applications. The main factor preventing practical applications is the complexity of these modified algorithms which are resource intensive and do not justify their usage with large and complex input data.

In the research described in this thesis we present a novel neural network model which has features that can be exploited for data mining. The objectives in developing such a new model are as follows.

1. Introduce adaptable architecture as a major step towards achieving *intelligent* neural networks and highlight the advantages of such a structure adapting SOM for the current needs of data mining applications.

2. Develop a neural network based on the concept of self organisation, which can claim similar advantages as that of SOM for data mining applications, and provide it with the ability to self generate the network during the training phase, thus removing the necessity for the network designer to pre-specify the network structure.

3. Provide the new model with flexibility to *spread out* according to the structure present in the data. Therefore the network has the ability to highlight the structure in the data set by *branching out* in the form of the data structure.

4. Provide the data analyst with better control over the spread of the projected input mapping.

5. Preserve the simplicity and ease of use of the SOM in the new model, such that the new model can be used in real applications.

In addition to the main objectives presented above, we explore the incrementally generating nature of our new model to extend the traditional usage of feature maps for data mining applications. Such extensions can be justified since the current usage of the SOM has been limited only to providing visualisation of the clusters whereas there is potential for exploiting the feature maps for other uses. The thesis explores this proposed model further with the following expectations.

1. Automated cluster identification and separation over current visual identification.

2. Stepwise hierarchical clustering using the new model, providing the ability to handle large data sets, and more flexible analysis by the user.

3. Rule generation from feature maps, thus breaking the traditional *black box* limitation of neural network models.

4. Ability for data monitoring and identifying change in data using the new model.

1.5 Main Contributions of the Thesis

The main contribution of this thesis is the development of a neural network model (called the Growing Self Organising Map - GSOM) with the ability to incrementally generate and adapt its structure to represent the input data. The major difference of this approach with the traditional neural networks is that the new model has an *input driven architecture* which has the ability to represent the data by the shape and the size of the network itself. Therefore the network designer does not have to predefine a *suitable* network since an optimal network is self generated during the training phase. The new model not only *relieves* the network engineer from the difficult task of predefining the structure, but also produces a better topology preserving mapping of the data set due to the network structure flexibility. Therefore the input clusters are better *visible* from the map itself and the inter and intra cluster distances are topologically more meaningful. Such better topology preservation is achieved since the clusters can spread un-restricted by fixed *borders* as in the case of conventional SOM neural networks. Several new concepts and features are introduced to achieve the self generating ability and these are given below.

1. A new learning algorithm for weight adaptation of the GSOM which takes

into consideration the dynamic nature of the network architecture thus changing with the number of nodes in the network.

2. The introduction of a heuristic criteria for the generation of new nodes in the network. The new nodes are added without loosing the two dimensional representation of the map, thus maintaining the ease of visualisation.

3. A method of initialising the weights of newly generated nodes such that they will smoothly merge with the already *partially trained* network.

4. A method of localised weight adaptation making use of the weight initialisation method in (3). Localised weight adaptation results in faster processing as lesser number of nodes are considered for weight adjustment.

5. A new method of distributing the *growth responsibility* of non boundary nodes such that the number of nodes in the network can proportionally represent the input data distribution. Such proportional representation results in a better 2 dimensional topological representation.

6. Introduction of a novel concept called the spread factor for controlling the spread of the network. The spread factor can be provided as a parameter and lets the data analyst decide the *level of spread* at a certain instance in time.

In addition to the main contribution of developing a new model, the following extensions are developed making use of the incrementally generating nature of the GSOM. These additional contributions are

1. A new concept called data skeleton modeling which can identify the paths joining the clusters in a network. The data skeleton is used to develop a method for automating the cluster identification and separation process.

2. Formalisation of the spread factor as a better method of increasing the spread of a map compared with the traditional method of increasing map length and width.

3. The use of spread factor in developing an algorithm for hierarchical clustering of the GSOM.

4. Development of a Attribute Cluster Relationship (ACR) model to provide a conceptual model of feature maps for addressing the current problem of comparing feature maps.

5. The use of ACR model for generating rules from the feature map clusters. A new idea called *query by attribute rule* is introduced whereby the data analyst can query the clusters and generate new rules.

6. A new method is proposed for identify movement or change in the data using the GSOM clusters and the ACR model. Identification of such shift

in the data is proposed as a useful way of monitoring trends in the data values for data mining.

7. Proposal for extending the *crisp* rules generated from the ACR model into fuzzy rules for more human understandability and generalization.

1.6 Outline of the Thesis

Chapter 2 provides the background for the concepts and methods discussed in this thesis. Since our main focus is on adaptable structure neural networks, this concept is described in detail. Since SOM is used as a base for developing the new neural network model, the traditional SOM algorithm is discussed at length in this chapter. A review of past work on adaptable network models is also presented highlighting their limitations for *real life* applications, specially for data mining.

Chapter 3 describes the development and implementation of the new Growing Self Organising Map (GSOM) model. The GSOM is built using the concepts of self organisation and also attempts to improve upon similar work in the past to arrive at a more practically usable model. The new features of the GSOM are described in detail with justifications in the section on parameterisation of the GSOM. Experimental results are presented comparing the GSOM with the traditional SOM. Advantages of the GSOM due to it's ability to attract attention

to the clusters by *branching out* is illustrated experimentally.

In chapter 4, the GSOM is used to develop a method for automating the cluster identification and separation. The paths of spread of the GSOM is initially identified to develop a *skeleton* of the data clusters. Parts of the skeleton called the path segments are removed to separate the clusters using inter cluster *distances*. The new method provides an efficient and more accurate way of cluster identification from feature maps compared to the traditional visualisation.

Chapter 5 highlights the usage of the spread factor for controlling the GSOM. At the beginning of any data analysis, a low spread factor will produce an abstract overview of the input data structure. The analyst can use such a map to obtain an initial understanding of this data. For example, *interesting regions* of the input space may be identified. These regions can then be further spread out for detailed analysis with a higher spread factor. Initially the use of spread factor in GSOMs is compared with the traditional method of network size enlargement in the SOM. The problems with the traditional SOM is described and the use of the spread factor is presented as a better alternative. The spread factor is then made use of to develop an algorithm for hierarchical clustering with the GSOM. The advantage of such hierarchical clustering for data mining applications is discussed using experimental results.

Chapter 6 describes a further extension to the GSOM by developing a conceptual layer of *summary* nodes on top of the GSOM. The conceptual layer is then used as a base for an Attribute Cluster Relationship (ACR) model which provides a conceptual view of the relationships between clusters. The ACR model addresses the problem of changeable cluster *positions* of feature maps. Due to this problem it becomes difficult to compare different feature maps. Using the ACR model, an algorithm is developed to compare feature maps and to identify change and movement in data. Chapter 6 also describes a method for rule extraction from the GSOM clusters using the ACR model. Such rule generation can be considered as extending the traditional usage of feature maps, from a tool for cluster identification, to a more complete data mining tool.

In chapter 7 the rule extraction using the ACR model is extended by interpreting the cluster summary nodes as fuzzy rules. The extension facilitates the generation of fuzzy rules from the new fuzzy ACR model. The fuzzy rules provide the ability for the data analyst to generate rules of a more abstract and general nature compared to the *crisp* rules described in chapter 6. The fuzzy rules also provide a more *human like* interpretation of the data clusters. The advantage of the GSOM-ACR model over other neuro fuzzy models is that the initial clusters are not biased by users, since they are self generated by the GSOM.

Chapter 8 provides the concluding remarks of the thesis. A summary of the work described in the thesis is presented. Areas and problems for future work are identified in this chapter.

Chapter 2

Structural Adaptation in Self Organising Maps

2.1 Introduction

The main contribution of this thesis is the development of a structure adapting neural network model based on the SOM, which has specific advantages for data mining applications. The purpose of this chapter is to provide the background knowledge on the SOM and the concepts of structural adaptation in neural networks. A brief introduction to data mining and the use of feature maps for data mining is also discussed. The chapter also provides a review of the past and existing work on the development of structurally adapting unsupervised neural network models.

Section 2.2 of this chapter provides a brief introduction to neural networks as an introductory step for section 2.3 where the SOM is presented and discussed in detail. Section 2.4 introduces the concept of data mining and discusses the SOM as a data mining tool. In section 2.5 we present the concept of structural adaptation in neural networks and section 2.6 describes the advantages of such structure adapting neural networks for data mining. Section 2.7 presents a review of past work on developing structurally adapting unsupervised neural network models, and section 2.8 provides the summary for the chapter.

2.2 Neural Networks

Research in the field of artificial neural networks has attracted increasing attention in recent years. Since 1943, when Warren McCulloch and Walter Pitts presented the first model of artificial neurons [Ash43], new and more sophisticated proposals have been made from decade to decade. Mathematical analysis has solved some of the mysteries posed by the new models but has left many questions open for future investigations [Roj96]. A very important feature of artificial neural networks is their adaptive nature, where *learning by example* replaced the *programming*. This feature makes such computational models very appealing in applications where one has little or incomplete understanding of the problem to be solved but where training data is readily available [Has95].

In all neural network models, a single neuron is the basic building block of the network. The operation of a single neuron is modeled by mathematical equations, and the individual neurons are connected together as a network. Each neural network has its learning laws according to which it is capable of adjusting the parameters of the neurons. This allows the neural network to learn [Sim96].

In most neural network models, the operation of a single neuron can be divided into two separate parts as, a weighted sum and an output function as shown in Figure 2.1.

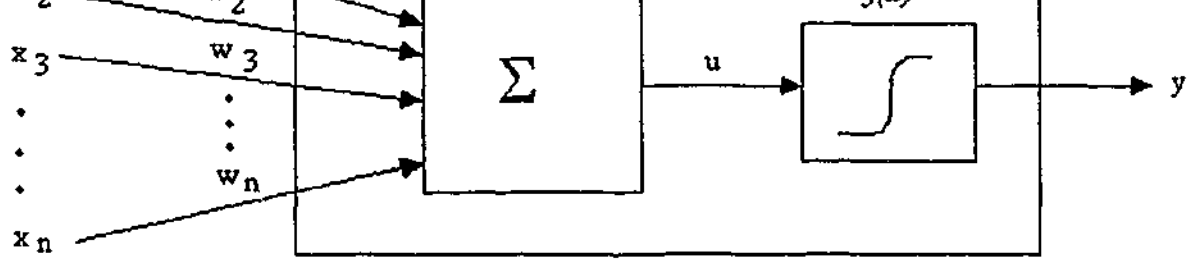


Figure 2.1: A typical artificial neural network

The weighted sum computes the activation level u of the neuron, while the output function $f(u)$ gives the actual output y of the neuron. The operation of the model shown in Figure 2.1 is as follows. Initially the inputs $x_i, i = 1, 2, \ldots, n$ are summed together according to a weighted sum as

$$u = \sum_{i=1}^{n} w_i \cdot x_i \tag{2.1}$$

where w_i are the weights of the neuron for the i^{th} input. Next, the activation level u is scaled according to the output function. The sigmoid is the most commonly used output function [Loo97], [Gur97], and can be expressed as

$$y = f(u) = \frac{1}{1 + e^{-cu}} \tag{2.2}$$

where c is a positive constant which controls the *steepness* (slope) of the sigmoid function. The sigmoid function amplifies small activation levels, but limits high activation levels. In practice, the output y has a value between $[0, 1]$. If negative outputs are also required, the hyperbolic tangent can be used in place of the sigmoid function as

$$y = f(u) = tanh(cu) = \frac{e^{cu} - e{-cu}}{e^{cu} + e^{-cu}} \tag{2.3}$$

The individual neurons are usually connected together as layers. The first layer is called the *input layer*, the following layers are called the *hidden layers* and the last layer is the *output layer*. The input layer does not usually process the input signals, but rather distributes them to the next layer. Depending on the neural network model, the number of hidden layers can vary from zero to several layers. A neural network with no hidden layers is called a *single layer network*, and a network containing at least one hidden layer is called a multi-layer network. Finally, the output layer gives the output of the network.

A neural network can either be *feed forward* or *recurrent* type network [Roj96]. The difference between these two is that the feed-forward type network has no

feed back connections while the recurrent type network has at least some feed back connections. Usually the feed-back connections are made from the output layer to the input layer.

The learning algorithms of different neural network models can be divided into two major categories: *supervised* and *unsupervised* learning algorithms. In supervised learning algorithms, the correct output is known during the training procedure. The task of the learning algorithms is to minimise the errors between the output of the neural network and the training data samples. In unsupervised learning algorithms, the correct output is unknown during the training process. After the training is over, the correct output can be used to label the neurons of the output layer such that the network gives correct outputs during normal operation. Unsupervised learning methods are better suited for exploratory data analysis at the beginning of a data mining operation since they provide a method of analysis without the bias introduced by a training data set. The Self Organising Map (SOM) is the most widely used unsupervised neural network model. In the next section we present the SOM concepts and algorithm in detail.

2.3 Self Organising Maps (SOM)

One of the most significant attributes of a neural network is its ability to learn by interacting with a source of information. Learning in a neural network is accomplished through an adaptive process, known as a learning rule. With such learning rules, a set of *weights* of the network are incrementally adjusted so as to improve a pre-defined performance measure over time. The main categories of such learning algorithms are *supervised learning* and *unsupervised learning*. In supervised learning (also known as learning with a teacher), each input pattern received is associated with a specific target pattern. Normally, at each step of the learning process, the weights are updated such that the error between the network's output and the corresponding target is reduced. Unsupervised learning involves the clustering of (or similarity detection among) unlabeled patterns of a given data set. With this method, the weights of the output network are usually expected to converge to represent the statistical regularities of the input data [Has95]. The self organising map (SOM) is a neural network which uses unsupervised learning method and has become very popular in the recent past specially for data analysis and mining applications. Since the focus of this thesis is in the development of a novel neural network model based on the SOM, the concept of self organisation and the SOM algorithm is presented in the next section in detail.

2.3.1 Self Organisation and Competitive Learning

Self Organisation is a process of unsupervised learning whereby significant patterns or features in the input data are discovered. In the context of a neural network, self organisation learning consists of adaptively modifying the weights of a network of locally interacting units in accordance with a learning rule until a final useful configuration develops. By local interaction, it is meant that the changes in the behaviour of a node only affected its immediate neighbourhood. Therefore the concept of self organisation is related to the naturally ordered phenomenon whereby global order arises from local interactions. Such phenomena applies to both biological and artificial neural networks where many originally random local interactions between neighbouring units (nodes) of a network couple and coalesce into states of global order. This global order leads to coherent behaviour which is the essence of self organisation [Has95].

Unsupervised learning can be separated into two classes as *reinforcement learning* and *competitive learning* [Roj96]. In reinforcement learning each input produces a reinforcement of the network weights in such a way to enhance the reproduction of the desired output. Hebbian learning [Heb49] is an example of reinforcement rule that can be applied in this case. In competitive learning, the elements (nodes) of the network compete among each other for the *right* to provide the output associated with an input vector. The concept of self-organisation can be achieved

using competitive learning and the self organising map uses an extended version of competitive learning in its algorithm. Since our focus is on the self organising maps, we first describe the competitive learning rule before a detailed discussion of the SOM algorithm.

Competitive learning

Let us assume a simple neural architecture considering a group of interacting nodes. An example of such a network is shown in figure 2.2.

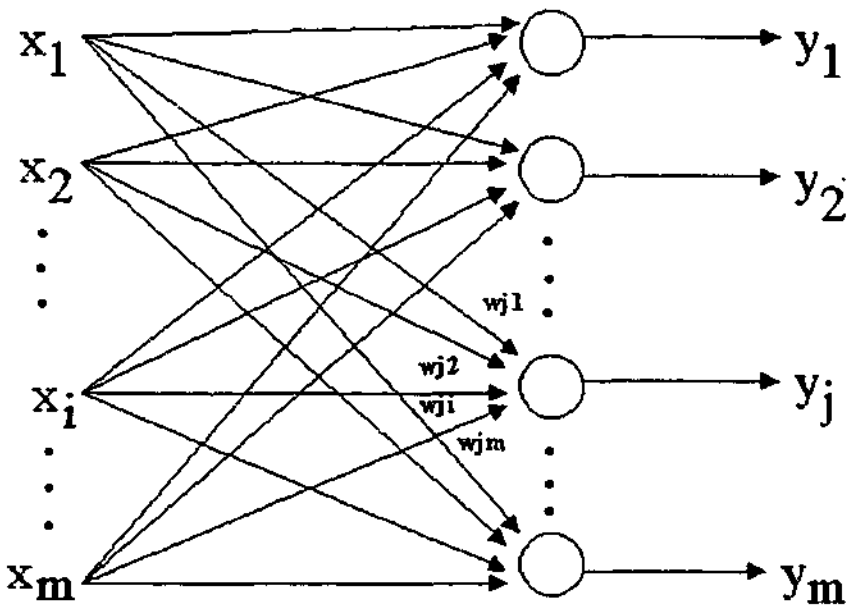


Figure 2.2: Competitive learning

Figure 2.2 has a single layer of nodes, each receiving the same input $x = [x_1, x_2, \ldots x_m]$ $x \in R^n$ and producing an output $Y_j, \quad j = 1 \ldots m$. It is also assumed that only one node can be active at any given input value. For a single input $x = [x_1, x_2, \ldots x_m]$,

the active node called the *winner* is determined as the node with the largest weighted sum of input x_k, net_i^k (for node i and $k = 1 \ldots m$), as

$$net_i^k = w_i^T x_k \tag{2.4}$$

where T denotes the transpose operator, $w_i = [w_{i,1}, \ldots, w_{i,D}]$ (D is the input dimension) and x_k is the current input. Thus node i is the winning node if

$$w_i^T x_k \leq w_j^T x_k \quad \forall j \neq i \tag{2.5}$$

which may be written as

$$|w_i - x_k| \leq |w_j - x_k| \quad \forall \ j \neq i \tag{2.6}$$

if $|w_i| = 1$, for all $i = 1, 2, \ldots, m$, then the winner is the node with the weight vector closest (in Euclidean distance), to the input vector. For a given input x_l (where $l = 1..N$, $N > 0$, are the set of inputs) drawn from a random distribution $p(x)$, the weight of the winning node is updated (the weights of all other units are unchanged) according to the following rule [Gro69], [Mal73].

$$\Delta w_i = \begin{cases} \rho(x_l - w_i) & ; \quad w_i \text{ is the weight vector of winning node} \\ 0 & ; \quad \text{otherwise} \end{cases} \tag{2.7}$$

where ρ in the learning rate. The preceding rule tilts the weight vector of the current winning node in the direction of the current input. The cumulative effect of the repetitive application of this rule can be described as follows [Has95].

> If we view the input and weight vectors as points scattered on the surface of a hypersphere, the effect of applying competitive learning

would be to sensitize certain nodes towards neighbouring clusters of input data. Ultimately, some nodes will evolve such that their weight vector points toward the *center of mass* of the nearest significant dense cluster of data points.

A modified version of the above simple competitive learning rule can be used to achieve self organisation as follows. Consider a network which attempts to map a set of input vectors x_l (where $l = 1..N$ are the set of inputs) in R^n onto an array of nodes (normally one or two dimensional) such that any topological relationships among the x_k patterns are preserved and are represented by the network in terms of a spatial distribution of the nodes. The more related two patterns are in the input space, the closer one can expect the position in the array of the two nodes representing these patterns. In other words, if x_1 and x_2 are similar or are topological neighbours in R^n, and if w_1 and w_2 are the weight vectors of the corresponding winner units in the net / array, then the Euclidean distance $|w_1 - w_2|$ is expected to be small. Also, $|w_1 - w_2|$ approaches to zero as x_1 approaches x_2. The idea is to develop a topographic map of the input vectors so that similar input vectors would trigger nearby nodes. Thus a global organisation of the nodes is expected to emerge. Such global ordering can be defined as self organisation and hence the self organisation can be realised by using a version of the competitive learning rule.

An example of such a topology-preserving self-organising mappings that exist in animals is the somatosensory map from the skin onto the somatosensory cortex [Koh95]. The retinotopic map from the retina to the visual cortex is another example [Koh95], [Rit92]. It is believed that such biological topology preserving maps are not entirely programmed by the genes and that some sort of unsupervised self-organising learning phenomenon exists that tunes such maps during development of a child [Pur94]. Initial work on artificial self organisation was motivated by such biological systems and two early models of topology-preserving competitive learning were proposed by von der Malsberg [Mal73] in 1973 and Willshaw and von der Malsberg [Wil76] for the retinotopic map problem. Although these models attempt to simulate the biological process as closely as possible, they were not feasible from a practical perspective, due to their complexity from attempting to too closely resemble the real brain. A simpler practical version of the self organisation concept was implemented by Kohonen [Koh82a], [Koh82b], called the self organising map (SOM). The SOM, due to its simplicity, ease of use and practicability has become the most widely used unsupervised neural network today. In the next section we will discuss the concept of SOM and its algorithm in detail.

2.3.2 The Self Organising Map Algorithm

In 1982 Teuvo Kohonen formalised the self-organising process defined by Malsberg and Willshaw into an algorithmic form that is now called the self organising

map (SOM) [Koh81], [Koh90], [Koh91], [Koh96a], [Rit91]. The development of the SOM was an attempt to implement a learning principle that would work reliably in practice, effectively creating *globally ordered maps* of various sensory input features onto a layered neural network. In the pure form, the SOM defines an *elastic net* of points (reference vectors), that are fitted to the input signal space to approximate its density function in an ordered way. The main applications of the SOM are thus in the visualisation of complex data in a two dimensional display, and the creation of abstractions as in many clustering techniques [Koh95].

The purpose of the SOM is to capture the topology and probability distribution of input data [Koh89]. This model generally involves an architecture consisting of two-dimensional structure (array) of nodes, where each node receives the same input $x_l \in R^n$ (Figure 2.3). Each node in the array is characterised by a n-dimensional weight vector, where n is equal to the dimension of the input data. The weight vector w_i of the i^{th} node is viewed as the position vector that defines the *virtual position* for node i in R^n.

The learning rule in the SOM is similar to that of competitive learning as in equation 2.7 and is defined as :

$$\Delta w = \rho \ \Omega(r_i, r_{win})(x^k - w_i) \quad \forall i = 1, 2, \ldots \tag{2.8}$$

where Δw is the weight change, ρ is the rate of weight change and r_{win} is the po-

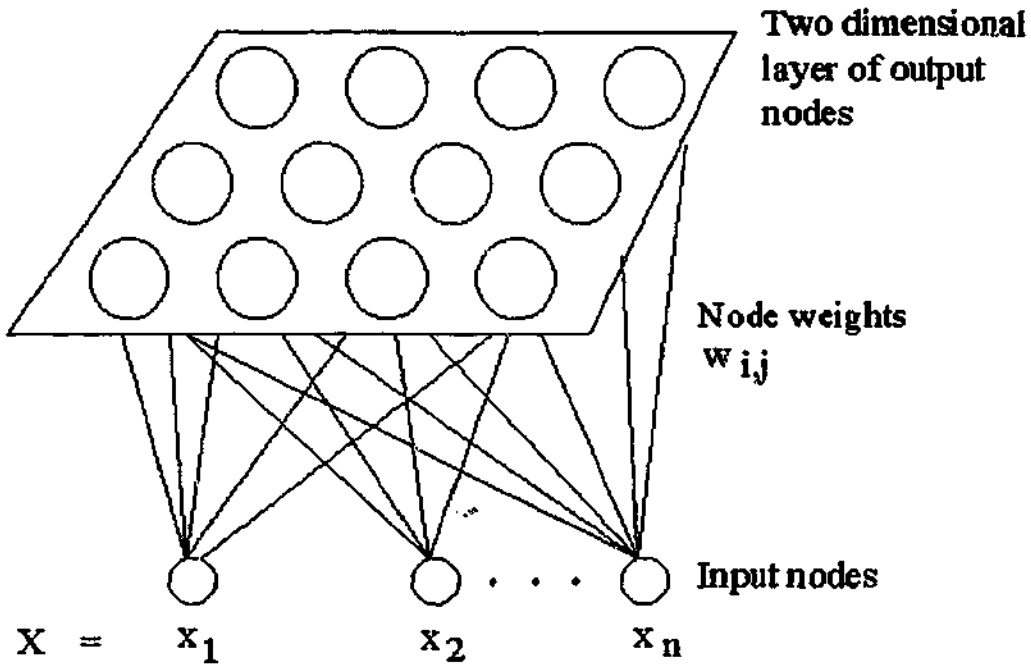


Figure 2.3: The Self Organising Map (SOM)

sition of the winning node. The winner is determined according to the Euclidean distance as in equation 2.6. The main difference between the SOM weight update rule and that of competitive learning is the neighbourhood function $\Omega(r_i, r_{win})$ in the SOM. This function is critical for the success in preserving of topological properties. It is normally symmetric ($\Omega(r_i, r_{win}) = (r_i - r_{win})$) with large values (values close to 1) for nodes i close to the winning node in the array, and monotonically decreases with the Euclidean distance $|r_i - r_{win}|$. At the beginning of learning, $\Omega(r_i, r_{win})$ defines a relatively large neighbourhood whereby almost all nodes in the net are updated for any input x^k. As learning progresses, the neighbourhood is shrunk down until it ultimately goes to zero, when only the winner is updated. The learning rate also must follow a monotonically decreasing sched-

ule in order to achieve this convergence. The initial large neighbourhood can be described as effecting an exploratory global search which is then continuously refined to a local search as the variance of $\Omega(r_i, r_{win})$ approaches zero. A possible choice for $\Omega(r_i, r_{win})$ is :

$$\Omega(r_i, r_{win}) = e^{-(|r_i - r_{win}|^2)/2\sigma^2} \tag{2.9}$$

where the variance σ^2 controls the width of the neighbourhood. Ritter and Schulten [Rit88] proposed the following update rules for ρ and σ.

$$\rho^k = \rho_0 \left(\frac{\rho_f}{\rho_0}\right)^{k/k_{max}} \tag{2.10}$$

$$\sigma^k = \sigma_0 \left(\frac{\sigma_f}{\sigma_0}\right)^{k/k_{max}} \tag{2.11}$$

where ρ_0, σ_0 and ρ_f, σ_f control the initial and final values of the learning rate and neighbourhood width respectively, k_{max} is the maximum number of learning steps anticipated. The computation by the repetitive use of the above process is captured by the following proposition due to Kohonen [Koh89].

> The w_i vectors tend to be ordered according to their mutual similarity, and the asymptotic local point density of the w_i, in an average sense, is of the form $g(p(x))$, where g is some continuous, monotonically increasing function.

Now we can present the self organising map algorithm in a summarised form as follows :

1. Initialisation: Define the required size of the network and start with the appropriate initial values for the weight vectors w_i, for each node. Random initialisation of weights is commonly used.

2. Input data: Choose (if possible) according to the probability density $P(x)$, a random vector x representing a *sensory signal* from the input data.

3. Response: Determine the corresponding *winning node* w_{win} based on the condition

$$|x - w_{win}| \leq |x - w_i|$$

for all $i \in A$ where A is the input space, x is the input vector, and w_i are the weight vectors of the nodes i in the network.

4. Adaptive step: Carry out a *learning step* by changing the weights according to

$$w_r^{new} = w_r^{old} + \rho\ \Omega(r, r')(x - w_r^{old})$$

where ρ is the learning rate adaptation and $\Omega(r, r')$ is the neighbourhood function.

5. Repeat the steps 2 thru 4 for each input until convergence (weight adaptation ≈ 0.

The mapping that is generated (ϕ_w) can be described as

$$\phi_w : V \mapsto A, \quad v \in V \mapsto \phi_w(v) \in A,$$

where $\phi_w(v)$ is defined through the condition

$$|w_{\phi_w(v)} - v| = min_{r \in A}|w_r - v|$$

This constitutes the neural map of the input signal space V onto the lattice A which is formed as a consequence of iterating steps 1 to 3 [Rit92]. Due to its ability to map the *features* (or relationships) in the input data in a topology preserving manner, such a map has been called a *feature map* of the input data.

2.4 Data Mining

In the recent past, the field of data mining and knowledge discovery has generated wide interest both in the academic community and industry. Data Mining has been defined as [Fay96b]

> *extraction of previously unknown, comprehensible, actionable, information from large data repositories.*

Although there has been many definitions and interpretations of the data mining process [Ber97b], [Hol94], [Big96], [Fay97], two main categories has appeared as, directed and undirected data mining or knowledge discovery. It can also be seen that there is a section of the data mining community who identify data mining solely with undirected data mining [Cab98], while categorising directed data mining with statistical hypothesis testing. The argument being that we *mine* for unknown, and unforeseen patterns and therefore if it is possible to build

a hypothesis of what we are looking for, it would not be data *mining*. The SOM is widely used, both in industry and research, as an undirected data mining technique for obtaining initial unbiased abstract view of *unknown* data sets.

2.4.1 The SOM as a Data Mining Tool

As described in section 2.3, the SOM is an unsupervised learning method which has been used for visualising clusters in data by generating a two dimensional map of a high dimensional data set. Clustering is a useful tool when the data mining analyst is faced with a large, complex data set with many variables and a high amount of internal structure. At the beginning of such a data mining operation, clustering will often be the best technique to use since it does not require any prior knowledge of the data. Once clustering has discovered regions of the data space that contain similar records, other data mining techniques and tools may be used to uncover patterns within the clusters. As such, due to its ability to generate dimensionality reducing clusters of a data set, the SOM is currently a popular tool with data mining analysts. The SOM is generally considered as an ideal tool for an initial exploratory data analysis phase in any data mining operation. As described in chapter 1, such exploratory analysis provides the data mining analyst with an unbiased overview of the data which can be made use of, to carry out better *targeted* secondary detailed analysis.

Although a number of attempts have been made at mathematically analysing and

interpreting the SOM process [Fla96], [Fla97], [Bou93], [Cot86], [Thi93], [Yan92], an accepted interpretation of the multi-dimensional case does not yet exist. But it has been theoretically proven that the SOM in it's original form does not provide complete topology preservation [Rit92], and several researchers in the past has attempted to overcome this limitation [Rit92], [Vil97]. The topology preservation achieved by the SOM, although not complete, has been widely made use of for knowledge discovery applications since the usage of SOM in data mining is mainly for obtaining an initial unbiased and abstract view of the data and not for highly accurate classifications. Therefore the advantages of the SOM for data mining applications can be listed as follows :

1. The SOM is an unsupervised technique and as such does not require previous domain knowledge, or target (expected) output to generate the clusters. Therefore it can be used to cluster an unknown data set. The analyst can then use these initial clusters to obtain an understanding of the data set and plan further experiments or queries. The SOM can also be used as a method for obtaining *unbiased* groupings from a data set about which some knowledge or *opinions* exist. Such unbiased clusters may accept or reject the existing knowledge. In either case it will contribute to the enhancement of the understanding of the data.

2. As discussed in section 2.3 that the SOM is generated using a competitive learning rule which was presented as equations 2.4, 2.5, 2.6. One of the

common applications of competitive learning is adaptive vector quantisation for data compression. In this approach, a given data set of x_k data points (vectors) is categorised into m templates so that later one may use an encoded version of the corresponding template of any input vector to represent the vector, as opposed to the vector itself. This leads to vector quantisation (compression) for storage and for transmission purposes (at the expense of some distortion). Such compressions achieved by the SOM is useful in initially observing a compressed abstract picture of a large data set, and to obtain an overall picture of the data. Data compression can also be made use of to initially study a small map and then select the areas of interest for further analysis with larger maps. This will save time and computer resources being spent on mapping *un-interesting* regions of the data.

3. As described above, the SOM also achieves dimensionality reduction by generating two dimensional map of a multi-dimensional data set. Such a map is advantages in visualising inter and intra cluster relationships in complex data with high dimensions.

Due to the above advantages and also the simplicity and ease of use, the SOM has become a very popular unsupervised neural network method for data mining [Wes98], [Big96].

2.4.2 Limitations of the SOM for Data Mining

As described in section 2.3, the SOM is normally represented as a two dimensional grid of nodes. When using the SOM, the size of the grid and the number of nodes has to be pre-determined. The need for pre-determining the structure of the network results in a significant limitation on the final mapping. It is therefore only at the completion of training steps that we realise that a different sized network would have been more appropriate for the data set. Therefore simulations has to be run several times on different sized networks to pick the optimum network [Ses94]. A further limitation when using SOM for knowledge discovery occurs due to the user not being aware of the structure present in the data. Therefore it not only becomes difficult to pre-determine the size of the network, but it is not possible to say when the map has organised into the proper cluster structure. In fact finding the proper structure is one of the goals in data mining.

It has also been shown that the SOM *forces* a shape on to the feature map by the shape and size of the grid. For example, when using a 2 dimensional grid of nodes, the length and the width of the grid will force the input data to be mapped to match the grid shape and size as shown in figure 2.4. Kohonen [Koh95] has called this problem *oblique orientation*. Since in data mining applications the analyst does not *know* the structure or shape of the data, it will not be possible to initialise a *proper* grid size to match the data, and the analyst would not

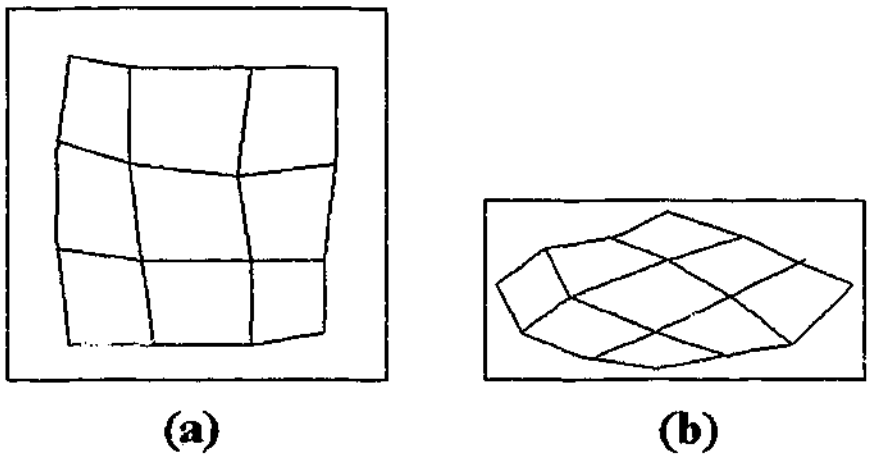


Figure 2.4: An example of oblique orientation (from [Koh95])

even be aware of the distortion to the map if and when such distortion occurs. Kohonen has suggested a method called the tensorial weights to improve the oblique orientation [Koh95], but this method only adjusts the map to fit into the pre-defined grid size. We suggest that the solution should be to adjust the grid size to match the map rather than the other way around. The distorted map gives the analysts an incorrect *picture* of the data. To summarise the problems with the SOM when used as data mining tool are :

1. the requirement of pre-defining the network structure, which is impossible with unknown data.

2. having to train several times with different grid sizes to obtain a *proper* map, even for the case of known data.

3. with unknown data, it will be impossible to identify such a proper map,

since the analyst is not aware of the relationships and clusters present in the data.

4. oblique orientation forcing the map to be distorted, thus providing an incorrect visualisation of the inter and intra cluster relationships.

One solution to the above mentioned problems would be to determine the shape and the size of the network during the training. Several researchers have attempted to develop such *dynamic* feature maps during the last few years. In the next section we will introduce the concept of structurally adapting neural networks and present the main work conducted towards developing such models.

2.5 Justification for the Structural Adaptation in Neural Networks

In the previous section, the limitations of the SOM as a tool for data mining applications is highlighted. It is also suggested that the structure of the SOM can be generated during training, and such a dynamic model will provide a more accurate representation of the data clusters. In this section we introduce the concept of structural adaptation in neural networks and review some of the important work in the development of such models.

As described in section 2.2 the ability to learn is the most important property

of an artificial neural network. In almost all of the neural network models the learning is done through modification of the synaptic weights of neurons in the network. Such change is generally done using the Hebbian learning rule or one of its adaptations. This kind of learning can be called a parameter adaptation process [Lee91]. The network designer must specify the framework or structure in which the parameters will reside. Specifically the network designer has to decide the number of nodes, and the relationships between the nodes in the network. When designing a feature map such as the SOM, the designer will also have to pre-define the *shape* of the network by specifying the length and the width. Selecting a *suitable* structure and relationships are important to the performance of the neural network and the accuracy of the outputs produced will depend upon such structure.

However choosing a suitable structure for the parameters is a difficult task, especially when the characteristics of the data are unknown. Even if a suitable structure is selected at the beginning, if the input data or the requirements of the application change, the system might not be able to produce useful results. The difficulties and limitations of neural networks due to conventional fixed pre-defined structures can be summarised as follows.

1. Neural networks in nature are not designed but evolves, through several generations, the system learns through interactions with the environment. It has been found that the structure of the neural system not only hap-

pens in the long term evolution process but also in the development process of an individual [Ede87]. Because the structure of the nervous system evolves rather than pre-specified, it would be very difficult to understand the function-structure relationship in the brain and try to map it into an artificial neural network design. Therefore, to build artifacts that display more sophisticated brain like behaviour, it would be necessary to allow the system to evolve its structure through the interaction with the environment. Since in artificial neural networks, the *environment* is represented by the input data, the structure of the neural network should be *decided* by the input.

2. In a fixed structure network, the system adaptability is limited which restricts the network capability. If the problem (specified by the input data) changes in size, the network might not be able to scale up to meet the requirements set by the new problem because the processing capability of the network is limited by the number of neurons in the system. Even if the problem does not change in *size*, it might change its context or the user requirements which might require a different organisation of neurons to be handled. There can be two possible solutions to this problem.

   (a) To design a network which is large enough and has sufficient connections to handle all possible requirements and change in scale.

(b) Allow the adapt its structure according to the characteristics of the as specified by the input data.

3. Artificial neural networks have limited resources compared to their natural counterparts. Therefore the network designer has to be careful when designing a network as to the usage of computing resources. Hence it will be advantages if a neural network can develop its structure as required by short term problem characteristics, rather than allocate all the resources during the life time to the network at once.

Structurally adapting neural networks are introduced as a solution to the problems listed above. Such neural networks should adapt

1. the number of neurons in the network defined by the process of node generation and removal,

2. the structural relationships between neurons by adding and removing connections.

Therefore structure adapting neural networks remove the restriction of pre-defined *frames* thus attaining the flexibility to better represent the data set. By incorporating unwanted node pruning, such networks can be further enhanced to *change* structure and connections to the changes in the input [Lee91].

2.6 Importance of Structure Adaptation for Data Mining

A neural network designer will consider the following when developing a new neural network.

1. the input data set,

2. the application and

3. the requirements of the users.

The difficulty faced by the network designer in this situation is mainly due to such dependence, since new networks will have to be built as the above specified requirements change. As described in section 2.4, in most data mining applications, the data analyst is not aware of the structure in the input data. In such situations, neural networks which are capable of self generating to match the input data have a higher chance of developing an optimal structure. We define an optimal network as a network which is neither too big nor too small for the specific data and application. When a network is too small there may be an information loss. For example, a feature map may not highlight the difference between some clusters. Alternatively, a network which is too large will result in an *over spread* effect where clusters may be spread out too far for proper visualisation. A larger map will also require more computing and system resources for

building the network.

The advantages of adaptable structure neural networks have caused some interest in the recent past. Several researchers have described such new neural architectures, both in the supervised and unsupervised paradigms. Although these new models obviously become more complex compared with their fixed structure counterparts, significant advantages could be gained from such dynamic models. The main advantage being their ability to grow (or change) the structure (number of nodes and the connections) to represent the application better. This becomes a very useful aspect in applications such as data mining, where it is not possible for the neural network designer to be aware of the inherent structure in the data.

2.7 Structure Adapting Neural Network Models

Several dynamic neural network models have been developed in the past which attempt to overcome the limitations of the fixed structure networks. Recent work on such supervised models have been reported in [Hal95],[Hal97], [Wan95], [Meh97]. There has also been several extensive reviews [Qui98], [Ash94], [Ash95b] on the supervised self generating neural architectures. Since the focus of this thesis is on self generating neural networks based on the SOM, we only consider

the previous work related towards the development of such models. A review of these models are presented below.

Growing Cell Structures (GCS)

The GCS algorithm [Fri91], [Fri94], [Fri92], [Fri96] has been based on the SOM, but the basic 2 dimensional grid of the SOM has been replaced by k-dimensional hypertetrahedrons being lines for $k = 1$, triangles for $k = 2$, tetrahedrons for $k = 3$. The vertices of the hypertetrahedrons are the nodes and the edges represent neighbourhood relations. Insertion and deletion of nodes is carried out during a self organising process similar to the SOM. For each new input v the best matching unit (bmu) is found using a similar criteria as the SOM. The weights of the bmu and its neighbours are adjusted as

$$w_{bmu}^{new} = w_{bmu}^{old} + \epsilon_{bmu}(v - w_{bmu}^{old}) \tag{2.12}$$

$$w_i^{new} = w_i^{old} + \epsilon_n(v - w_i^{old}) \quad (\forall i \in N_{bmu}) \tag{2.13}$$

where N_{bmu} is the neighbourhood of the bmu, and ϵ_{bmu} and ϵ_n are the learning rates for the bmu and its neighbouring nodes respectively. Unlike the SOM, the learning rates and the neighbourhood of winning node is kept constant. Therefore only the bmu and its direct neighbours weights are adapted.

Every node has a local *resource variable* which is incremented when it is selected as the bmu [Fri93]. After a fixed number of adaptation steps, the node q with the

highest resource value is selected as the point of insertion. The direct neighbour of q with the largest distance in input space (with q) is selected as

$$|w_f - w_q| \geq |w_i - w_q| \quad (\forall i \in N_q) \tag{2.14}$$

where w_q, w_f are the weight vectors of the highest resource node and its furthest immediate neighbour and N_q is the neighbourhood of the node with the highest resource value. A new node r is inserted between nodes q and f and the weight of the new node is initialised as

$$w_r = 0.5(w_q + w_f) \tag{2.15}$$

The resource variable of the new neuron is initialised by subtracting from its neighbours. Neurons which receive *very low* inputs (almost never become bmu) are deleted from the network. After both node addition and deletion, further edges are added or removed to rebuild the structure to maintain the k-dimensional hypertetrahedron.

With 2 dimensional input it can easily be verified that the network accurately represents the input space by plotting the weight vectors in 2 dimensions. However when the input is high dimensional, the structure may not be easily drawable. Therefore visualising the clusters becomes a complex task. Also, the arbitrary connectivity makes topological neighbourhoods ambiguous beyond directly connected nodes. Any node may be connected to any number of neighbours, extracting topological relationships of the input space from this structure may not be

easy. As such this method cannot be practically considered as a cluster identification method for data mining.

Incremental Grid Growing (IGG)

IGG [Bla95] builds the network incrementally by dynamically changing it's structure and connectivity according to the input data. IGG network starts with a small number of initial nodes, and generates nodes from the boundary of the network whenever a boundary node exceeds a pre-defined error value. The error value of a node is increased when the node is mapped by an input vector by the square of the difference between the nodes weight vector and the input vector. Adding nodes only at the boundary allows the IGG network to always maintain a 2 dimensional structure which results in easy visualisation. The new nodes generated are directly connected to the parent boundary node and their weights are initialised as

1. if any other directly neighbouring grid spots of the node with the highest *error* are occupied, then

$$w_{new,k} = 1/n \sum_{i \in N} w_{i,k} \tag{2.16}$$

where $w_{new,k}$ is the k^{th} component of the new nodes weight vector and n is the neighbouring nodes of the new node.

2. if no direct neighbours exist other than the parent node,

$$w_{par,k} = 1/(m+1)(w_{new,k} + \sum_{i \in M} w_{i,k}) \tag{2.17}$$

where $w_{par,k}$ is the k^{th} component of the parent node weight vector, and M is the set m of existing neighbours of the parent. Connections between nodes are added when an inter-node weight difference drops below a threshold value and connections are removed when weight differences increases above the threshold. Unlike the GCS method, the IGG has been successfully tested on some real data [Bla96].

One of the main limitations of the IGG method is that due to the node generation only from the boundary of the network, it does not provide proportionate representation of the input distribution by allocating nodes in the network. Such non-proportionate representation may provide a distorted visualisation of the clusters when the data is not uniformly distributed. The order of the data presented to the network also may result in such a distorted picture. Another limitation with the IGG is the time consuming calculations required for initialising the newly generated node weights a shown in equations 2.16 and 2.17. The new node weight initialisation may become unacceptably time consuming when used with real, complex data set [Bla95].

Neural Gas Algorithm

Martinetz et al. [Mar91] proposed an approach in which the synaptic weights w_i are adapted independently of any topological arrangement of the nodes within the neural network. Instead, the weight adaptation steps are affected by the topological arrangement of the receptive fields in the input space. Information

about the arrangement of the receptive fields within the input space is implicitly given by the set of distortions $D_v = |v - w_i|, i = 1 \ldots N$ associated with each input v. Each time an input v is presented, the ordering of the elements of the set D_v determines the adjustment of the synaptic weights w_i. The weight adaptation is carried out according to the formula

$$\Delta w_i = \epsilon f_i(D_v)(v - w_i) \tag{2.18}$$

where $f_i(D_v)$ is a function depending on the distortion D_v, which ensures higher amount of adaptation for nodes which are further away in the distortion list and vice versa. The formula is implemented as

$$w_i^{new} = w_i^{old} + \epsilon e^{-k_i/\lambda}(v - w_i^{old}) \tag{2.19}$$

where $\epsilon \in [0, 1]$ is the rate of weight adaptation, k_i is the number of nodes which have *distortions* greater than the specified node (i), and λ is the number of nodes with *significant* weight change.

Simultaneously with the weight adaptation, nodes i, j with receptive fields M_i, M_j adjacent on the manifold M develop connections between each other. The connections are described by setting the matrix element $C_{i,j}$ from zero to one. The resulting connectivity matrix $C_{i,j}$ at the end of the learning process represents the neighbourhood relationships among the input data. The Neural Gas also includes an aging mechanism for connections, where a connection $C_{i,j}$ is removed

when a pre-specified life time T is exceeded.

Neural Gas uses a fixed number of units which have to be decided prior to training. This again results in the same limitations as the SOM in data mining applications. The dimensionality of the Neural Gas depends on the respective locality of the input data. Therefore the network can develop different dimensionality for different sections, which can result in visualisation difficulties. Fritzke [Fri95a] has developed a Growing Neural Gas method where the initial number of nodes do not have to be specified. But the new method still has the limitation of variable dimensionality, and as such is not used with complex data.

Other Algorithms

The SPAN algorithm [Lee91] and the Morphogenetic algorithm [Joc90] and the Growing Grid [Fri95a], are other attempts to dynamically create SOMs thereby reducing the limitations of the fixed structure networks. Although these algorithms have been tested with low dimensional artificial data sets, no results have been published on realistic data. The SPAN algorithm includes a node generation method where new nodes can be inserted in the middle of the network. This method can become highly complex with real data sets due to the need for adjusting the network structure after each new insertion or deletion. The Morphogenetic algorithm generates a row or column of the network and as such has the potential of generating a large number of unnecessary nodes. The Growing

Grid (GG) is developed by Fritzke [Fri95b] with similarities to the GCS method but maintains a rectangular shape at all times by inserting and deleting whole rows and columns. The GG also may generate more nodes than required at a time and the rectangular grid removes the capability of the network to spread out according to the *shape* of the cluster structure.

Some related work has also been conducted on motion perception with SOMs [Mar90] [Mar95], [Sri97], temporal Kohonen maps [Cha93], and self organisation in the time domain [Tay95]. Although this work cannot be directly categorised as structurally adapting neural networks, they consist of modifications to the SOM such that change in data can be identified, and visualised. Such modifications provide the SOM with some dynamic ability and can become useful in data mining applications.

The different structure adapting neural network models considered in this chapter have been based on the SOM. Although they claim better and more accurate topology preservation, the simplicity and the ease of use of the traditional SOM has not been maintained. The GCS and the Neural Gas models can develop maps with different dimensionality, which can become difficult to interpret. The formulas used also result in high calculation overheads. Of all these new models, only the IGG has been tested on realistic, and complex data sets. The main reason being that the IGG always maintains a two dimensional output mapping

which can be easily visualised. As discussed above, even the IGG can result in non-proportional mappings, and also high computational complexity. Another limitation of the IGG is that due to only boundary nodes having the ability to initiate new node growth, the order of the input data presentation may have significant influence on the shape and the spread of the network. Our work in this thesis has been focussed on the development of a structurally adapting neural network, which improves on the usage of the SOM while maintaining its practicability.

2.8 Summary

The purpose of this chapter is to provide the background knowledge required for an understanding of structurally adapting self organising feature maps, which is the focus of this thesis. The competitive learning rule and the concept of self organisation is discussed in detail, and these concepts are used to describe the SOM algorithm. The reasons for the current popularity of the SOM as a data mining tool is presented and limitations in such usage are highlighted. Structural adaptation is introduced as a method of obtaining a more representative mapping of the data.

The second part of this chapter discussed the concept of structural adaptation in neural networks and highlights the advantages of such networks for data mining.

A review of past work where such models have been developed based on the SOM has been presented. The limitations of these models for realistic applications such as data mining have been highlighted. The major limitation of such models is either improper adaptation and/or computational complexity. We propose a novel neural network model called the Growing Self Organising Map (GSOM) which is also a structure adapting model based on the SOM, but overcomes the limitations of the previous models that have been developed so far.

The next chapter formalises our proposed model and the subsequent chapters discuss the optimisation of such a structurally adapting neural network and its use in data mining functions.

Chapter 3

The Growing Self Organising Map (GSOM)

3.1 Introduction

In the previous chapters data mining has been introduced as an useful activity mainly in commercial, but also in various other fields of science and research. The Self Organising Map (SOM) was presented as a useful unsupervised neural network method for data mining by identifying clusters in a data set. Due to its unsupervised learning ability, the SOM is used in scientific and commercial data mining, specially as a tool for obtaining an initial understanding of a data set about which little prior knowledge is available. Although widely being used, the SOM has shown significant limitations for knowledge discovery applications and these limitations are identified in chapter 2. The previous attempts in eliminat-

ing the limitations of the SOM with dynamic structure adapting neural networks such as the Growing Cell Structure (GCS), Incremental Grid Growing (IGG) and Neural Gas, are also discussed in chapter 2. The main focus of these dynamic models has been the improvement of the topology preserving ability of the SOM and thus obtaining an improved and more accurate mapping of the structure present in the input data. The cost of such improved accuracy has been a significant increase in complexity of the algorithms.

Knowledge discovery is an application where large and sometimes complex data sets have to be analysed. The usage of the SOM in such applications has been to discover the segments in the data set with a view to obtaining an initial understanding. Therefore the popularity of the SOM as a knowledge discovery tool has been due to its advantage as an unsupervised algorithm and also its simplicity and ease of use. As such, improved accuracy achieved with increased complexity of the algorithm will not be acceptable for data mining.

This chapter introduces the concept of the Growing Self Organising Map (GSOM) [Ala98e], [Ala98a], [Ala98d], [Ala98b], and discusses its properties. The limitations of the previous structure adapting feature maps are taken into consideration when developing the GSOM algorithm. The main purpose of developing the GSOM is to reduce the limitations faced by the SOM specially for knowledge discovery applications. The GSOM attempts to preserve the simplicity, ease of

use and efficiency of the SOM, but removes the need of pre-defining the structure and size of the network. Therefore apart from originating as a small network and *growing* neurons as required, the GSOM incorporates a modified learning rate adaptation method, a more localised neighbourhood weight adaptation, a method for initialising newly generated nodes and also a measure for controlling the spread of the network in terms of the spread factor.

As mentioned in chapter 1, the main focus of this thesis is the development of the GSOM as a more useful and efficient tool over the SOM for knowledge discovery applications. Therefore the GSOM concepts and algorithms presented in this chapter can be considered as the foundation for the rest of this thesis. In section 3.2 the concept of the GSOM and the corresponding algorithms are presented with implementation details given in section 3.3. Section 3.4 highlights the novel features of the GSOM which develops it into a more advantageous method compared to the SOM. Section 3.5 is a discussion of the advantages of the GSOM for knowledge discovery by using two benchmark data sets. The first benchmark is used to compare the GSOM with the SOM. The second experiment highlights the possibilities of unsupervised clustering with a traditionally *supervised data set.* Finally section 3.6 provides a summary of the contents of this chapter.

3.2 GSOM and the Associated Algorithms

3.2.1 The Concept of the GSOM

As described in chapter 2, the SOM is usually initialised to a two dimensional grid of nodes with weight values randomly selected from the input data range. The self organisation process discussed in chapter 2, orders and then adjusts the weights to represent the input data. The GSOM can be considered as a novel neural network model which is based on the same self organisation concept. In the GSOM the nodes are generated as the data is input and only if the nodes already present in the network are insufficient to represent the data. As such, the weights as well as the network size and shape can be said to self organise in the GSOM. Therefore the GSOM finally arrives at a map (network) which is a better representation of the input data as well as having fewer redundant nodes compared to the SOM. Hence GSOM has the flexibility to spread out and thus arrive at a more representative shape and size for a given data set. In other words instead of initially starting with the complete network as in the SOM, the GSOM starts with a small network and generates new nodes where required, as identified by a heuristic. Similar to the SOM and many other neural network models, the GSOM has two modes of activation, namely the training mode and the testing mode. The actual network construction and the weight adaptation occurs during the training mode, while the testing mode is used to calibrate the

trained network with *known* inputs, when such inputs exist. The training mode consists of the following three phases.

1. Initialisation phase.

2. Growing phase.

3. Smoothing phase.

These phases are described in section 3.3. With the GSOM, the network designer does not have to figure out an optimum network structure at the beginning of training phase since all GSOMs are initialised with four nodes. By defining a parameter called the spread factor (SF) at the beginning of network construction, the user (data analyst) has the ability to control the spread of the GSOM. The spread factor is used to calculate a growth threshold (GT) which is then used as a threshold for initiating new node generation.

Figure 3.1 shows a few steps of how GSOM gradually forms the network as input is presented. In figure 3.1(i), a GSOM starts initially with four nodes, the weight values of which are randomly initialised. This initial structure is selected since it is the most appropriate as a starting point for implementing a two dimensional rectangular lattice structure.

Once the network is initialised, input is presented to the network. For each input the node with the weight vector closest to the input (measured as Euclidean dis-

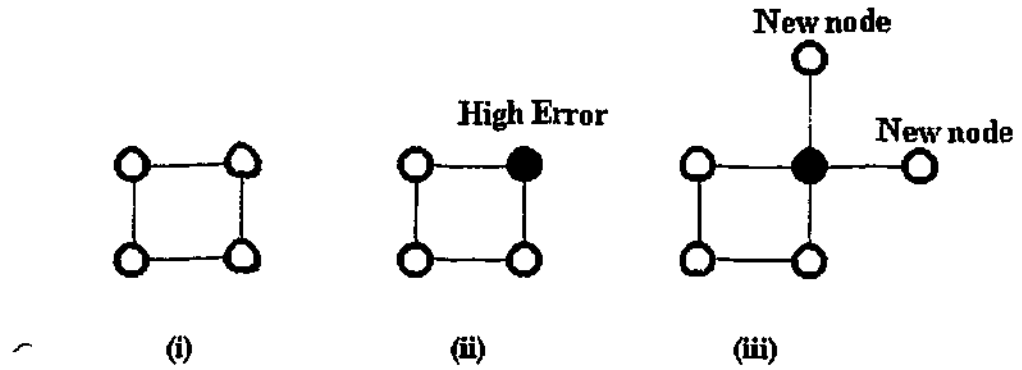


Figure 3.1: New node generation in the GSOM

tance) is judged as the winner and neighbourhood weights are *nudged* (adjusted) closer to the input value by a factor called the learning rate adaptation. This process is similar to the SOM, but as justified in section 3.4, localised neighbourhood adaptation is sufficient in the GSOM since there is no ordering phase as in the SOM. Each time a node is selected as the winner, the difference between the input vector and the weight vector is calculated and accumulated in the respective node as an *error* value. The network keeps track of the highest such error value and periodically compares this value with the growth threshold (GT). When the error value of a node exceeds the GT, a new node generation is initiated (as indicated in Figure 3.1(ii)). New nodes are generated from all *free* positions of the selected node (Figure 3.1(iii)). This process continues until all inputs have been presented. If the number of input is small, the same input set is repeatedly presented several times until the frequency of new node generation drops below a specified threshold.

After the above described node generation phase, the same input data is presented to the fully developed network. On this occasion the weight adjustment of winner and its neighbours continue without new node generation. At the beginning of this phase the initial learning rate adjustment value is reduced from the value used in the node generation phase, as well as the neighbourhood for weight adjustment is restricted to the winner's immediate neighbours. The purpose of this phase is to smooth out any nodes which has not yet settled into a match with their respective neighbourhoods, and can be compared to the *convergence phase* in the SOM [Koh95]. This process is continued until convergence (error ≈ 0) is achieved. In the next sub-section, the three phases of the training mode are presented in detail as the GSOM algorithm.

3.2.2 The GSOM Algorithm

The GSOM training algorithm is presented below according to the three phases of initialisation, growing and smoothing. The process is started by generating the initial network of four nodes and the user providing the parameter values for the initial learning rate adaption, spread factor and the number of instances (records) in the input data set. Therefore the training algorithm can be presented as follows.

1. Initialisation Phase

(a) Initialise the weight vectors of the starting nodes in the initial map with random numbers between $0 \ldots 1$.

(b) Calculate the growth threshold (GT) for a given data set according to the spread factor (SF) using the formula

$$GT = -D \times \ln(SF)$$

where D is the dimensionality of the data set. The growth threshold is used to identify nodes from which new nodes need to be *grown*. The criteria for such node growth is presented in the next section. The formula for GT is explained further in section 3.4.

2. Growing Phase

(a) Present input to the network.

(b) Determine the node with the weight vector that is closest to the input vector, using Euclidean distance measure (similar to the SOM). In other words find a node q' in the current network such that $|v - w_{q'}| \leq |v - w_q| \; \forall q = 1 \ldots \mathcal{N}$ where v, w_q are the input vector and weight vector of the node q respectively, q is the position vector for nodes in the map and $\mathcal{N}$ is the number of existing nodes in the map.

(c) The weight vector adaptation is applied only to the neighbourhood of the winner and the winner itself. The neighbourhood is defined as a set of nodes which are topographically close in the network up to a

certain geometric distance. For example, 4 neighbours are defined in the 4 directions. In the GSOM the starting neighbourhood selected for weight adaptation is smaller compared to the SOM (localised weight adaptation). The amount of adaptation also known as learning rate is reduced monotonically over the iterations such that the weight values will converge to the input data distribution. (Such learning rate reduction is required for convergence in self organising system). Even within the neighbourhood, amount of weight adaptation of a node is in proportion to the respective node distance from the winning node, and this will eventually result in similar input values getting clustered (or assigned to nodes which are closer in the network) in the map. More formerly the weight adaptation can be described as

$$w_j(k+1) = \begin{cases} w_j(k), j \notin N_{k+1} \\ \\ w_j(k) + LR(k) \times (x_k - w_j(k)) \in N_{k+1} \end{cases}$$

where the learning rate $LR(k), k \in \mathcal{N}$ is a sequence of positive parameters converging to 0 as $k \to \infty$. $w_j(k)$, $w_j(k+1)$ are the weight vectors of the the node j, before and after the $(k+1)^{th}$ iteration, and N_{k+1} is the neighbourhood of the winning neuron at $(k+1)^{th}$ iteration. The decreasing of $LR(k)$ in the GSOM, depends on the number of nodes

that exist in the network at a given time k (which will be described in section 3.4).

(d) Adjust the error value of the winner (error value is the difference between the input vector and the weight vector) as :

$$E_i^{new} = E_i^{old} + \sqrt{\sum_{i=1}^{D} (v_i - w_i)^2}$$

where E_i is the error of node i, D is the dimension of the data and x_i and w_i are the input and weight vectors of node i respectively.

(e) When $E_i \geq GT$ (where E_i is the total error of node i and GT is the growth threshold), grow nodes if i is a boundary node, else distribute the error of the winner to its neighbours if it is a non-boundary node. The method of error distribution is described in section 3.4.5.

(f) Initialise the new nodes weight vectors to match the neighbouring node weights (described in 3.3.2).

(g) Initialise the learning rate (LR) to it's starting value, which is the value provided as one of the parameters by the user at the beginning of the initialisation phase.

(h) Repeat steps a to g until all inputs have been presented, and the frequency of node growth is reduced to a low level, which can be decided by the user by providing a threshold value (such as, stop the growing phase when a new node added after only after 100 iterations).

3. Smoothing Phase

   (a) Calculate new parameter values for learning rate and the starting neighbourhood by reducing the learning rate and the starting neighbourhood from respective values used in the growing phase. For example, in the experiments with the GSOM the initial learning rate was reduced by half in the smoothing phase and the starting neighbourhood was fixed at only the immediate four neighbouring nodes.

   (b) Present input (same inputs as in growing phase) to the network.

   (c) Find winner and adapt weights of winner and neighbours with the changed parameter values.

   (d) Repeat steps (b) and (c) until the error value (between input and weight of the winning node) approaches zero. When the error ≈ 0 the weights of the nodes are *converged* and the GSOM creation process is considered as complete.

From the above algorithm we can say that the SOM attempts self organise by weight adaptation while the GSOM adapts its weights and architecture to represent the input data.

3.3 Description of the Phases in the GSOM

In this section we provide detailed description of each of the phases in the GSOM algorithm. Although based on the SOM, the GSOM includes a number of significant variations which are required to implement its adaptable structure. Section 3.3.1 describe the starting parameters required by the GSOM. Section 3.3.2 describes the growing phase. The criteria for deciding new node growth and the method of initialising the weights of such new nodes are described in this section. Section 3.3.3 discuss the need and the function of the smoothing phase.

3.3.1 Initialisation of the GSOM

We initialise the network with four nodes in a two dimensional form. Such a structure is selected because

- it is the most appropriate starting position to implement a two dimensional lattice structure, since it is the smallest possible grid (we do not consider initialising with one node due to implementation difficulty).

- with this initial structure, all starting nodes become boundary nodes (nodes at the boundary of the map) which can initiate new node growth if required, thus each node has the same freedom to grow in it's own direction at the beginning (Figure 3.2).

The starting four nodes are initialised with random values from the input vector value range. Since we normalise the input vector attributes to the range 0..1, the

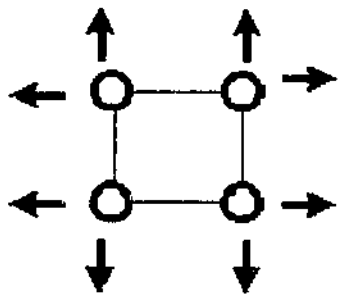


Figure 3.2: Initial GSOM

initial weight vector attributes will take random values in this range. Therefore at the beginning, we do not enforce any restrictions on the directions of the lattice growth. Thus the initial rectangular shape with the randomly initialised weights, lets the map grow in any direction solely depending on the input values. A numeric variable *HiErr* is initialised to zero at the start. This variable will keep track of the highest accumulated *error* value in the network. A value called the spread factor (SF) has also to be specified. The SF allows the user (data analyst) to control the growth of the GSOM, and is independent of the dimensionality of the data set used. SF can take values in the range 0..1, and the data analyst needs to decide an appropriate value at the beginning of the algorithm. A *low* SF will result in less spread out in the map and a *high* SF will produce a well spread map. SF is used by the system to calculate the growth threshold (GT). This will act as a threshold value for initiating new node generation. The justification for the SF and the derivation of the formula for GT is described in section 3.4.

3.3.2 Growing Phase

The second phase of building the GSOM is the growing phase. In this phase we will describe the criteria used for generating new nodes and also the novel method used in initialising the weights of the new nodes. The input vectors will be used to grow the network as well as adapt its structure. A number of parameters that are used in this phase will determine the final shape of the network. Both the method of growing the network and the effect of the parameters will be described in this section.

New node generation

In the growing phase, the GSOM needs to generate nodes to represent the input data space. Therefore it is necessary to identify a criterion to initiate new node generation. To determine the criterion, we consider the following behaviour during training of a feature map.

- If the neural network has enough neurons to process the input data then during training the weight vectors of the neurons are adapted such that the distribution of the weight vectors will represent the input vector distribution.

- If the network has insufficient neurons, a number of input vectors, which otherwise would have been spread out to neighbouring neurons, will be accumulated on a single or a small set of neighbouring neurons.

Therefore we introduce a measure called the error distance (E) as

$$E_i^{new}(t) = E_i^{old}(t) + \sum_{i=1}^{D} Met(v_k, w_{j,k})^2 \tag{3.1}$$

The error measure is calculated for neuron i at time (or iteration number) t, D is the dimension (number of attributes) in the input data, and v, w are input and weight vectors respectively. Met is a metric which measures the distance between the vectors v and w. Thus for each winner node, the difference between the weight vector and the input vector is calculated as an *error* value. This value is accumulated over the iterations if the same node wins on several occasions. Using Euclidean distance as the metric we can re-write equation 3.1 as

$$E_i^{new} = E_i^{old} + \sqrt{\sum_{i=1}^{Dim} (v_i - w_i)^2} \tag{3.2}$$

A variable $HiErr$ is used such that at each weight updating, if $E_i^{new} > HiErr$ then $HiErr = E_i^{new}$ else $HiErr$ is unchanged. Thus $HiErr$ will always maintain the largest error value of the neurons in the network. The error value calculated for each node can be considered as a quantisation error for that node and the total quantisation error would be

$$QE = \sum_{i=1}^{N} E_i \tag{3.3}$$

where N is the number of neurons in the network and E_i is the error value for neuron i. We use QE as a measure of determining when to generate a new neuron (or node). If a neuron i contributes substantially towards the total quantisation error, then its Voronoi region V_i in the input space is said to be under-represented

by neuron i. Therefore a new neuron is created to share the load of neuron i. To determine the neuron which is being over burdened with the inputs we use the partial differentiation of the total quantisation error by Voronoi region, there by identifying the quantisation error for the regions. Therefore, the criteria for new node generation becomes

$$\frac{\partial QE}{\partial E_i} E_i > GT \tag{3.4}$$

where GT is the growth threshold calculated using the spread factor as described in section 3.2. Since in our implementation $HiErr$ contains the largest error value for a neuron, we can re-write this equation 3.1 as

$$\text{grow new nodes if } HiErr > GP$$

New nodes will always be grown only from a boundary node. A boundary node is a node which has at least one of its immediate neighbouring positions free. Since we assume a 2 dimensional network and the initial network has 4 nodes organised on a square, each node will have 4 immediate neighbouring positions. Thus a boundary node can have from 1 to 3 neighbouring positions free (2 free positions each for the initial 4 nodes). If a node is selected for growth because of $HiErr > GP$ as explained above, then new nodes are created in all it's free neighbouring positions. We generate new nodes on all free neighbouring positions since it is easier to implement than calculating the exact position for the new node. This will create some redundant (dummy) nodes, but we can easily identify and

remove the dummy nodes after a few iterations as they will accumulate less (almost zero) hits (or less error value).

Weight initialisation of new nodes

The newly grown nodes need to be assigned some initial weight values. Since the older nodes would be at least partly organised at this stage (self organisation of the existing nodes happens from the start), random initialisation of the new nodes will introduce weight vectors which do not match their neighbourhoods. Therefore, we need to take into consideration the smoothness already achieved by the existing map and thus initialise the new weights to match their neighbourhoods. For the weight initialisation of new nodes there are four different situations that need be considered (Other situations are mirror images of these situations) as shown in Figure 3.3. In Figure 3.3 new node is indicated by a circle while the existing neighbouring nodes are indicated by black circles.

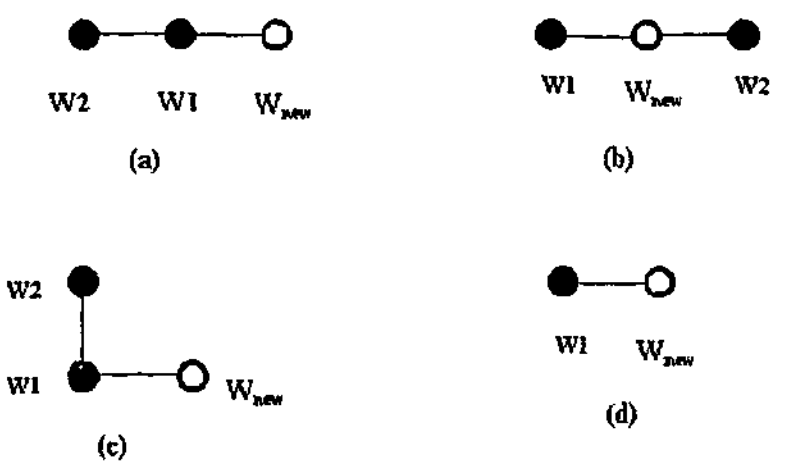


Figure 3.3: Weight initialisation of new nodes

First consider the case where the new node has two consecutive old nodes, both on one side (Figure 3.3(a)).

if $w2 > w1$

then $w_{new} = w1 - (w2 \tilde{-} w1)$

if $w1 > w2$

then $w_{new} = w1 + (w1 - w2)$

The rationale behind this new weight calculation is to initialise the new node to *fit in* with the existing weight values in a monotonically increasing or decreasing manner. This will ensure that the weights of the map are *ordered* even at initialisation. Such order will reduce the amount of *weight smoothing* required and will result in less chances of twisted (deformed) maps [Koh95].

In the second case (Figure 3.3(b)) when the new node is in between two older nodes (with weights w_1 and w_2) the weight of the new node will be calculated as

$w_{new} = (w_1 + w_2)/2$

The same rationale as in the first case of *smooth merging* of weights is also used in this case.

The third case is when the new node has only one direct neighbour (the parent) older node (Figure 3.3(c)). But the older node has a neighbour on one side which

is not the side directly opposite the new node itself. In this case

if $w2 > w1$

then $w_{new} = w1 - (w2 - w1)$

if $w1 > w2$

then $w_{new} = w1 + (w1 - w2)$

In this case, since there is no node directly *in-line* to measure the monotonically increasing/decreasing nature, we take the closest neighbour *on that side* as an alternative.

The fourth situation is where the new node has only one neighbouring node. This can occur when a node which has become isolated due to node removal, initiates growth again. Therefore this is a situation which will occur rarely due to an unusual sequence of input presentation to the network (Figure 3.3(d)). In this case, the weight value of the new node will be

$w_{new} = m$ where $m = (r1 + r2)/2$

and $r1, r2$ being the lower and upper values of the range of the weight vector distribution (in most applications, since the input values are scaled to the range [0..1], $r_1 = 0, r_2 = 1$). The rationale for calculating this value is that, since

sufficient neighbourhood nodes do not exist to calculate a suitable new weight for this node, we initialise it with the middle of the weight distribution range. The self organisation process then adjusts the weight to fit in with the global spread of weights. This method is also used to provide initial values to the nodes when the values calculated by the other methods (shown in Figure 3.3(a), (b) and (c)) are out of the weight range (for example < 0 or > 1).

The above weight initialisation method has been developed taking the properties of the *organised* SOM into consideration. Once the weights converge the map will achieve a state where the weight vectors of the nodes in the map will take monotonically increasing or decreasing values from end to end. This can be described as the *flow* of the weight vectors in the converged map.

3.3.3 Smoothing Phase

The smoothing phase occurs after the growing phase. The growing phase is halted when new node growth saturates, which can be identified by the low frequency of new node growth. Once the node growing phase is complete, the weight adaptation is continued with a lower rate of adaptation in the smoothing phase. No new nodes are added during this phase. The purpose is to smooth out any existing quantisation error, specially in the nodes grown at the latter stages of the growing phase.

During the smoothing phase the same inputs as the growing phase are input to the network. The starting learning rate (LR) in this phase is less than in the growing phase, since the weight values should not fluctuate too widely since this will hinder convergence. The input data is repeatedly entered to the network until convergence is achieved. The smoothing phase is terminated when the error values of the nodes in the map become very small.

Therefore the smoothing phase has the following differences from the growing phase.

1. The learning rate (LR) is initialised to a lesser value.
2. The neighbourhood for weight adaptation is constrained only to the immediate neighbourhood (even smaller than in the growing phase).
3. The learning rate depreciation (rate of decrease) is smaller.
4. No new nodes are added to the network.

3.4 Parameterisation of the GSOM

In the previous section, the concept of the GSOM and its algorithm is presented in detail. The GSOM algorithm is based on the SOM but has been extensively modified to cater for the needs of structure adaptation by node growing. The novel aspects of the GSOM algorithm are :

- The new learning rate adaptation method
- Localised neighbourhood weight adaptation
- Criteria for new node generation
- New weight initialisation method for the new nodes
- Error distribution when a non-boundary node satisfies the criteria for node generation
- The concept of the spread factor to control the growth (size) of the network.

In this section we analyse and justify the introduction of these new methods and discuss the implications and methodology for choosing the values for the parameters.

3.4.1 Learning Rate Adaptation

The use of learning rate adaptation of the GSOM is described in section 3.2. Since the learning rate has to be provided as a parameter at the start of building the network from the given data, we need to consider it as part of the initialisation of the GSOM. As described in chapter 2, the learning rate should gradually decrease with iterations, finally approaching to zero when the node weights converge. One possible formula for learning rate reduction used in the SOM is given in equation 3.5.

$$LR(k+1) = LR(k) \times \alpha \tag{3.5}$$

where α is the learning rate reduction, and is implemented as a constant value $0 < \alpha < 1$, and $LR(k)$ is the learning rate at the k^{th} iteration. The use of α in the equation 3.5 makes $LR(k)$ to converge to 0 as $t \to \infty$. In the GSOM, LR is first initialised with a *high* value, similar to that of SOM. However since the GSOM has only a small number of nodes in the initial stages, completely *different* inputs can be presented to the map consecutively and can be mapped to neighbouring nodes in the network. This can occur when the input vectors are presented to the network in random order. This situation can be described with figure 3.4 where the initial GSOM is shown with the initial randomly initialised weight values. The weight values for the nodes P, Q, R, S in figure 3.4 are assumed to be,

$W_P = (1, 0.85, 0.7), W_Q = (0.4, 0.3, 0.2), W_R = (0.05, 0.[illegible]), W_S = (0.7, 0.8, 0.3)$.

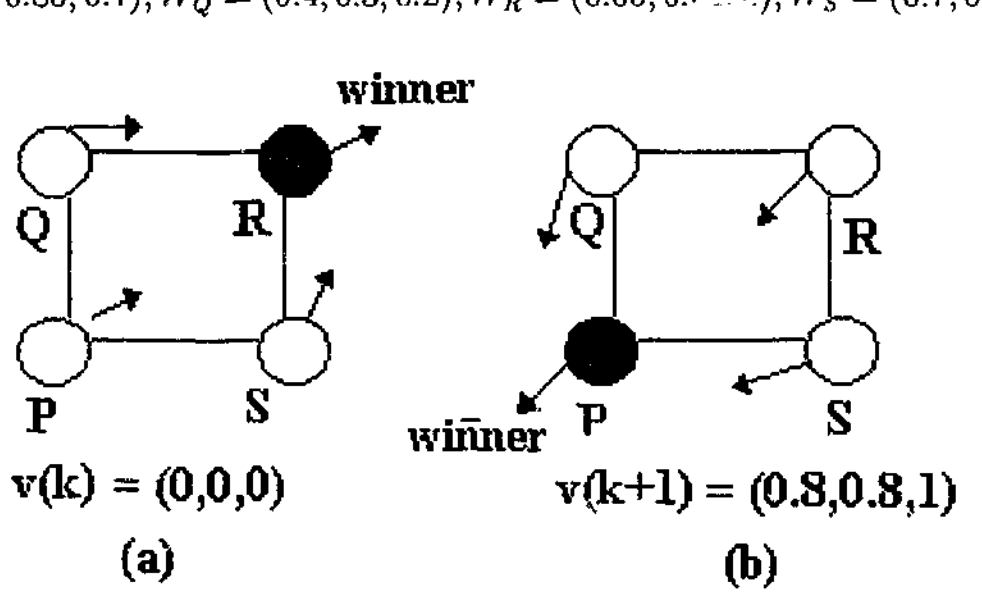


Figure 3.4: Weight fluctuation at the beginning due to high learning rates

Assume an input $v(k) = (0, 0, 0)$, then according to the rule described in section 3.2 it will be mapped to the node with weight W_R because the error value will

be the smallest. Since there are only four nodes in the network, the whole network will become the neighbourhood and the weights of the neighbours would be adjusted towards W_K. Since the inputs are randomly presented, we assume a situation where the next input to be $v(k+1) = (.8, .8, .9)$, and this input vector will get mapped to node P which will cause all the node weights to be adjusted, now in the *opposite* direction. As such due to the small network size, weight values of the whole network will keep fluctuating in different *directions* until the network is sufficiently *grown* such that the whole network will not be included as the neighborhood. Therefore a modification of the learning rate formula is required to rectify this problem, which can otherwise result in distorted maps.

The GSOM uses a similar method as SOM for initialising the LR to a *high* value and letting it decrease to zero after a number of iterations. However, in the GSOM, during the growth phase, the LR is initialised to its starting value with each new input. Such renewing of LR is required in the GSOM to accommodate newly grown nodes into the existing network. Therefore with a small initial network a high LR value is repeatedly used on all the nodes, each time a new input is presented, and if consecutive inputs have large variation, the weights of all the nodes can fluctuate by a large value in different directions. This has proved to be detrimental to the generation of a well spread map, since the initial small network should ideally provide the directions of expansion for the network according to different clusters of similar weights. Therefore the initial *indecision*

by the small network (due to the wide fluctuation) will result in a map which does not sufficiently separate the different regions (clusters) in the input.

One way of improving this situation is to order the data according to attribute values, since the ordered data vectors will make the map grow in a certain direction. Therefore by the time a *different* type of input is presented, the map will be sufficiently grown, such that all the nodes will not fluctuate towards the new input. Although this method is practical with a *known*, small data set, it would be impossible to order an unknown data set since the data analyst is not aware of the relationships between the attributes.

Therefore as a solution, we have introduced a new learning rate reduction rule which takes the number of current nodes in the neural network into consideration. The new learning rate reduction rule is

$$LR(k+1) = \alpha \times \psi(n) \times LR(k) \tag{3.6}$$

where $\psi(n)$ is a function of $n(t)$, the number of nodes at iteration t in the map, and is used to manipulate the LR value. That is, $\psi(n)$ is a function which gradually takes a higher value as the map grows and the number of nodes becomes larger. One simple formulation for $\psi(n)$ is

$$\psi(n) = (1 - R/n(t)) \tag{3.7}$$

and in our experiments we have chosen $R = 3.5$.

$$\psi(n) = (1 - 3.5/n(t))$$

The new formula will force the following functionality on the weight adjustment algorithm.

1. At the initial stage of growth, when the number of nodes are few, the high $\breve{LR}$ values will be reduced with the $\psi(n)$. This will result in reducing the fluctuations of the weights of a small map.

2. As the network grows, the $\psi(n)$ will take gradually higher values which will result in the LR not being reduced as much as in the situation described in (1). Since the map is now larger, we require the high LR to *self organise* the weights within the regions of the map (the high LR will not affect the other regions at this stage due to the map being larger and the other regions will not belong to the same neighbourhood).

Therefore the modified formula provides the GSOM with the required weight adjustment, which results in providing a better organised set of weight values which will then be further smoothed out during the smoothing phase.

3.4.2 Criteria for New Node Generation

The weight vectors of the nodes (neurons) in the GSOM can be considered as a vector quantisation of the input data space. Therefore each neuron i can be said

to represent a region V_i where all the points in the region are closer to w_i (the weight vector of neuron i) than any other neuron in the network. Therefore we can say that the weight vectors partition the input space into *Voronoi regions* [Oka92], with each region represented by one neuron.

In section 3.2 we described the difference between the input vectors and the corresponding weight vectors which become the accumulated error values of each node. Therefore the quantisation error of the system (at iteration t) can be shown as the expectancy value of the variance between the input vectors x and the representative vectors τ by

$$QE(t) = E[|x - \tau(x)|^2] \tag{3.8}$$

Using the weight vector w as the representative vector, equation 3.8 can be shown as an integration

$$QE(t) = \int_V |x - w|^2 p(x) d(x) \tag{3.9}$$

where $V = \bigcup_{i \in S} V_i$ is the input data space with S input records and $p(x)$ is the probability distribution of the input vectors v. If a neuron i contributes *significantly* to the total error (distortion) of the network, then it's Voronoi region V_i in the input data space is thought to be under represented by the neuron i. Therefore a new neuron is generated as a neighbour of the neuron i to achieve a better representation of the region. Such new neuron generation is expected to distribute the error in the region among the old and new neurons thus reducing

the individual node error. As explained in section 3.2, the error value of a node can exceed the GT (growth threshold) value for growth to occur. Whenever an input vector falls (assigned) into the Voronoi region V_i, the input is mapped to neuron i. Therefore we can write the *distortion* (or quantisation error) for the network by integrating the individual error values over the total input regions (all Voronoi regions) as

$$QE(t) = \sum_{i=1}^{M} \int_{V} |x(t) - w_i(t)|^2 p(x) d(x) \tag{3.10}$$

where x, w are the input and weight vectors respectively, $p(x)$ is the probability of the input x being assigned to Voronoi region V and t is the iteration number. The probability of an input being assigned to neuron i (the Voronoi region V_i) will be

$$P_i = \int_{V_i} p(x) d(x) \tag{3.11}$$

Equation 3.10 can be now replaced by

$$QE(t) = \sum_{i=1}^{M} (\int_{V_i} |x(t) - w_i(t)|^2 \ \frac{p(x)}{P_i} d(x)) P_i \tag{3.12}$$

By substituting from equation 3.8 we get

$$QE(t) = \sum_{i=1}^{M} E[|x(t) - w(t)|^2 \ x \ \in \ V_i] \ P_i \tag{3.13}$$

$$QE(t) = \sum_{i=1}^{M} e_i P_i \tag{3.14}$$

where e_i is the average error for neuron i.

The equation 3.14 shows that any neuron i contributes $e_i P_i$ to the total quantisation error. Therefore we can see that the total error is made up of the difference between input and weight vectors (e_i) and also the probability of the input being mapped to a neuron (P_i). The probability of input to a region can be interpreted as the density of input distribution for that region. Therefore we can see that the new node generation can occur in two situations.

1. A very high number of *hits* in the Voronoi region of the node itself creating a high density region.

2. A boundary node (nodes at the edge of the network) being assigned inputs which would have been mapped to another node, if such nodes existed in the network.

We use Figure 3.5 to demonstrate the two different instances of new node generation using Voronoi regions. Figure. 3.5 shows four nodes with weight vectors W_1 to W_4 and their respective Voronoi regions in two dimension. v_1 to v_{n+1} are the input vectors which have been assigned to the regions. R_1, R_2 and R_3 are 3 Voronoi regions. We will consider the two situations (1) and (2) mentioned above with this diagram.

In Figure 3.5(a), a large number (n) of inputs have been assigned to the node with weight value W_2 (region R_1, which is a boundary region). In this case the error value $E = \sum_{i=1}^{n} (v_i - w_i)^2$, can exceed the GT (Growth Threshold) value

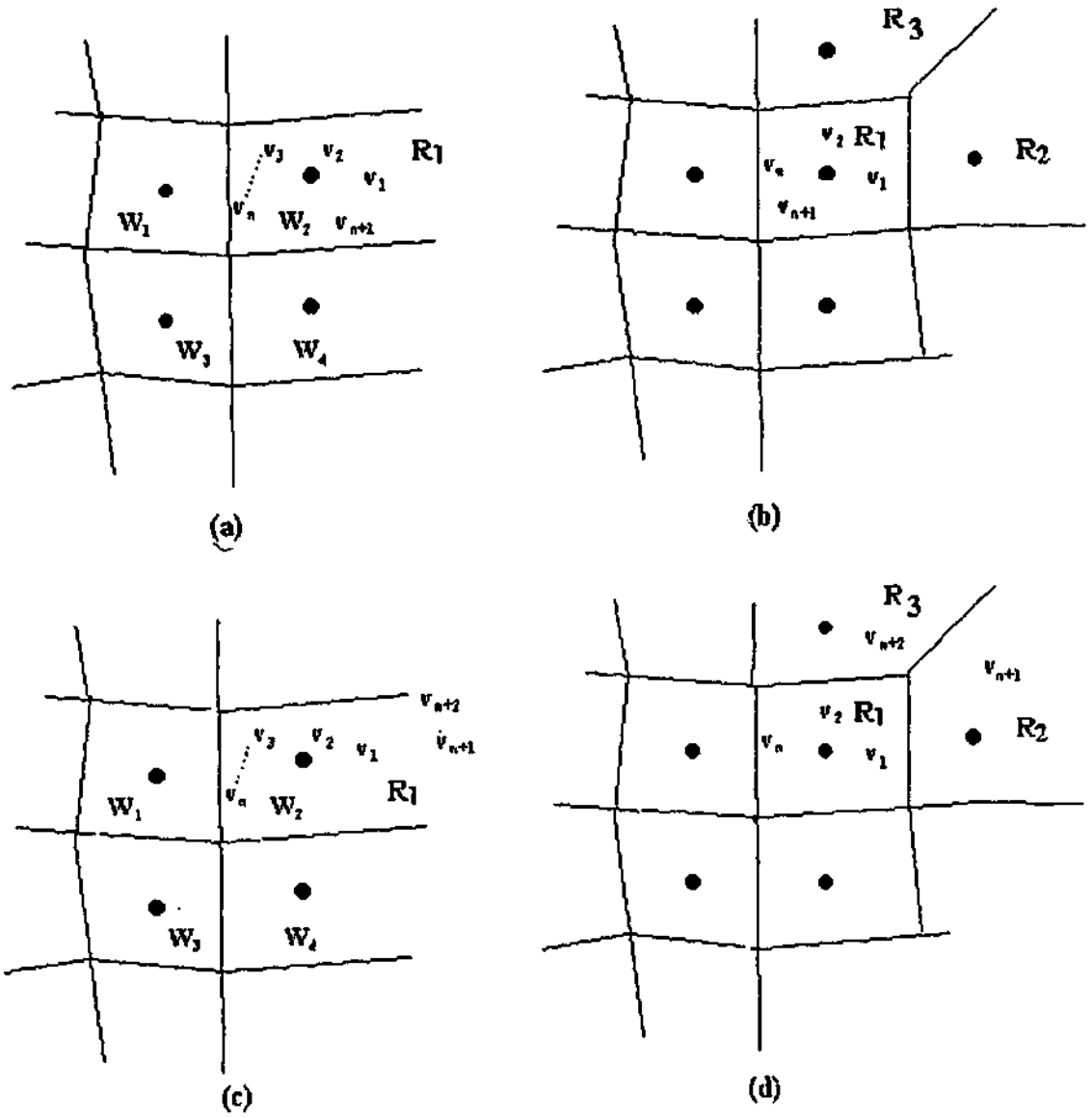


Figure 3.5: New node generation from the boundary of the network

due to the high density of inputs in the region R_1. Therefore new nodes will be generated as shown in Figure 3.5(b) (regions R_2, R_3), which will now compete with R_1 for the inputs. If the high error value in region R_1 is mainly due to a very high density in the area, further inputs will be assigned to region R_1 instead of being diverted to regions R_2 and R_3 (input values will be *closer* to R_1 than R_2

and R_3). In such a case, error of region R_1 will keep on accumulating without the ability to grow new nodes (R_1 is now a non-boundary region). This becomes a major limitation as it will result in dis-proportionate map areas being assigned to different subsets in the input. The traditional SOM relies on the self organisation process to *smooth out* this problem but we have developed a novel method which has shown good results. The significance of this limitation of dis-proportionate representation and our proposed measures for reducing such limitations is discussed in section 3.4.5.

Figure. 3.5(c) illustrates the situation where region R_1, is a boundary node and has been assigned inputs which should be mapped to further nodes if such nodes existed. That is, the region has been assigned with inputs (v_{n+1}, v_{n+2}) in the Figure 3.5(c)) which have large variances with weight W_2. As such the only reason the inputs have been assigned to the region is that it is the *closest* available due to the lack of suitable weight values. Since the error values for such inputs will be high, error (E) in the region will exceed GP mainly due to the high values of $(W_i - v_i)$ and not due to the high density in the region as discussed earlier. In this case (Figure 3.5(d)), new regions will be generated and the newly generated regions R_2 and R_3 will be assigned with some inputs (as new inputs are read in) which were earlier assigned to region R_1. Therefore, the creation of the new region results in the network *growing* to better accommodate the input data. Such growth results in the gradual generation of GSOM, where the rate of growth of

new nodes will decrease as sufficient nodes for representing the input data become available.

3.4.3 Justification of the New Weight Initialisation Method

In the SOM, the weight values are initialised with random numbers generated within the input range. The weight initialisation method of the GSOM is described in section 3.2. This method has been developed for the purpose of assigning weight values for the newly generated nodes such that the weights of the nodes are always in an *ordered state* according to their position in the network. We describe such an ordered state below by using a one dimensional array of nodes with position index values of 1 to n and with weight of w_1 to w_n respectively. If w_i is the weight vector of node i, the degree of ordering can be simply expressed by an *index of disorder* D as,

$$D = (\sum_{i=2}^{n} |w_i - w_{i-1}|) - |w_n - w_1| \tag{3.15}$$

where $D \geq 0$. This one dimensional array lets us intuitively picture the ordered state as the values $w_1, \ldots, w_n$ are ordered if and only if $D = 0$, which is equivalent to saying that $w_1 \geq w_2 \geq \ldots \geq w_n$ or $w_1 \leq w_2 \leq \ldots \leq w_n$. This can also be described as monotonically increasing or decreasing values of w_i.

There are two reasons for using such a new method for weight initialisation in the GSOM.

1. The new nodes in the GSOM are generated during the growing phase, interleaved with the self organising (weight adaptation) process. In fact, it is the weight adaptation process which decides when and where to generate new nodes. Therefore, a newly generated node will find its neighbours (and the whole map) has been partially *ordered* by the weight adaptation so far. Random weight initialisation at this stage would introduce *unordered* weights in to the partially ordered map. Although the GSOM algorithm has been developed to recover from such situations by initialising the learning rate (LR) for each iteration as described in section 3.4.1, unrecoverable distortions might still occur if the new weight has a large deviation from the neighbouring node weight values. The new weight initialisation method uses the weights of neighbouring nodes to calculate the weight values of the new nodes. Therefore chances of an un-recoverable distortion is greatly reduced.

2. Kohonen has described a possible weight initialisation method called the linear initialisation [Koh95], where any ordered initial state is described as *profitable*. The profitability in such a case is considered as faster and smoother convergence due to the ordered weight values letting the network directly start the learning with the convergence phase, whereas randomly initialised weights will first have to be ordered. This will allow the use of small values for LR ($0 \leq LR \leq 0.5$ instead of $0.5 \leq LR \leq 1$) at the

beginning, to approach the equilibrium smoothly. With the new weight initialisation method, the GSOM will always have an ordered set of weights. Therefore, once the node growing is complete, the smoothing phase will easily *smooth out* any existing deviations.

The weight initialisation method of GSOM therefore imposes the one dimensional definition of ordering in equation 3.15 to calculate the weight values for new nodes. Since the node growth occurs according to a two dimensional axis system, the two axis can be separately considered as two one dimensional arrays of nodes to define their *order* according to the above one dimensional definition. For example, for any node i, we consider the vertical and horizontal one dimensional arrays to confirm their *orderliness*. For an grid of $n \times m$ nodes, the confirmation of $n+(m-1)$ nodes for order can confirm the order of the whole grid. The weights are further smoothed out by the on going weight adaptation (self organisation) process.

3.4.4 Localised Neighbourhood Weight Adaptation

During SOM training, the neighbourhood (for weight adaptation) is large at the beginning and will shrink linearly to one node during the ordering stage. The *ordering* of the weight vectors w_i occurs during this initial period, while the remaining steps are only needed for the fine adjustment (convergence) of the map.

The GSOM does not require such an ordering phase, since new weight nodes are

initialised to fit in with their neighbourhoods, but require repeated passes over a small neighbourhood where the neighbourhood size reduces to unity. The starting (fixed) neighbourhood size can be considered small compared to that of SOM neighborhood since the initial neighbourhood in the SOM is normally considered to be the whole network. Therefore the GSOM achieves the convergence with localised neighbourhood adaptation. During the growing phase, the GSOM initialises the learning rate and the neighbourhood size, to the starting value at each new input. Weight adaptation is then carried out with reducing neighbourhood and learning rate until neighbourhood is unity, and initialises again for the next new input. It is possible that such weight adaptation will occur in overlapping neighbourhoods for consecutive inputs as can be seen in Figure 3.6 where the overlapping neighbourhood is shown by the shaded area.

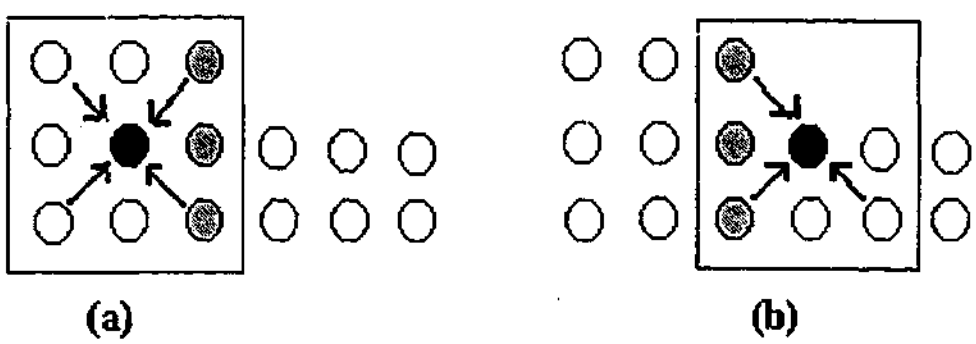


Figure 3.6: Overlapping neighbourhoods during weight adaptation

It is important that such localised neighbourhood weight adaptation does not create *disorder* among the existing ordered weights. We show that such a disorder

will not occur as follows. The weight adaptation of the GSOM can be considered as

$$\frac{dw_i}{dt} = \psi(n) \times LR \times (v - w_i) \tag{3.16}$$

for $i \in N_k$, and

$$\frac{dw_i}{dt} = 0 \quad Otherwise \tag{3.17}$$

An ordered set of weights $w_1, w_2, \ldots w_n$ cannot be disordered by the adaptation caused by equation 3.16, since if all partial sequences are ordered, then equation 3.16 cannot change the relative order of any pair $(w_i, w_j), i \neq j$. This proves that once the weights are ordered (by initialisation and self organisation), further weight adaptation according to equation 3.16 will not create disorder. Therefore, we can say that weight updating in overlapping neighbourhoods (overlapping for different inputs) will not create disorder in the map.

The outcome of the localised weight adaptation are

1. the reduction in processing speed due to the need of repeatedly initialising the neighbourhood size of new inputs in the GSOM is offset by the small initial neigbourhood size compared to the SOM. Having to start with a large neighbourhood results in slow processing in the SOM.

2. the small neighbourhood also results in the need of updating weights of fewer nodes. This also results in faster processing of inputs in the GSOM.

3.4.5 Error Distribution of Non-boundary Nodes

The GSOM generates new nodes only from the boundary of the network. The advantage of this is that the resulting network is always a two dimensional grid structure which is easier to visualise. A limitation with this method arises when the *error* value of a non-boundary node exceeds the growth threshold (GT) due to high density areas in the data. In such instances, the non-boundary node is unable to generate new nodes. This will result in certain non-boundary nodes accumulating a large number of hits due to high density regions in the input data space. Therefore the resulting map may not give a proportionate representation of the input data distribution. Such proportionate representation has been called as the *Magnification Factor* [Koh95]. In biological brain maps, the areas allocated to the representation of various sensory features are often believed to reflect the importance of the corresponding feature sets. Although we do not attempt to justify the GSOM with biological phenomena, such proportional representation of the frequency of occurrence by the area in a feature map would be useful for a data analyst to immediately identify regions of high frequency, thus giving some idea about the distribution of the data.

Due to the restriction of only being allowed to grow from the boundary, it was found that other than for the very simple data sets, the GSOM will stop *growing* when a certain amount of spread has been achieved. This occurs when sufficient

nodes to map the different *regions* of input data has been generated depending on the order of input presentation to the network. For example, if the data is presented in some sorted order, the map will spread in an orderly fashion in one direction, but when the data is presented randomly, an input data region with high density can get caught inside the network due to being mapped to a node which has become a non-boundary node. In this case, the high density region will keep on accumulating hits without the ability to spread out.

In the SOM this situation is left to be handled by the self organisation process, and sufficient number of iterations will result in the map spreading out, provided a large enough grid is created initially. In the GSOM since the requirement is to grow nodes as and when required, there has to be a mechanism to initiate growth from the boundary for the map to spread out. We have implemented the following error distribution mechanism to handle such a situation.

When a boundary node is selected for growth, the error value of a non-boundary nodes in the neighbourhood is reduced according to the equation below.

$$E^w_{t+1} = GT/2 \tag{3.18}$$

where E^w_t is the error value of the winner, and GT the growth threshold. The error value of the immediate neighbours of the winner are increased as

$$E^{n_i}_{t+1} = E^{n_i}_t + \gamma E^{n_i}_t \tag{3.19}$$

where $E_{t+1}^{n_i}(i = 1..4)$ is the error value of i^{th} neighbour of the winner E_t^w and γ is a constant value called the *Factor of Distribution* (FD) which controls the error increase and $0 < \gamma < 1$.

By using equation 3.18, the error value of the high error node is reduced to half the growth threshold. With equation 3.19, the error value of the immediately neighbouring nodes are increased, therefore the two equations produce an effect of spreading the error outwards from the high error node. This type of spreading out will in time (iterations) ripple outwards and cause a boundary node to increase it's error value. This is pictorially shown in Figure 3.7.

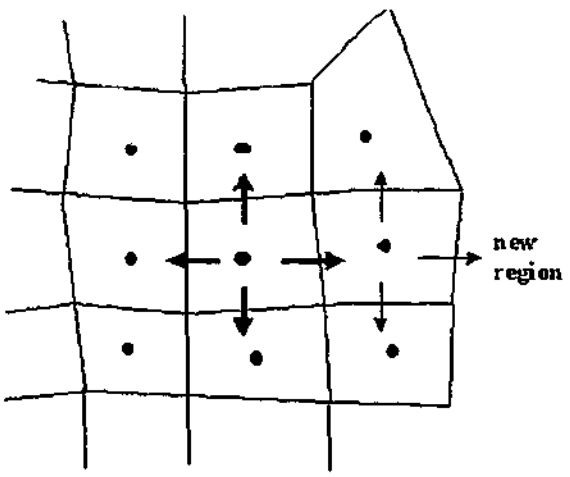


Figure 3.7: Error distribution from a non-boundary node

Therefore the purpose of the equations 3.18 and 3.19 is to give the non-boundary nodes some ability in initiating (although indirectly) the node growth.

The additional equations result in a better spread map which provides easier

visualisation of the clusters. To demonstrate the significance of this method experimentally, we use a data set consisting of 99 animals. The data is 17 dimensional and mainly consist of binary attributes, and the full data set is listed in appendix A. For the experiments described 16 of the 17 attributes are selected, leaving out the *type*. We let the unsupervised GSOM algorithm cluster the data without introducing the pre-defined bias by providing the *type* as an attribute. The non-binary attribute (number of legs) is *coded* to take values in the range [0..1]. Figure 3.8 shows the GSOM generated for the data without applying the error distribution equations 3.18 and 3.19 and Figure 3.9 shows the GSOM produced by using the formulas. It can be seen from Figure 3.8 that the map gives a very congested view of the data. This is due to the fact that the growth can only occur from the boundary of the map. It can be seen that the Figure 3.9 gives a much better spread of the data set.

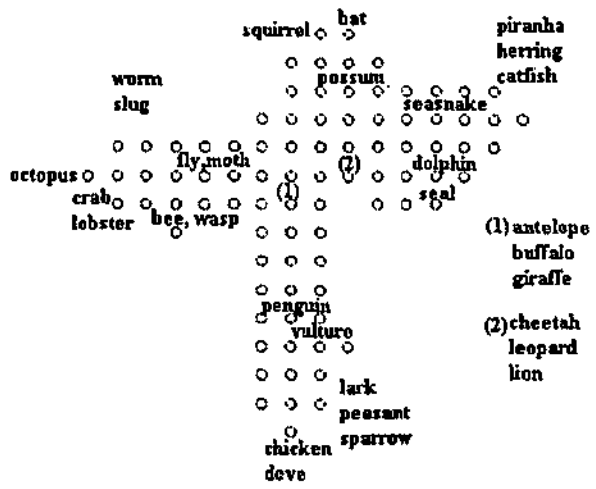


Figure 3.8: The animal data set mapped to the GSOM, without error distribution

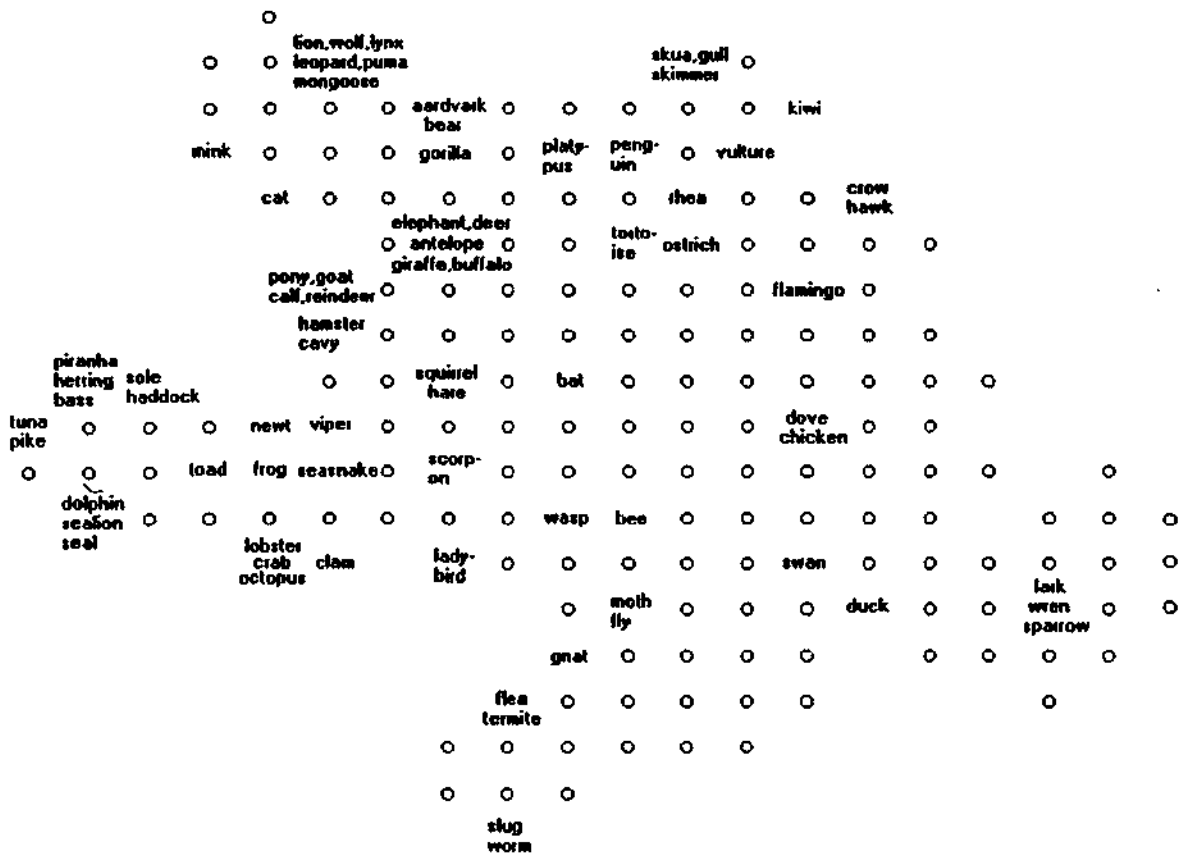


Figure 3.9: The animal data set mapped to the GSOM, with error distribution

3.4.6 The Spread Factor (SF)

As described in section 3.2, the GSOM uses a threshold value called the growth threshold (GT) to decide when to initiate new node growth. GT will decide the amount of spread of the feature map to be generated. Therefore, if we require only a very abstract view of the data, a large GT will result in a map with a fewer number of nodes. Similarly, a smaller GT will result in a well spread out map. When using the GSOM for data mining, it is a good approach to first generate a *smaller* map, only showing the most significant clustering in the data, which will give the data analyst a summarised picture of the inherent clustering in the total

data set.

The node growth in the GSOM is initiated when the error value of a node exceeds the GT. The total error value for node i is calculated as

$$TE_i = \sum_{H_i} \sum_{j=1}^{\mathcal{D}} (x_{i,j} - w_j)^2 \tag{3.20}$$

where H is the number of hits to the node i and $\mathcal{D}$ is the dimension of the data. $x_{i,j}$ and w_j are the input and weight vectors of the node i respectively. The requirement for new node growth can be stated as

$$TE_i \geq GT \tag{3.21}$$

The GT value has to be experimentally decided depending on the requirement for map growth. As can be seen from equation 3.20 the dimension of the data set will make a significant impact on the accumulated error (TE) value, and as such will have to be considered when deciding the GT for a given application or data. Since $0 \leq x_{i,j}, w_j \leq 1$, the maximum contribution to the error value by one attribute (dimension) of an input would be :

$$max|x_{i,j} - w_j| = 1$$

Therefore, from equation 3.20 :

$$TE_{max} = D \times H_{max} \tag{3.22}$$

Where TE_{max} is the maximum error value and H_{max} is the maximum possible number of hits (the number of times the particular node became the winning

node). If we consider $H(t)$ to be the number of hits at time (iteration) t, the growth threshold (GT), will have to be set such that :

$$0 \leq GT < D \times H(t) \tag{3.23}$$

Therefore we have to define a constant GT, depending on our requirement of the map spread. The GT value will depend on the dimensionality of the data as well as the number of hits. Identifying an appropriate GT value for an application can be a difficult task specially in applications such as data mining, where it is necessary to analyse data with different dimensionality as well as the same data under different attribute sets. It also becomes difficult to compare maps of several data sets since the GT cannot be compared over different data sets if their dimensionality is different.

Therefore we define a term called the Spread Factor (SF) which can be used to control and calculate the GT for GSOMs, without the data analyst having to worry about the different dimensions. We redefine GT as

$$GT = D \times f(SF) \tag{3.24}$$

where $SF \in R$, $0 \leq SF \leq 1$ and $f(SF)$ is a function of SF, which is defined as follows.

We know that the total error TE_i of a node i will be

$$0 \leq TE_i \leq TE_{max} \tag{3.25}$$

where TE_{max} is the maximum error value that can be allowed. Substituting equation 3.20 in equation 3.25 we get :

$$0 \leq \sum_{H} \sum_{j=1}^{D} (x_{i,j} - w_j)^2 \leq \sum_{H_{max}} \sum_{j=1}^{D} (x_{i,j} - w_j)^2 \tag{3.26}$$

Since the purpose of the growth threshold (GT) is to let the map grow new nodes by providing a threshold for the error value, and the minimum error value is zero, it can be said that, for growth of new nodes we have the relationship

$$0 \leq GT \leq \sum_{H_{max}} \sum_{j=1}^{D} (x_{i,j} - w_j)^2 \tag{3.27}$$

Theoretically we can assume the upper bound on the number of hits (H_{max}) to be infinity

$$0 \leq GT \leq \infty$$

Hence we need to identify a function $f(SF)$ such that

$$0 \leq SF \leq 1$$

and

$$0 \leq D \times f(SF) \leq \infty$$

In other words we require a function $f(x)$ which takes the values 0 to ∞, when x takes the values 0 to 1. Napier logarithmic function of the type $y = -a \times \ln(1-x)$ is one such function which satisfies these requirements [Hor99]. Letting x be SF and substituting D for a,

$$GT = -D \times \ln(SF) \tag{3.28}$$

Therefore, instead of having to provide a GT, which would take different values for different data sets, the data analyst has to provide a value SF, which will be used by the system to calculate the GT value depending on the dimension of the data. This will allow different GSOMs to be identified with their spread factors, and can form a basis for comparison of different maps.

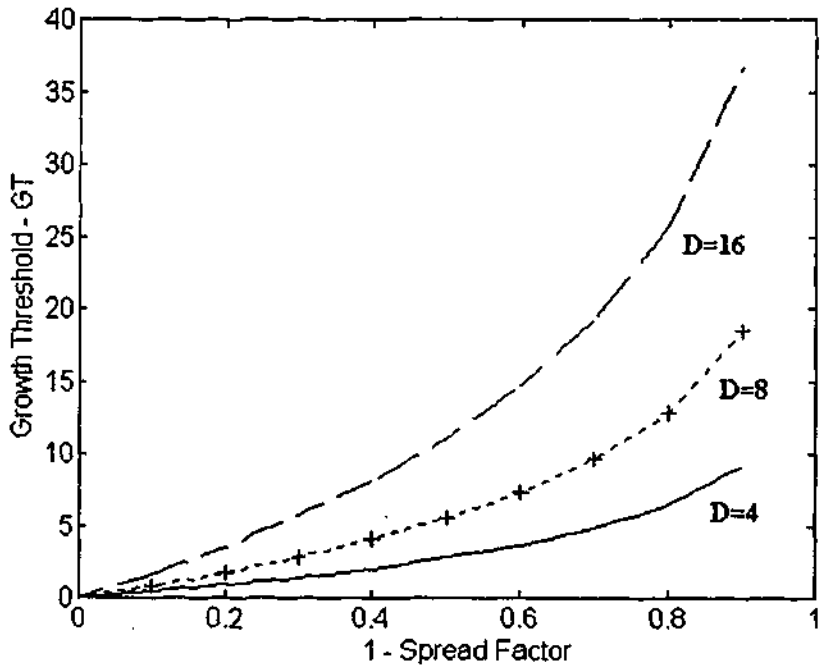


Figure 3.10: Change of GT values for data with different dimensionality (D) according to the spread factor

The graph in Figure 3.10 shows how the GT value of data sets with different dimensions change according to the spread factor. The SF can also be used to implement hierarchical clustering with the GSOM by initially generating a

small map with a low SF and then generating larger maps on selected regions of interest. Such hierarchical clustering will be described in detail in chapter 6.

3.5 Applicability of GSOM to the Real World

In this section we present several experiments to demonstrate the functionality of the GSOM. Experiments are carried out with bench mark data sets. The main purpose of these experiments is to introduce the GSOM as a novel feature mapping method which has significant advantages over the traditional SOM. In the first experiment, an artificial data which has been used by Kohonen [Koh95] to demonstrate the SOM is used to compare the SOM with the GSOM. In the second experiment, the iris data set [Bla98] is used to highlight the usefulness of the GSOM for data mining and exploratory analysis of a traditionally *supervised* data set.

3.5.1 Experiment to Compare the GSOM and SOM

Table 3.1 shows an artificial data set used by Kohonen to highlight the features of the SOM. The data consist of 13 binary attributes on 16 well known animals. The animals have been selected such that they belong to several clusters (groups) such as *birds*, *meat eating mammals*, *non meat eating mammals* and *domestic animals*. Once the feature map is produced, the clusters in the data can be identified and

also some inter and intra cluster relationships become visible from the spatial positioning of the clusters in the map.

Table 3.1: Kohonen's animal data set

	dove	hen	duck	goose	owl	hawk	eagle	fox	dog	wolf	cat	tiger	lion	horse	zebra	cow
small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
fly	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

A SOM with a 5×5 node structure was first presented with the data. After the organisation process the network was calibrated with the same data for the purpose of identifying the distribution of the animals in the network. The resulting output is shown in Figure 3.11 (a). A similar experiment as above is carried out using the same data, on a SOM with a 10×10 node structure. The resulting map with the animal names are shown in Figure 3.11 (b). The same data set is used input to generate a GSOM to compare with the respective SOMs. All maps were generated with an initial learning rate of 0.1, and the spread factor of 0.4

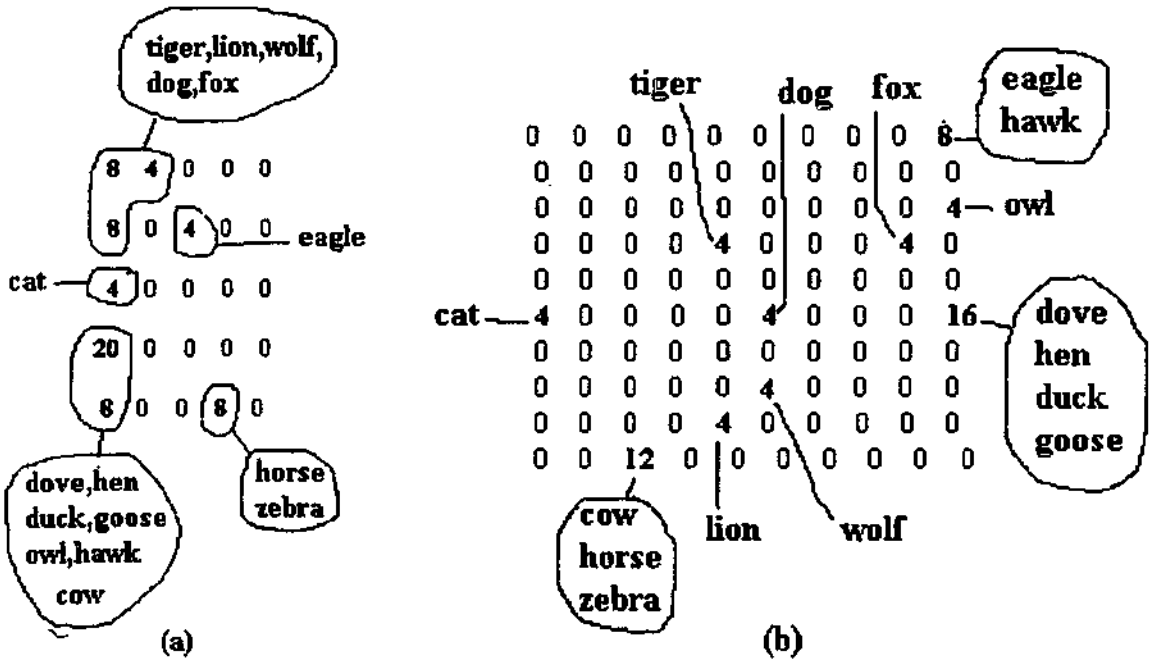


Figure 3.11: Animal data set mapped to a 5 × 5 SOM (left) 10 × 10 SOM (right)

for the GSOM. The GSOM is shown in Figure 3.12.

It can be seen from the 5 × 5 map (Figure 3.11 (a)) that data has been mapped on to separate sections of the network according to the characteristics of the animals. We can very roughly identify three (3) clusters as the *four legged meat eaters*, *birds* and *grass eaters*. But we arrived at this conclusion only because of our clear knowledge of the input data set. We could have otherwise concluded that there is one large cluster (meat eaters and birds) and a small cluster of grass eaters.

The 10 × 10 map (Figure 3.11 (b)) spread out the *four legged meat eaters* cluster

across the network. This spreading out is necessary to identify the differences and similarities within the clusters (between the members of a cluster) and between different clusters. The birds and the grass eaters clusters have not been spread out. Once a SOM is organised the weight vectors of the nodes will represent (by the position in the map) the inherent characteristics of the attributes of the input data. From this experiment it can be seen that the advantage of the SOM is partly lost if we do not use a map of the proper size, as some of the clusters would be piled up and mapped to the same set of nodes. This would be due to the size of the map not being sufficiently large enough to represent the spread of the clusters. As such, the spatial relations between and inside the clusters are deformed when the appropriate network shape and size is not initialised.

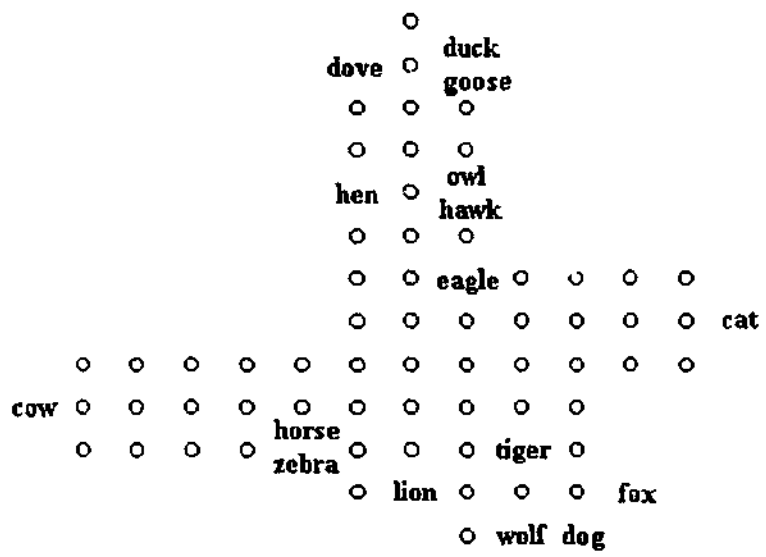


Figure 3.12: Kohonen's animal data set mapped to a GSOM

The clusters in the data are more clearly visible and has meaningful groupings

in the GSOM (Figure 3.12). The basic cluster structure of the *birds*, *four legged meat eaters* and *grass eaters* is apparent. In addition, we can distinguish that horse and zebra has been positioned closer to the other wild animals, and away from the cow. The horse and zebra has moved closer to the lion due to having a mane. The eagle being a bird is positioned with the birds but has moved towards the other meat eaters. Therefore, we can distinguish a further sub-clustering inside the main clusters. ie *meat eating birds* such as (eagle, hawk, owl), *grass eating wild animals* such as (horse, zebra). The cow and the cat are two clusters by themselves. The cat being the only small four legged animal, and the cow being the only grass eater which does not have a mane, and not in the *running* category, are the most probable reasons.

The main advantage for visualising the clusters with the GSOM is its ability to spread outwards in a manner such that the clusters in the data dictate the shape of the map. For example, in Figure 3.12 the final shape of the map is generated by spreading out in all directions thus highlighting the *birds*, *mammals*, *cat*, and *cow* groupings. Thus it can be seen that the GSOM has produced a more informative clustering structure than that of the corresponding SOMs. Analysing the distances between the clusters and the sub clusters in the GSOM would give us a further understanding of the data.

3.5.2 Applicability of the GSOM for Data Mining

In this section we present an experiment to highlight the usefulness of the GSOM as an exploratory data analysis tool in data mining. The data set selected is the iris data, which is a popular bench mark data set in classification literature. The difference in our experiment is that the data is used for *unsupervised* clustering as opposed to the traditional usage of *supervised* classification.

In his introduction on the method of discriminant analysis, R.A. Fisher presented an analysis of measurements on Irises [Bla98]. There were 50 flowers from each of three species of Iris - Iris setosa, Iris versicolor, and Iris virginica. There were four measurements on each flower - petal length, petal width, sepal length, and sepal width. Because species type is known, the problem is one of statistical pattern recognition, and the data can be analyzed using discriminant analysis or a supervised learning approach. A summary of the statistics for the data set is presented in table 3.2. In this experiment, Fisher's iris data set is used to demonstrate the usefulness of the GSOM as an unsupervised data mining tool. Therefore, instead of the traditional expected outcome of the accuracy of the 3 cluster identification, we are interested in finding whether there are any *unforeseen* groupings or sub-groupings in the data. The attribute values were scaled to the range 0.0 to 1.0 for inputing to the the GSOM. Since 150 values would not be sufficient to train the GSOM, the data set is repeated several times to make

Table 3.2: Summary statistics of the iris data

	Min	Max	Mean	SD	Class Correlation
Sepal length	4.3	7.9	5.84	0.83	0.7826
Sepal width	2.0	4.4	3.05	0.43	-0.4194
Petal length	1.0	6.9	3.76	1.76	0.9490
Petal width	0.1	2.5	1.20	0.76	0.9565

up the input data set. The resulting maps are shown in Figures 3.13 and 3.14.

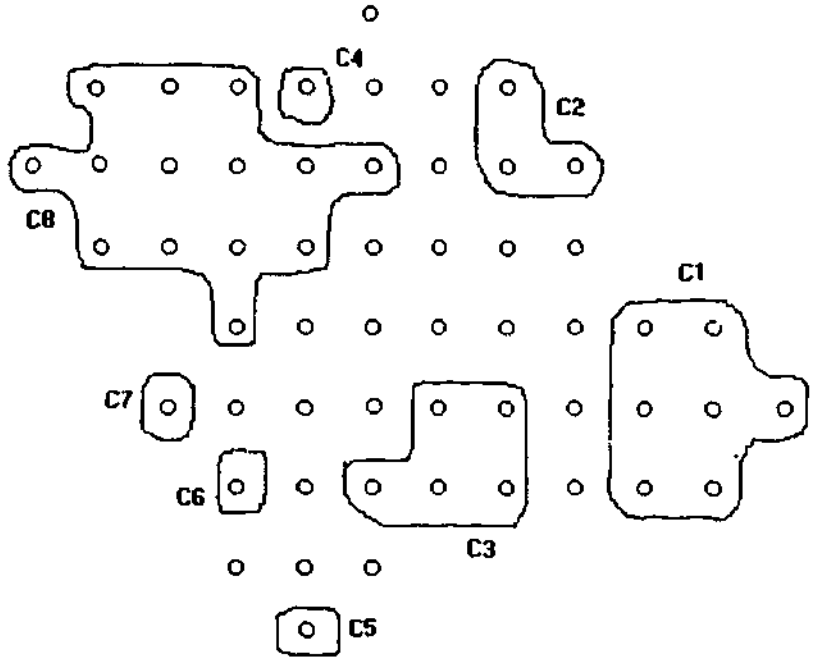


Figure 3.13: Unsupervised clusters of the Iris data set mapped to the GSOM

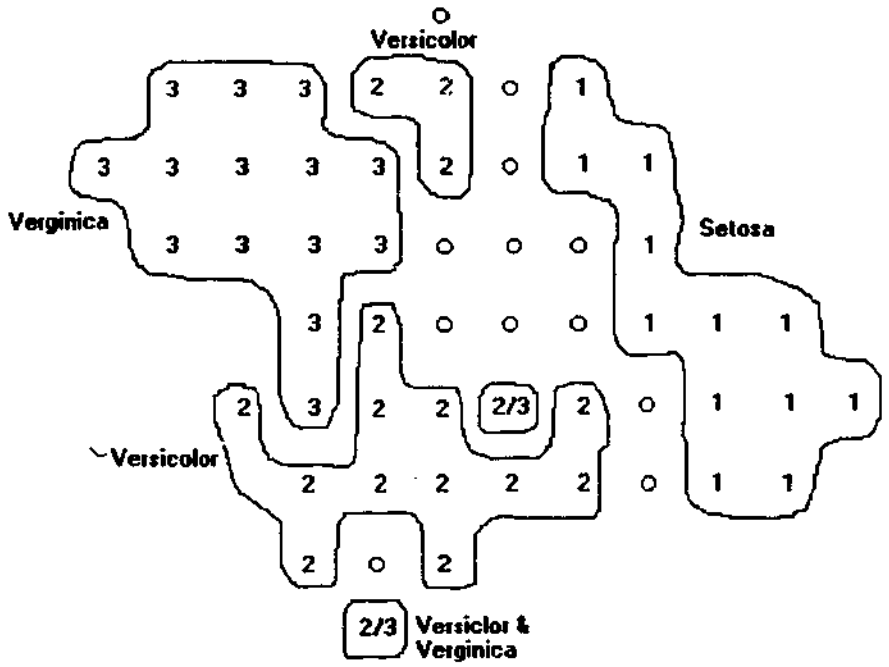


Figure 3.14: Iris data set mapped to the GSOM and classified with the iris labels, 1 - Setosa, 2 - Versicolor and 3 - Verginica

Figure 3.13 shows the mapping of the number of *hits* when the 150 instances were calibrated after training. The purpose of this map is to try and identify how the instances have clustered without giving any consideration to the class of irises. With this experiment we do not attempt to classify the inputs into the 3 classes of irises. Since the GSOM is an unsupervised learning method, we let the network learn the structure in the input data without using any known structure in the input to train the network.

Figure 3.13 shows the clusters which have been identified from the formed map. The clusters are initially visually identified. Then the significant sub-clusters are

separated by comparing the output of nodes within each cluster. Therefore we have used the spatial positioning of the nodes, as well as the outputs to separate the clusters and sub-clusters. For simplicity, only those nodes which have been mapped more than once (more than one hit), were considered for the clustering, since our aim is to demonstrate the usefulness of the method more than to accurately identify the clusters. It is seen from Figure 3.13 that 7 separate clusters have been identified and labeled as C1 to C7.

The clusters can now be described by the max, min, mean, standard deviation of their attribute values. This will provide an identity for the clusters since there are no labels as in the supervised clustering methods. For example, cluster 6 (C6) and cluster 7 (C7) can be represented by table 3.3 and table 3.4. It can be seen that clusters 6 and 7 have very small variation (6.5%-8%) in their attribute values. A data analyst has the option of deciding whether such a difference is significant or interesting for the application. In most data mining applications, the analyst is initially interested in obtaining an overview of the data and as such only require an overview of the most significant groupings. Therefore, if the small difference in clusters are not sufficiently significant, the clusters 6 and 7 may be merged together and considered as one cluster at this initial *overview* stage of the analysis. In such a case, if the analyst decides to conduct a more detailed analysis at a later stage, the two clusters can be considered separately.

Table 3.3: Cluster 6 attribute summary

	Sepal length	Sepal width	Petal length	Petal width
Average	6.4	2.9375	4.4875	1.375
Max	6.7	3.1	4.7	1.5
Min	6.1	2.8	4.3	1.3
Std dev	0.2390	0.0916	0.1553	0.0707

Table 3.4: Cluster 7 attribute summary

	Sepal length	Sepal width	Petal length	Petal width
Average	6.82	3.04	4.82	1.5
Max	7	3.2	5	1.7
Min	6.7	2.8	4.7	1.4
Std dev	0.1304	0.1517	0.1304	0.1225

Cluster 1 (C1 presented in table 3.5) shows significant difference from both cluster 6 and cluster 7. These results are also visually presented by the cluster positions in the GSOM. By this type of analysis it is also identified that cluster 5 (C5) consists of irises with sepal width significantly lower than the rest of the irises.

Table 3.5: Cluster 1 attribute summary

	Sepal length	Sepal width	Petal length	Petal width
Average	4.7	3.02	1.42	0.18
Max	5	3.2	1.6	0.3
Min	4.3	2.3	1.1	0.1
Std dev	0.2186	0.2085	0.1334	0.0562

The analysis so far has been without considering the 3 classes of irises. When such information is available we can make use of this *knowledge* to gain further insight into the data. Figure 3.14 shows the 3 classes of irises mapped to the GSOM. The labels 1, 2 and 3 refer to Setosa, Versicolor and Verginica respectively. Setosa has been separated from Versicolor and Virginica, which have been mapped to one large cluster. Versicolor has separated into two clusters on either side of Virginica. Virginica has been mapped to near-by nodes except for 3 instances which have been separated.

Comparing Figures 3.13 and 3.14 it can be seen that the main Setosa cluster has two significant sub clusters. The reason for this sub clustering can be analysed which may provide interesting information regarding such sub-groupings. It is further possible to find the reason for the separation of Versicolor into two clusters,

and also the reason for 3 instances of Virginica moving away from the rest of the instances. Such analysis can further be carried out to extract, more useful information regarding the data set. Only some of the possible instances have been highlighted in this analysis to demonstrate the potential that arise from such analysis. Further detailed data analysis with the GSOM will be described in chapter 5 and chapter 6.

3.6 Summary

In this chapter we presented the main contribution of this thesis which is a novel unsupervised neural network model called the Growing Self Organising Map (GSOM). Initially, the concept of the GSOM is discussed and the algorithm is presented in detail. The GSOM has 3 main phases of generation (initialisation, growth, and smoothing) and these phases are discussed and justified. The new features included in the GSOM have been introduced and an analysis and justification of the implication and methodology for choosing values for the parameters have been presented.

The proposed structure adapting feature map has several advantages over the traditional feature maps. The main advantage being the self generating ability and therefore the network designer not having to pre-define the network structure. In addition the following advantages are also present in the GSOM.

1. Enhanced visualisation of the clusters by spreading out the feature map in to shapes which are representative of the inherent distribution in the data. Therefore the GSOM has a cluster driven shape thus highlighting the data clusters by the shape of the network itself.

2. The number of nodes in the final map is found to be less than required for the traditional SOM. Therefore with the GSOM, it is not necessary to process nodes which are *unnecessary* for representing the data set.

3. Smoother training without the need for an ordering phase due to the novel weight initialisation method, which also reduces the possibility of twisted maps.

4. The ability to compare and control the growth of feature maps with the use of a spread factor (SF).

Experiments are carried out to demonstrate

1. the comparison of the feature maps created by the SOM and the GSOM.

2. the use of GSOM in mapping a well known benchmark data set for exploratory data analysis.

The results show the potential of the GSOM, and this will be further demonstrated in the subsequent chapters. In the next chapter we present a novel method of automating the cluster identification and separation using the incrementally generating nature of the GSOM.

Chapter 4

Data Skeleton Modeling and Cluster Identification from a GSOM

4.1 Introduction

The novel self generating neural network model called the GSOM is introduced in chapter 3. The differences between the traditional SOM and the GSOM are also discussed, highlighting the advantages of the GSOM due to its flexible structure. It is seen that the GSOM *grew* nodes and spread out, while it self organised to generate a structure which better represents the input data. The resulting feature maps are of different shapes and sizes and it is seen that the shapes of the maps resulted from the inherent clustering present in the data. Therefore the

GSOM clusters were easier to visually identify compared to the SOM clusters by observing the *directions* of growth.

In many current commercial and other applications, the clusters formed by feature maps are identified visually. Since the SOM is said to form a topology preserving mapping of the input data, it is possible to visually identify the clusters and some relationships among them by studying the proximity or the distance among the clusters. Although the accuracy obtained from such visualisation is not very high, it has proved sufficient for many applications, specially, in industry as a tool for database segmentation [Big96], [Deb98]. It is shown in chapter 3, that the GSOM highlights the clusters by branching out in different directions, thus making it easier to identify the clusters visually.

Visually identifying the clusters can have certain limitations such as :

- It has been shown in [Rit92] that the SOM does not provide complete topology preservation. Therefore it is not possible to accurately translate the inter-cluster distances to a measure of their *similarity* (or *difference*). Therefore visualisation may not provide an accurate picture of the actual clusters in the data. This would specially occur in a data set with a large number of clusters with a skewed distribution, and will result in erroneous allocation of data points into clusters due to the inaccurate identification of cluster boundaries.

- In certain instances it is useful to automate the cluster identification process. Since clusters in a data set are dependent on the distribution of data, it would be difficult to completely automate the process unless parameters such as the number of clusters or the *size* of a cluster are pre-defined. A useful partial automation can be implemented whereby the system will provide the analyst with a number of clustering options. For example, the system can provide the analyst with a list of distances between groupings in data and allow the analyst to make the final decision as to the optimal clustering for a given situation. Having to visually identify the clusters will be a hindrance in automating such a process.

In this chapter a method for automating the cluster identification process is proposed. This method takes into consideration the shape of the GSOM, as well as the visual separation between data to identify the clusters. The advantages of this method are the identification of more accurate clusters, minimisation of the possibility of erroneous allocation of data into clusters and the possibility of automating the cluster identification process.

In section 4.2 the usefulness of an automated cluster identification for data mining is highlighted. The methods that can be employed for automated identification are discussed and their limitations are identified. Section 4.3 presents a description of the proposed method and the proposed algorithm. Artificial and real data

sets are used to demonstrate the applicability and usefulness of this method in section 4.4. Section 4.5 presents a summary of this chapter.

4.2 Cluster Identification from Feature Maps

In this section we provide the basis and justification for the automated cluster identification method from the GSOM. We first introduce the traditional SOM as a vector quantisation algorithm which can be used to identify a set of representative quantisation regions from a given set of data. Then we discuss the possible methods of identifying clusters from such a quantised set of data by using a feature map. The difficulties faced by an analyst in identifying clusters from a traditional SOM are also highlighted and the advantage of the GSOM in this regard is discussed.

4.2.1 Self Organising Maps and Vector Quantisation

Vector quantisation is a technique that exploits the underlying structure of input vectors for the purpose of data compression or equivalent bandwidth compression [Hay94]. This method supposes that the input data are given in the form of a set of data vectors $x(t), t = 1, 2, 3, \ldots$ (generally in high dimension). The index t identifies the attributes of the individual vectors. In vector quantisation, an input data space is divided into a number of distinct regions and a *reconstruction (reference) vector* is defined for each such region. This can be considered as defining

a finite set W of reference vectors, such that a *good* approximate vector $w_s \in W$ can be found for each input data vector $x(t)$. The set of such reconstruction vectors is called a *vector quantiser* for a given input data. When the quantiser is presented with a new input vector $x(t)$, the region in which the vector lies is first determined, by identifying the reference vector w_s with the minimum norm of the difference $\delta = |x(t) - w_s|$. From then $x(t)$ is represented by the reconstruction (reference) vector w_s. The collection of possible reconstruction vectors is called the *codebook* of the quantiser.

A vector quantiser with such minimum encoding distortion is called a *Voronoi quantiser*. When the Euclidean distance similarity measure is used to decide on the region to which an input vector belongs, the quantiser is a Voronoi quantiser. The Voronoi quantiser partitions its input space into Voronoi cells or regions, and each region is represented by one of the reconstruction vectors w_i. The i^{th} Voronoi region contains those points of the input space which are closer (in a Euclidean sense) to the vector w_i than to any other vector $w_j, j \neq i$. Figure 4.1 shows an input space divided into four Voronoi cells with the respective Voronoi vectors shown as circles. Each Voronoi cell contains those points of the input space that are the closest to the Voronoi vector (circle in Figure 4.1) among the totality of such points.

The Self Organising Feature Map (SOM) , represented by the set of weight vectors

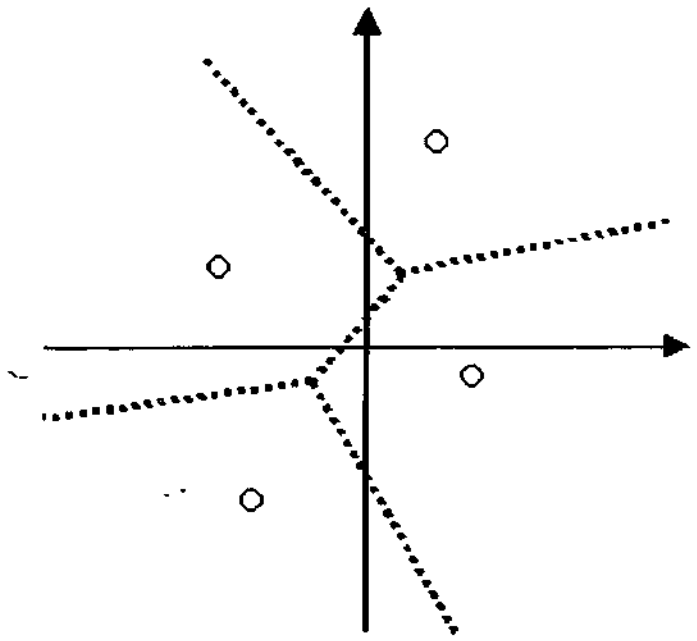


Figure 4.1: Four Voronoi regions

$\{w_j | j = 1, 2, ..N\}$, is said to provide a good approximation to the input data space [Hay94]. It can be interpreted that the basic aim of the SOM algorithm is to store (represent) a large set of input vectors by finding a smaller set of reference vectors, so as to provide a *good* approximation of the input space. The basis of this idea is the *vector quantisation theory* as described above and can be used as a method of dimensionality reduction or data compression. Therefore the SOM algorithm can be said to provide an approximate method for computing the Voronoi vectors in an unsupervised manner, with the approximation provided by the weight vectors in the feature map.

4.2.2 Identifying Clusters from Feature Maps

As described in the previous section, the SOM provides a vector quantisation of a set of input data by assigning a reference vector for each value in the input space. The mapping of input data to the reference vectors is a *non-linear projection* of the probability density function of the high dimensional input data space on to a two dimensional display [Koh95]. With this projection, the SOM achieves a dimensionality reducing effect by preserving the *topological relationships* that exist in the high dimensional data into a two dimensional map. Such topology preserving ability has been described as mapping of the features in the input data and as such these maps have been called *feature maps* [Koh95]. We can thus identify the main advantages of the feature maps as two fold.

1. With the vector quantisation effect, the map provides a set of reference vectors which can be used as a codebook for data compression.

2. The map will also result in producing a two dimensional topology preserving map of a higher dimensional input space, thus making it possible to visually identify similar groupings in the data.

Therefore the feature map provides the data analyst with a representative set of two dimensional feature vectors, for a more complex and multi dimensional data set. The analyst can then concentrate on identifying any patterns of interest in this codebook using one or more of the several existing techniques.

The focus of this thesis is the GSOM, which is a neural network developed using an unsupervised learning technique. As described in chapter 2, the advantage of unsupervised learning is that it is possible to obtain an unbiased segmentation of the data set without the need for any external *guidance*. Therefore we will now consider some existing techniques which can be used on the feature map to identify the clusters. We will then use these techniques as the basis for justifying a novel method of cluster identification from the GSOM. The three main techniques, which can be used to identify clusters from a feature map are described below.

K-means Algorithm

The K-means algorithm was first introduced by McQueen in 1967, and also as ISODATA by Ball and Hall in 1967 [Mir96]. The steps of the algorithm are as follows :

1. Select K seed points from the data. The value K, which is the number of clusters expected from the data set has to be decided by the analyst using prior knowledge and experience. In some instances the analyst may even decide that the data needs to be segmented into K groups for the requirements of a particular application.

2. Consider each seed point as a cluster with one element and assign each input record in the data set to the cluster *nearest* to it. A distance metric

such as the Euclidean distance is used for identifying the *distance* of each input vector from the initial K seed points.

3. Calculate the centroid of the K clusters using the assigned data records. The centroid is calculated as the *average* position of all the records assigned to a cluster on each of the dimensions. i.e. if data records for the j^{th} cluster can be denoted as $x_1, x_2, \ldots, x_n$, (each with D dimensional information), then the centroid for the clusters is defined as

$$Cluster_{j,centroid} = ((\sum_{i=1}^{n} x_{i,1})/n, (\sum_{i=1}^{n} x_{i,2})/n, \ldots, (\sum_{i=1}^{n} x_{i,D})/n)$$

where D is the dimension of the data.

4. Re-assign each of the data records to the cluster centroid nearest to it, and recalculate the cluster centroids.

5. Continue the process of re-asssiging records and calculating centroids until the cluster boundaries stop changing.

There are several approaches when applying the K-means algorithm to a feature map.

1. Consider all the nodes (reference vectors are represented by the nodes) which have been assigned with at least K, where $K \geq 0$, input vectors (records), as seed points. The value K will have to be decided by the analyst. The other nodes are then assigned to the clusters according to the above algorithm.

2. Pre-define the number of seed points with external knowledge or according to the needs of the application and randomly assign nodes from the map as the seed points. The rest of the nodes are then assigned to the seed points according to the algorithm described above.

In the first approach the advantage is that the analyst uses information from the initial mapping to decide the seed points. Hence, the decision is not completely biased on pre-conceived ideas and opinions. The deciding of the value K as a threshold for deciding the *important* nodes still can cause problems. In the second approach, the analyst is completely depending on external knowledge which may introduce bias and also may be difficult with *unknown* data.

Agglomeration methods

In the K-means method the analyst has to start with a fixed number of clusters and gather data records around these points. Another approach to clustering, is the agglomeration methods [Sch96]. In these methods, the analyst will start with each point in the data set as a seperate cluster and gradually merge clusters until all points have been gathered into one cluster (or a small number of clusters). At the initial stages of the process the clusters are very small and very pure, with the members of the clusters being very closely *related.* Towards the end of the process the clusters become very large and less well defined. With this method the analyst can preserve the entire history of the cluster merging process and has the advantage of being able to select the most appropriate level of clustering for

the application. The main disadvantage with this method is that, for a very large data set, it will be almost impossible to start with each element of the data set as a seperate cluster.

In feature maps, all the nodes with at least one input assigned to it can be considered as an initial cluster. The *nearby nodes* can then be merged together until the analyst is satisfied, or a threshold value can be used to terminate the cluster merging. Therefore the data compression ability of the feature map will become very useful when using an agglomeration method on a large set of data since this will cut down the number of clusters to be processed.

Divisive methods

Divisive methods [Sch96] are similar to agglomeration methods except it uses a top down cluster breaking approach compared to the bottom up cluster merging method used in the agglomeration method. The advantage of this method is that the whole data set is considered as one cluster at the beginning. Therefore it is not necessary to keep track of a large number of seperate clusters as in the initial stages of the agglomeration method. A distance method for calculating the distance between clusters will have to be defined to identify the points of cluster separation by defining a threshold of separation. Such a threshold will depend on the requirements of the application.

4.2.3 Problems in Automating the Cluster Selection Process in Traditional SOMs

In most applications the SOM is used as a visualisation method, and once the map is generated, the clusters are identified visually. Visually identified clusters have certain limitations as identified in section 4.1, but have proven to be sufficient for applications with a high level of human involvement and where high accuracy is not required.

When it is necessary for automating the cluster identification from the SOM any of the above described methods can be used. With the K-means method, the analyst will need to identify the initial seed points and then the nodes with inputs assigned to them need to be considered as seperate data points for the process. The main limitation is that a large amount of distance calculations have to be performed during the processing. The SOM is normally implemented as a two dimensional grid with each internal node having four or six immediate neighbouring nodes (in the case of GSOM it is four). Therefore each node will have four connections with other neighbours which have to be processed to determine the closest seed point. It is an iterative process which terminates when a satisfactory set of clusters is achieved. Hence the method can be time consuming for large maps. A large amount of information about the neighbours need to be stored

throughout the process.

With the Agglomeration method, initially each of the nodes are considered as seperate clusters. For each node, (or cluster), the neighbouring node (cluster) distances have to be measured and the nodes (clusters) with the smallest separation will be merged together as one cluster. This process will also result in a large amount of processing for large SOMs. The processing becomes complex since some nodes (or clusters) may not have immediate neighbouring nodes which have received *hits*. In such a situation the neighbourhood search algorithm has to consider the next level of neighbours and the distances from these neighbours will need to be calculated.

When considering the Divisive method for clustering a SOM, all the used nodes are considered as one cluster at the beginning. The distance calculations with neighbours will have to be made to identify the largest distance where the *break* is to be made. This method again has the same limitations as the K-means method. Considering a SOM as a two dimensional grid as mentioned above, a node will not be separated from the grid by just eliminating one connection. Therefore the same node may have to be processed several times for such separation. This will also need a large amount of information stored for processing.

4.3 Automating the Cluster Selection Process from the GSOM

The methods that can be used for automating the traditional visual cluster identification from feature maps were described in the previous section. The limitations of these methods were also identified and discussed. In this section we propose a novel method for cluster identification which takes advantage of the incrementally generating nature of the GSOM [Ala00c]. The advantage of the new method over the existing methods and techniques are also highlighted in this section.

4.3.1 The Method and it's Advantages

The identification and separation of clusters from the GSOM is performed after the GSOM has been fully generated and stabilised (converged). i.e. after the growing and smoothing phases described in chapter 3. Therefore the cluster identification process that we propose can be considered as an utility available to the data analyst using the GSOM. The analyst can use visualisation to identify the clusters as is traditionally done with the SOM. Another option would be to complement the visualisation with the proposed automated cluster separation method. The proposed method is designed such that, although the actual cluster identification is automated, a high user involvement can be accommodated. Since the main focus in developing the GSOM was data mining applications, it is essential that the data analyst has the freedom to select the *level* of clustering

required. We use the term *level* as a way of referring to the value of threshold for cluster separation. A high value of threshold is considered as a low level of clustering where only the most significantly apart clusters are identified. A low threshold will result in a finer clustering where even the not so obvious (sub) clusters are separated.

The new cluster identification method is recommended for the data mining analyst in the following situations.

1. When it is difficult to visually identify the clusters due to unclear cluster boundaries.

2. When the analyst needs to confirm the visually identifiable boundaries with an automated method.

3. When the analyst is not confident of a suitable level of clustering, it is possible to break the map into clusters starting from the largest distance and progressing to the next largest. Therefore the data set is considered as one cluster at the beginning and gradually broken into segments, providing the analyst with a progressive visualisation of the clustering.

Before describing the method, we will define terms that are required for specification of the algorithm. A pictorial description of the terms are provided after the definitions.

Definition 4.1 : Path of Spread

The GSOM is generated incrementally from an initial network with four nodes. Therefore the node growth will be initiated from the initial nodes and spread outwards. A path of spread (POS) is obtained by joining the nodes generated in a certain *direction* in the chronological order of their generation. Therefore all POS will have their origin at one of the four starting nodes.

Definition 4.2 : Hit-Points

When the GSOM is calibrated using test data, the input test data values get mapped to some nodes of the feature map. There will also be a number of nodes in the map, which do not get mapped by any input. These are nodes which are generated during the growing phase to preserve the *distance* between clusters thus preserving the topology of the input clusters. Therefore such nodes can be called *stepping stones* for spreading the map out for proper representation of the inter cluster distances. Such nodes are not considered as representing any input vectors, and as such will not get mapped by any inputs during the testing phase. The nodes which get mapped with test inputs are called the *hit points.*

Definition 4.3 : Data Skeleton

Once all the POS are identified, it will be seen that some hit-points do not occur on the POS. These points are then linked to the *closest* points in a POS (where the closeness is calculated by the Euclidean distance between weight values of the

out of POS node to the nodes in the neighbouring positions in the map). All the POS joined to the initial four nodes and the external hit-points linked will result in the data skeleton for a given set of data.

Definition 4.4 : Path Segments and Junctions

When external hit-points are connected to the POS, if the point on the POS which is linked is not a hit-point, it will become a junction. The link between two consecutive hit-points, two neighbouring junctions or neighbouring junction, hit-point pair, is called a *path segment.* The value of a path segment is calculated as the Euclidean difference in weight vector value of the end points.

The basic concept of the new method is to consider the *paths of spread* of the network starting from the initial square grid of four nodes. Since the GSOM spreads out by new node generation, the paths of spread define the structure of the data set by following the *paths* along which the GSOM is generated.

A path of spread is identified by joining one of the four starting nodes with the nodes which *grew* new neighbours. Such joining is performed in the *direction* of growth or spread of the map. Always all the paths of spread begin at one of the four initial nodes of the map and will spread away from the initial network. Figure 4.2 shows the initial GSOM consisting of nodes 1 to 4 and three POS

which terminates at nodes A, B and C.

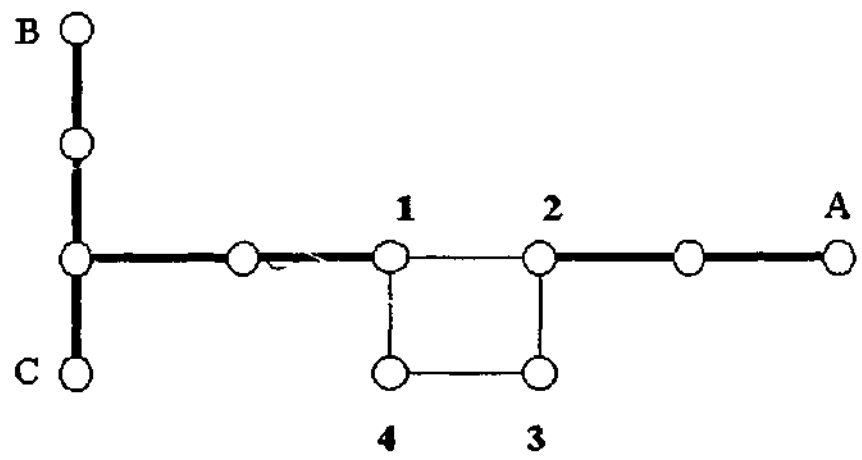


Figure 4.2: Path of spread plotted on the GSOM

As described in the definitions, once the POS are identified, there may be some *hit-points* which are not part of the POS. Such nodes exist due to the the POS identification algorithm which considers a node as part of a POS, only if it generated new nodes (the algorithm is presented later). As such there may be some nodes in the edge (boundary) of the network, which do not initiate new node generation, but still represent input data. Since these nodes also have to be considered while identifying the clusters, we join these nodes to the POS as shown in Figure 4.3. The POS joined by all the remaining *hit points* is defined as the *data skeleton*. A data skeleton is shown in Figure 4.3 where the external hit-point connections to the POS is shown by the broken lines. We propose that the data skeleton diagram represents the input data distribution in a skeletal form. The data skeleton thus generated can be used to identify and seperate the clusters by

a progressive elimination of path segments which will be described in the next section.

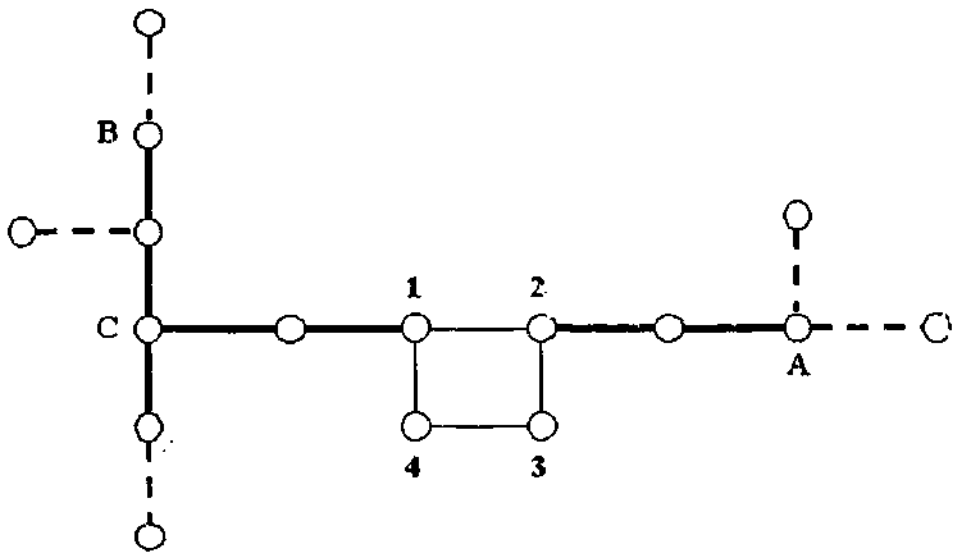


Figure 4.3: Data skeleton

4.3.2 Justification for Data Skeleton Building

The feature map generated by the SOM can be considered as a Voronoi quantiser with individual nodes (represented by their weight vectors) becoming the set of code book vectors representing the input data space (refer section 4.2). The GSOM can be considered as an extended version of the SOM which is incrementally built, and therefore, once fully built can also be described as a Voronoi quantiser for the input data used to generate it. In the previous section we used the incrementally generating nature of the GSOM to identify the paths of spread (POS), which are then used to build the skeleton of the input data. Since the POS creation process made use of the order of the node creation, we need to

identify the sequence of region generation to interpret the POS. An incremental method of Voronoi diagram construction described by Okabe et. al [Oka92] can be used to analyse the POS identification from the GSOM.

Incremental Method of Voronoi Diagram Construction

This method [Oka92] starts with a simple Voronoi diagram for a few points (called generators) and modifies the diagram by adding other generators one by one. For $l = 1, 2, .., n$, let V_l denote the Voronoi diagram for the first l generators $P_1, P_2, \ldots P_l$. The method is used to convert V_{l-1} to V_l for each l. Figures 4.4, 4.5 and 4.6 shows the incremental addition of generators. Figure 4.4 shows the Voronoi diagram V_{l-1}. Figure 4.5 shows the addition of generator p_l to V_{l-1} such that it will become V_l. First, we need to find the generator p_i whose Voronoi region contains p_l, and draw the perpendicular bisector between p_l and p_i. The bisector crosses the boundary of $V(p_i)$ at two points. Let the points be w_1 and w_2 in such a way that p_l is the left of the directed line segment joining the points w_1 and w_2. The line segment w_1w_2 divides the Voronoi polygon $V(p_i)$ into two portions, the one on the left belonging to the Voronoi polygon of p_l. Thus we get a Voronoi edge on the boundary of the Voronoi polygon of p_l.

Starting with the edge w_1w_2, the boundary of the Voronoi polygon of p_l is grown by the following procedure, which is called the *boundary growing procedure*. The bisector between p_i and p_l crosses the boundary of $V(p_i)$ at w_2, entering the ad-

jacent Voronoi polygon, say $V(p_j)$. So the perpendicular bisector of p_l and p_j is drawn next which identifies the point at which the bisector crosses the boundary of $V(p_j)$. This point is shown as w_3 in the Figure 4.5. The rest of the new region as shown in Figure 4.5 is calculated in a similar fashion. Figure 4.6 shows the final V_l Voronoi diagram after the newly added region to V_{l-1}.

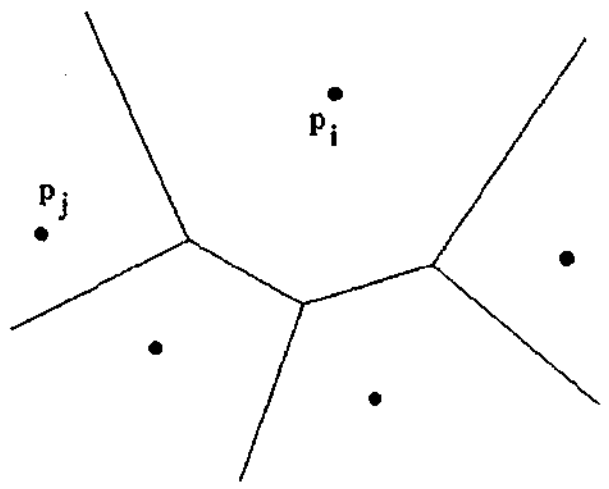


Figure 4.4: Initial Voronoi regions (V_{l-1})

Now in the case of GSOM when a new node is grown, the parent node becomes a non-boundary node and therefore we apply a modified version of the above described boundary growing procedure. In Figure 4.5, we consider point p_l as the parent node and p_i as the newly generated node. Therefore a new finite Voronoi region is assigned to parent p_l since it has now become a non-boundary node. The child p_i represents an infinite region (unbounded region) since it is in the boundary of the network.

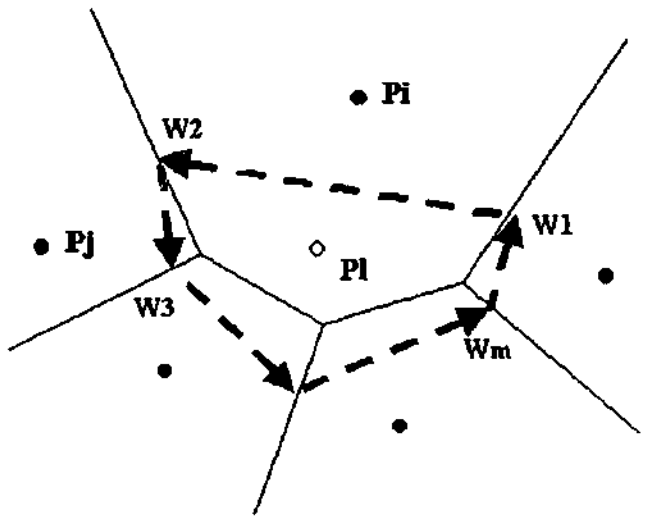


Figure 4.5: Incremental generation of Voronoi regions

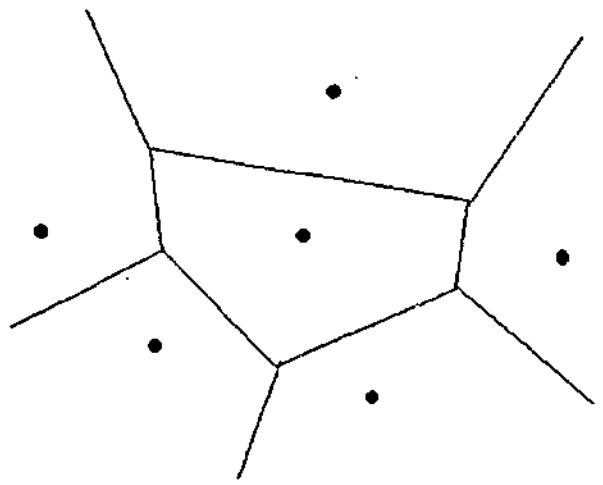


Figure 4.6: Voronoi diagram with the newly added region (V_l)

Consider a GSOM shown in Figure 4.8, and the corresponding Voronoi diagram in Figure 4.7. It can be seen that the GSOM in Figure 4.8 has spread out in one direction (to the right). In Figure 4.7 points A, B, C and D represents the initial four regions (which represent the nodes A, B, C, and D in Figure 4.8) and E, F and G has been identified as the path of spread from point C. The POS is

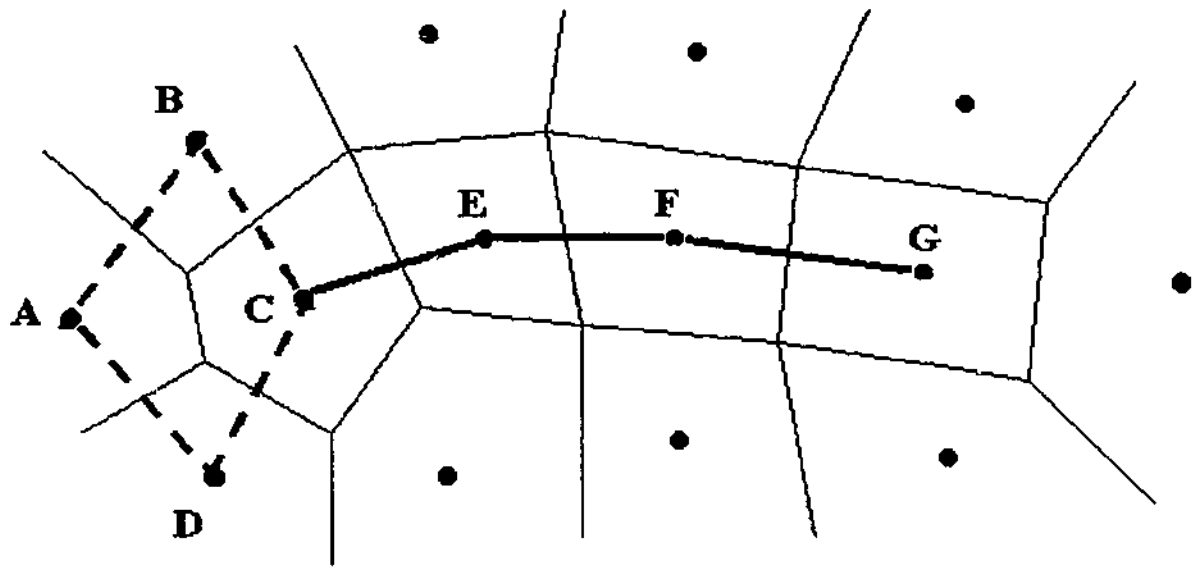


Figure 4.7: Path of spread plotted on the Voronoi regions

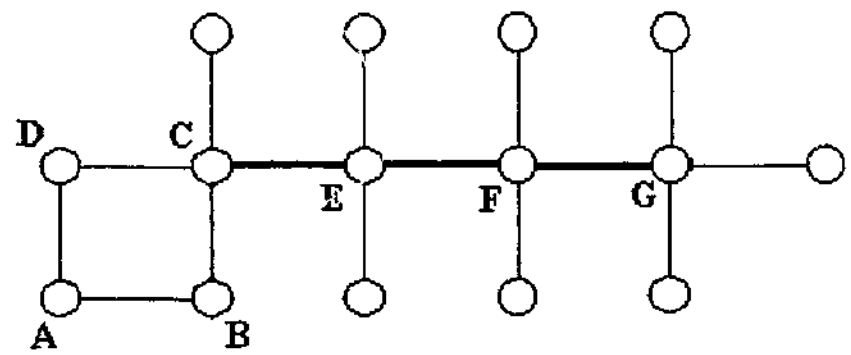


Figure 4.8: The GSOM represented by the Voronoi diagram in Figure 4.7

marked according to the chronological order of nodes becoming parents for new node growth in a particular direction, i.e. from E to G. Therefore points E, F and G represent a set of regions that are generated one after other in incremental region generating method as described above. Since these points represent the nodes in a feature map, their order of generation describes the direction of map growth. Therefore we denote the line joining points E, F and G as a path of

spread (POS) of the feature map as shown in Figure 4.8.

Once the feature map is generated, all the POS are identified using log records generated during the growing phase. Then the hit-points are identified by calibrating the map with a set of test data. It will be seen that some of the nodes along the POS do not get mapped with inputs and thus do not become hit points. These nodes are the *stepping stone* nodes which were described before, which are generated to maintain the distance between the clusters to preserve the topology of the input data. It also can be seen that some of the hit points do not lie on the POS. As described before these points are the boundary of the clusters and as such do not get included as part of the POS, since they do not generate new nodes. Since these points have *attracted* inputs, they represent some part of the input space. Therefore we have to take these points into consideration when selecting the clusters. These points are joined to the *nearest position* of the POS considering them as *branches* of the POS. As defined above, the positions on the POS which are joined to such external hit-points are called junctions.

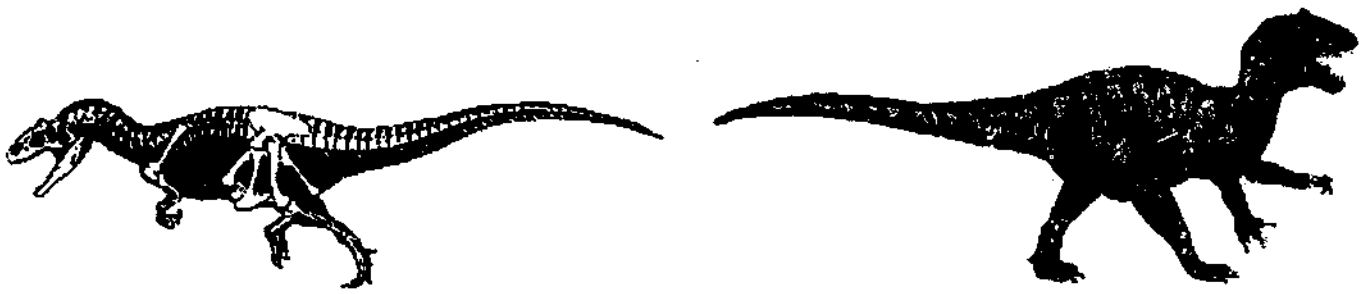
All the POS connected together by the initial four nodes, and connected with the sub branches is called the skeleton of the input data space. As described in section 4.2.1, each point (node represented by a reference vector), can be called a codebook vector which represents the input data values *around that region* (Voronoi region). Therefore, the data skeleton consists of a set of representative vectors

(of the input data), and the path segments along the skeleton represent the inter and intra cluster relationships in the data set. As such the data skeleton can be considered as a *summarised view* of the input data clusters and their relationships. Since the data skeleton consists of the paths (links) joining the clusters, it not only provides a summarised view, but also helps in the visualisation of the spread of the input data structure. Visualisation of the spread may provide a data analyst with additional information from which hypothesis can be built about *unknown* data. The skeleton also provides an insight in to the *order* or sequence of input data presentation to the network for generating the GSOM.

Therefore, we present the concept of the data analysis and hypothesis building ability using the data skeleton as similar to a dinosaur expert hypothesising about the shape and structure of a dinosaur (Figure 4.9(right)) from its skeleton as shown in Figure 4.9(left). Therefore we propose that the structure of the input space can be visualised with the data skeleton built using the POS from the GSOM. The skeleton can then be used to automate the cluster identification process as described in the next section. Further usage of the data skeleton for data mining is described in chapter 6.

4.3.3 Cluster Separation from the Data Skeleton

One of the main uses of feature maps is to identify the clusters in a set of input data. In current usage, cluster identification from feature maps is mainly done



Charging Allosaurus fragilis/atrox, by Gregory Paul (1988)

Illustration from Indianapolis Children's Museum

Figure 4.9: Creating a dinosaur from it's skeleton

using visualisation and feature maps are being called a visualisation tool. We identified the limitations of depending solely on visualisation as a cluster identification method and presented three methods, K-means, divisive and agglomerative as alternative methods.

The main limitation of K-means method is that, the number of clusters K has to be pre-defined. As we highlighted in chapter 2, the main advantage of using the SOM or the GSOM for data mining is the unbiased nature of their unsupervised clusters. Therefore, forcing a pre-defined number of clusters results in a loss of the independence of the feature map. Such pre-defining of clusters will introduce user bias into the clusters thus reducing the value of the feature maps for exploratory data analysis. The divisive and agglomerative methods do not have this limitation of pre-defining the number of clusters. Therefore the data analyst has the option of selecting the level of clustering required according to the needs of the application. In the agglomerative method, the analyst can *watch* the

clusters merging until the appropriate level of clustering is achieved. Similarly, in the divisive method the analyst can decide when to stop the clusters breaking apart. As described in section 4.2.2 the main limitation of these methods is the number of connections that have to be considered to identify the proper clusters.

The proposed method is a hybrid of these three techniques, and the clusters are derived using the data skeleton from a GSOM. Once the data skeleton is built the path segments have to be identified. The *distances* or lengths of the path segments are calculated as the difference between the weight values of the respective junction/ hit-point nodes. The path segment lengths are then listed out in descending order, such that the *weakest* segment (in the sense of closeness of the clusters) is listed on top. The data analyst can then remove the path segments starting from the weakest segment. This process will continue until the analyst decides that there are sufficient or appropriate number of clusters.

Since the method uses the data skeleton, it only has to consider a smaller number of connections (with neighbours) compared with the agglomerative and divisive methods. This will result in faster processing and also quicker separation of clusters since in most cases only a single path has to be broken to seperate the different parts of the skeleton. The method is demonstrated with an artificial and a real data set in section 4.4.

4.3.4 Algorithm for Skeleton building and Cluster Identification

The steps in the algorithm for data skeleton building and cluster separation is given below. The input to this algorithm is a fully generated GSOM.

1. **Skeleton building phase :**

   (a) Plot the *node identification numbers* for each node of the GSOM. The node numbers (1..N) are assigned to the nodes as they are generated. The initial four nodes have the numbers 1..4. Therefore the node numbers represent the chronological order of node generation in the GSOM.

   (b) Join the four initial nodes to represent the *base* of the skeleton.

   (c) Identify the nodes which initiated new node generation (from the log file generated during the growth phase), and link such parent nodes to the respective child nodes. This linking process is carried out in the chronological order of node generation. The node numbers of nodes which initiate growth are stored for this purpose during the growing phase of the GSOM creation.

   The linking process is described with Figure 4.10. Figure 4.10(a) shows the list of node numbers which initiated new node growth. Figures 4.10(b), (c) and (d) show the process of node linking according

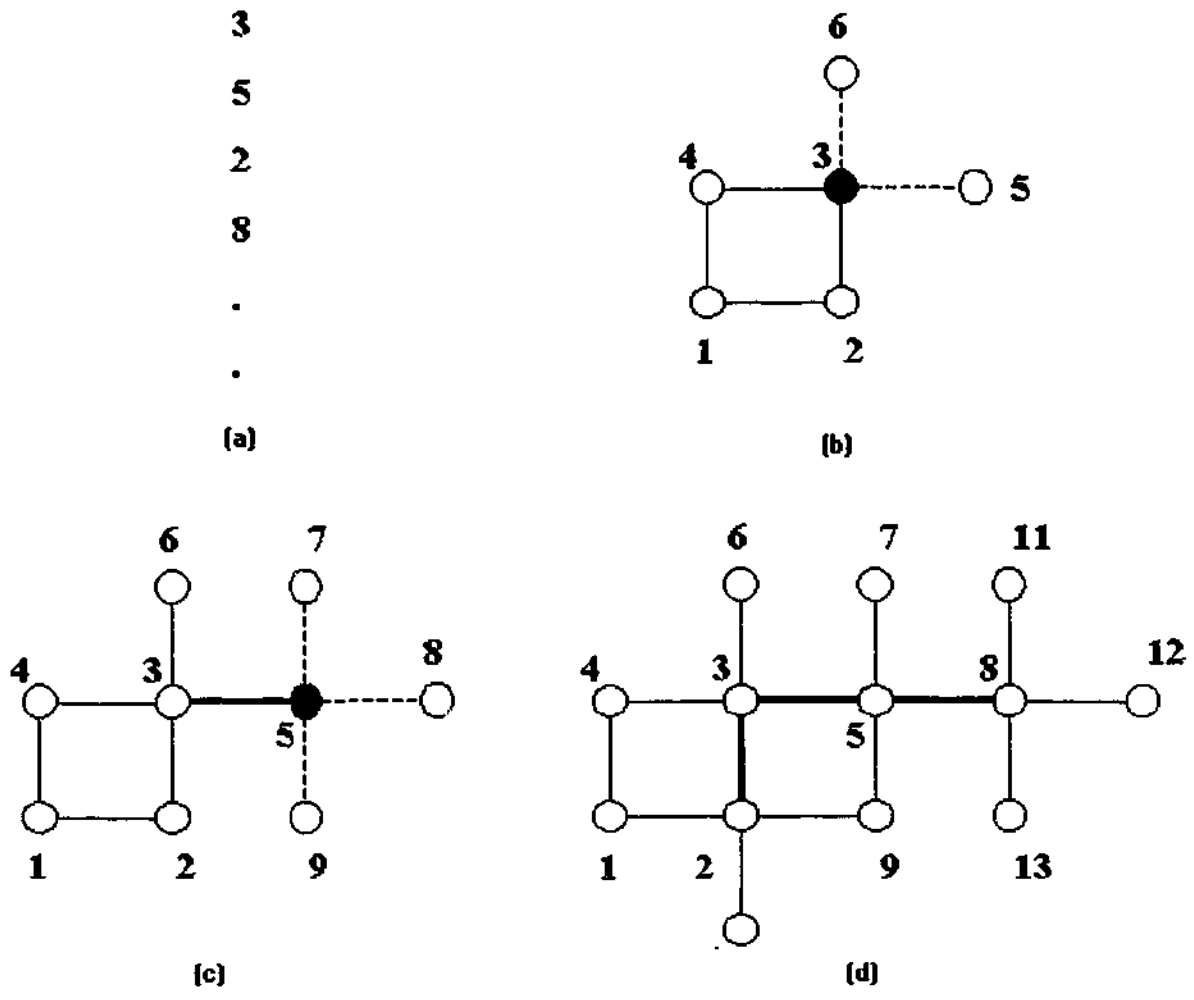


Figure 4.10: Identifying the POS

to the order specified by the list in 4.10(a). Therefore the process can be described as a simulation of the GSOM generation.

(d) Identify the paths of spread (POS) from the links generated in (c).

(e) Identify the *hit-points* on the GSOM.

(f) Complete the data skeleton by joining the used nodes, not on the POS, to the respective POS as sub-branches.

(g) Identify the junctions on the POS.

2. **Cluster separation phase :**

   (a) Identify the path segments by using the POS, hit-points and the junctions.

   (b) Calculate the distance between all neighbouring junctions on the skeleton. Euclidean metric is used as the distance measure which is

$$D_{AB} = \sum_{i=1}^{D} (w_{i,A} - w_{i,B})^2$$

   where A, B are two neighbouring hit-points / junctions and the line joining A, B is called the path segment AB.

   (c) Delete path segments starting from the largest value as :

   find $D_{max} = D_{X,Y}$ such that

$$D_{X,Y} \geq D_{i,j} \quad \forall i, j \in [1..maxnodes]$$

   where X, Y, i, j are node numbers.

   Delete segment XY.

   (d) Repeat (c) until the data analyst is satisfied with the separation of clusters in the GSOM.

The above algorithm results in a separation of clusters using the data skeleton such that the data analyst can observe such separation. Such visualisation of the clusters provide a better opportunity for the analyst to decide on the ideal *level* of clustering for the application.

4.4 Examples of Skeleton Modeling and Cluster Separation

In this section we use three data sets to demonstrate the skeleton building and cluster separation process. The first two experiments demonstrate the process using artificial data sets generated for the purpose of describing different spreads of GSOMs. These input data sets are selected from two dimensional space between the coordinates $(0,0),(0,1),(1,1),(1,0)$. The third experiment uses a realistic data set of 28 animals (selected from the 99 animal data set given in appendix A) to describe the same process.

4.4.1 Experiment 1 : Separating Two Clusters

In this experiment we use a set of data selected from the two dimensional region as shown in figure 4.11. The input data are selected as two clusters from the top right and bottom left corners of the region which are shown as the shaded area in Figure 4.11. Each cluster consists of 25 input points which are uniformly distributed inside the cluster.

Figure 4.12 shows the GSOM generated on this data with a SF of 0.25. The nodes shown as black circles and the shaded nodes represent the nodes that are mapped with inputs when calibrated with the same data. It is easy to see that two clusters are shown separately in the GSOM and could be identified visually.

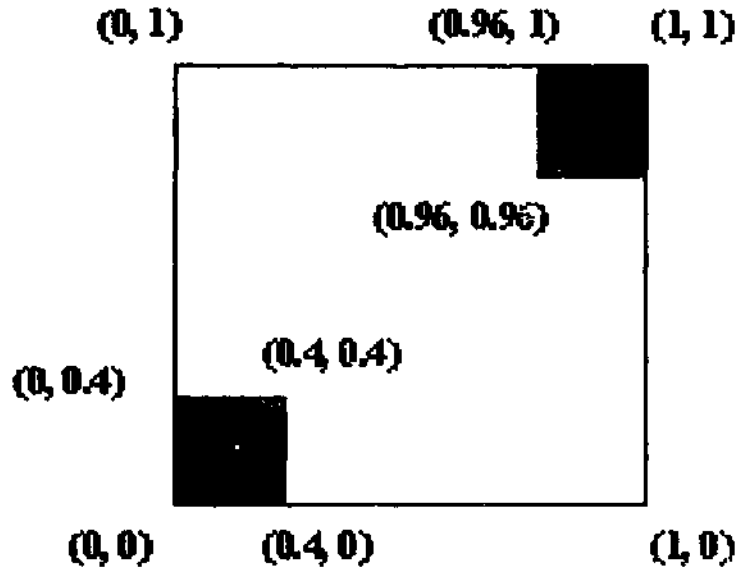


Figure 4.11: The input data set

The purpose of using this simple data set is to demonstrate the cluster separation method, in such a way that the effect of the method could be easily seen.

Figure 4.13 is the data skeleton for the cluster data set in Figure 4.11. The broken lines in the middle of the figure shows the initial (starting) GSOM. The thicker lines show the paths of spread (POS) spreading outwards from the initial four nodes of the GSOM. The thin lines are the connections from the nodes outside the POS, which forms the skeleton. The hit nodes are shown shaded in grey and the nodes coloured in black are the junctions. Path segments are given identification numbers for future reference.

Once the data skeleton has been built the distances for the path segments are

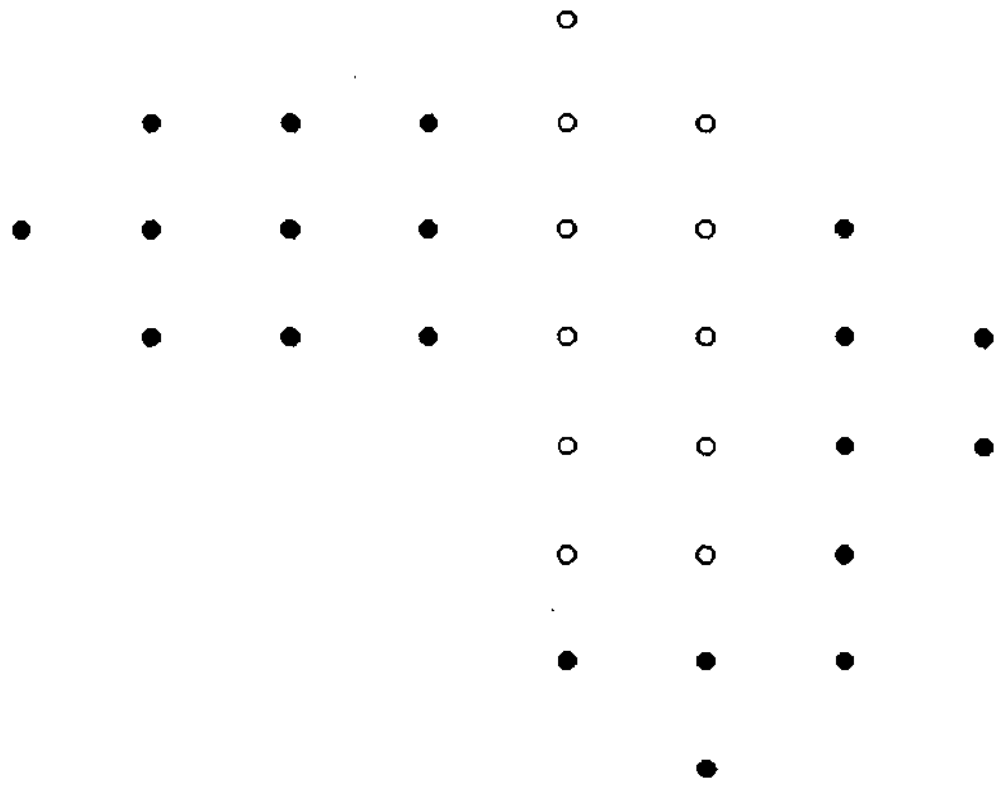


Figure 4.12: The GSOM with the *hit points* shown as black and shaded circles

calculated. Table 4.1 shows the distances calculated for the path segments for the data set in Figure 4.11. The third column in the table provides the value of $D_s - D_{s+1}$, where D_s and D_{s+1} are the distances of two adjacent path segments. This value is provided for comparison by the analyst to identify significant variations in such differences. The distance values in table 4.1 are ordered in descending order for the ease of identifying the significant separations.

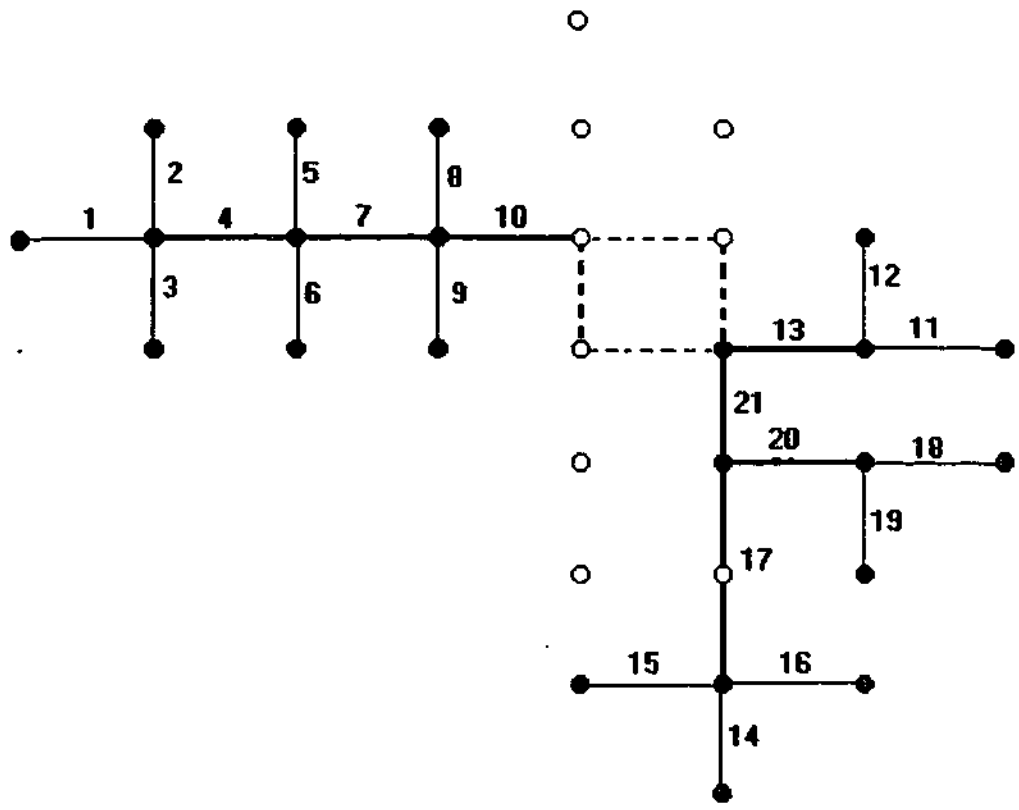


Figure 4.13: Data skeleton for the two cluster data set

It can be seen from table 4.1 that segment 10 contains a large distance value compared to the rest of the values. When this is related to Figure 4.13 it can be seen that the segment 10 *joins* two clusters which corresponds with two clusters in the input data set. The path segment will now be removed starting from the largest value. Figure 4.14 shows the data skeleton with segment 10 being removed. Now there are two clusters which are separated. By observing the table 4.1 values, any further removal of path segments will not produce any significant separations of the data points.

Table 4.1: Path segments of two cluster data set

Segment No.	$\sum_{i=1}^{Dim} (w_{i,A} - w_{i,B})^2$	Difference
10	1.06945733	
13	0.0935717	0.97588563
17	0.03946001	0.05411169
20	0.03246394	0.00699607
21	0.01611298	0.01635096
19	0.00010408	0.0160089
8	0.0000797	0.00002438
12	7.88E-05	9E-07
4	0.0000701	8.7E-06
5	0.00006642	3.68E-06
1	0.00006617	2.5E-07
9	0.0000593	0.00000687
3	0.00005393	0.00000537
6	0.00005378	1.5E-07
7	0.00005365	1.3E-07
14	4.84E-05	0.00000523
2	0.00004514	0.00000328
16	3.33E-05	0.00001182
11	1.77E-05	0.00001559
15	0.00001322	0.00000451
18	1.11E-05	0.00000217

4.4.2 Experiment 2 : Separating Four Clusters

In this experiment, four clusters are selected similar to the the first data set except that these clusters are selected from the corners of the square. The clusters selected are shown in Figure 4.15. This experiment is carried out to demonstrate the spread out of the POS's and the skeleton with a higher number of clusters. Both previous and this experiment attempt to show the representation of the

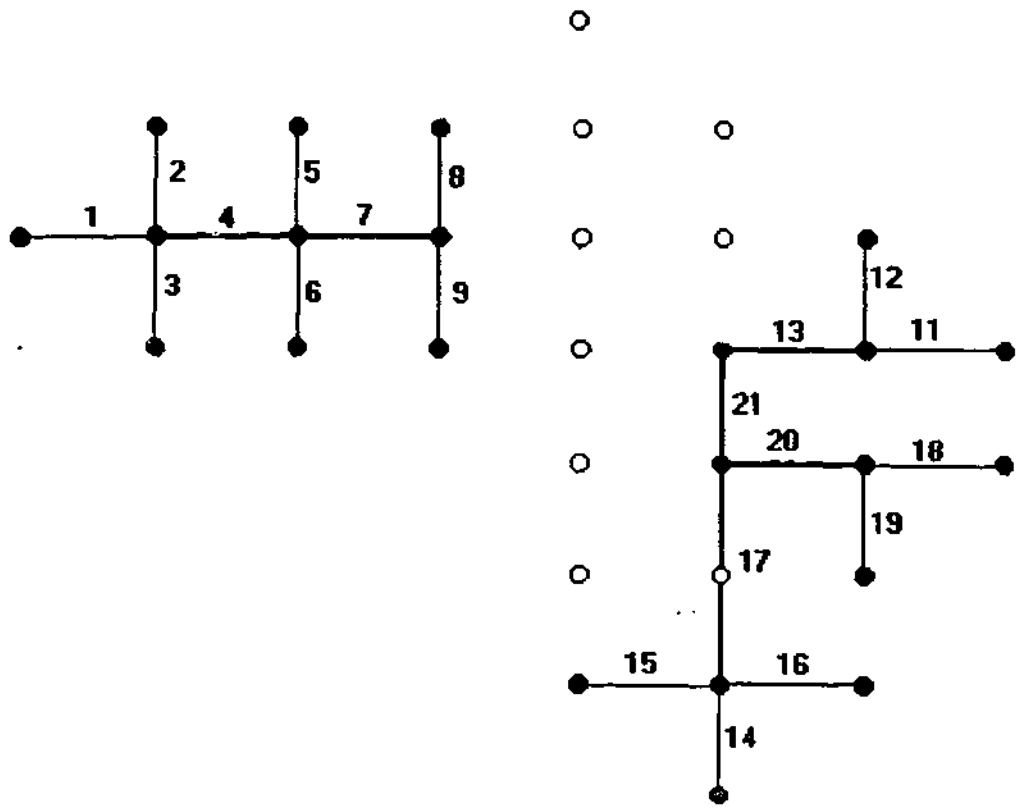


Figure 4.14: Clusters separated by removing segment 10

data set by the skeleton and to emphasize that the data skeleton provides an initial idea of the structure of the input data set.

Figure 4.16 shows the GSOM generated with a SF of 0.05 for the four cluster data set. In Figure 4.16, the hit nodes are shown by shaded areas, similar to two cluster experiment. It can be seen that the GSOM has separated the four clusters and it is possible to easily visualise the cluster separation. Now we can build the data skeletons and use the automated cluster separation method to separate these clusters. Figure 4.17 is the data skeleton for the four clusters with the segments numbered with the same notation used in the first experiment.

Table 4.2 shows the path segments from the skeleton for the four cluster data.

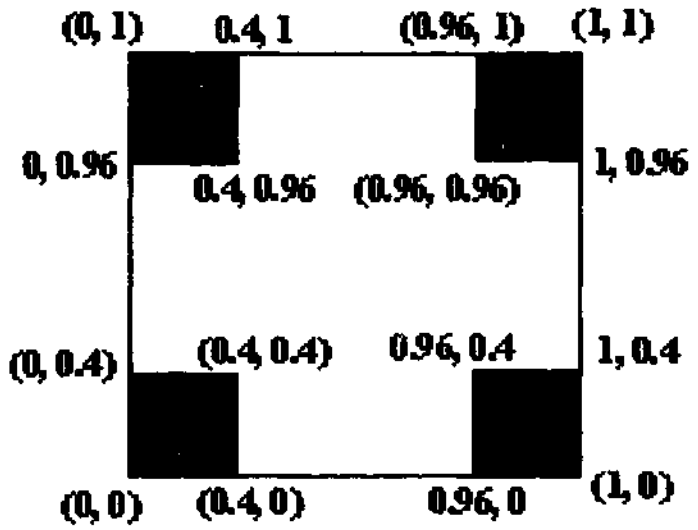


Figure 4.15: The input data set for four clusters

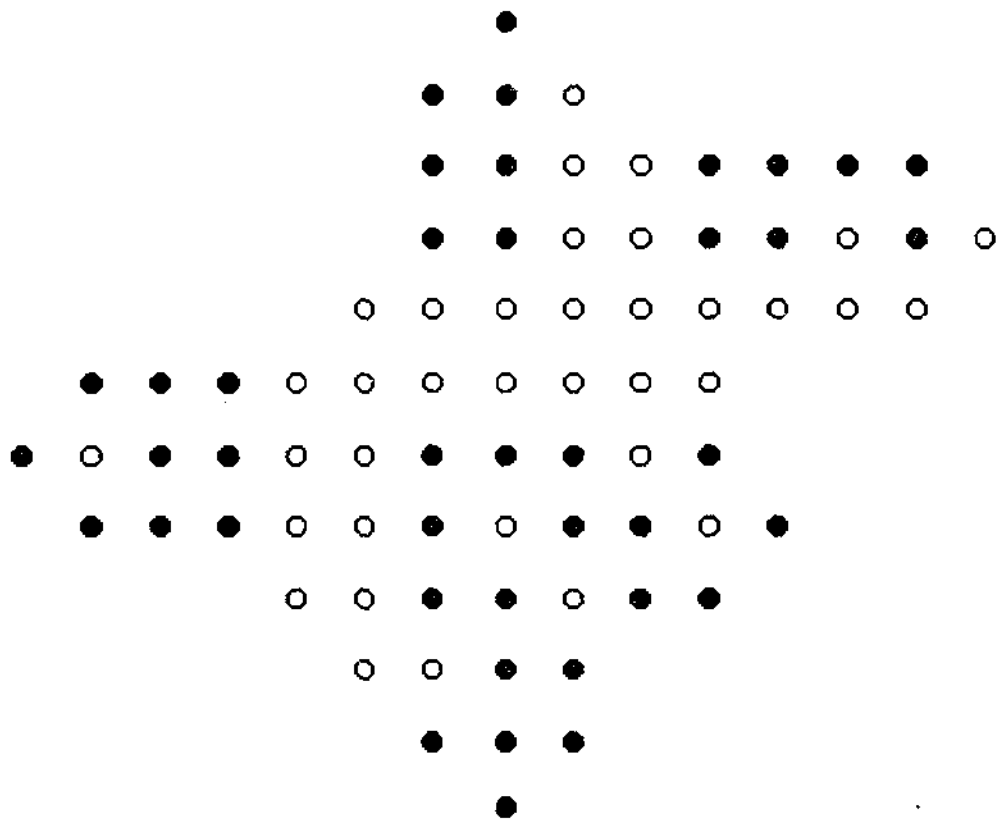


Figure 4.16: The GSOM for four clusters with the hit nodes in black

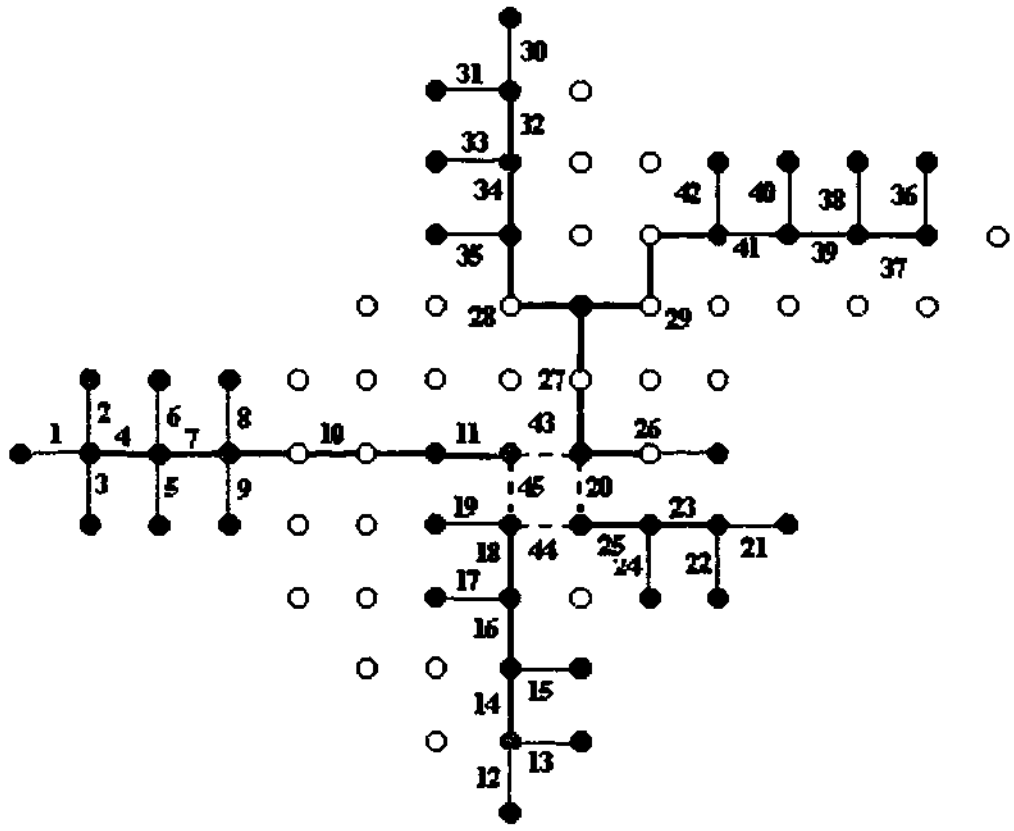


Figure 4.17: Data skeleton for four clusters

From this table, it can be seen that the path segment distances of segments 10, 27, 28, 29 are significantly larger than the rest.

Figure 4.18 shows the skeleton with these segments removed. Now the resulting skeleton clearly shows that the four clusters have been separated in Figure 4.18.

Table 4.2: Path segments of four cluster data set

Segment No.	$\sum_{i=1}^{Dim}(w_{i,A}-w_{i,B})^2$	Difference
10	0.86493025	
28	0.59618458	0.26874567
27	0.35666577	0.23951881
29	0.21922388	0.13744189
34	0.00146005	0.21776383
35	0.00078201	0.00067804
11	0.00036194	0.00042007
36	0.00034612	0.00001582
12	0.00016785	0.00017827
42	0.0001517	0.00001615
26	0.0001465	5.2E-06
38	0.00012517	0.00002133
20	0.0001154	0.00000977
18	0.000106	0.0000094
37	0.00010418	1.82E-06
19	9.95E-05	0.00000472
45	9.95E-05	0
30	9.09E-05	0.00000856
40	8.75E-05	0.00000337
17	8.20E-05	0.00000553
3	0.00008	2E-06
41	7.58E-05	4.23E-06
2	0.00006989	0.00000588
24	6.73E-05	0.00000264
16	6.69E-05	3.6E-07
6	0.00005993	6.96E-06
5	0.00005965	2.8E-07
15	5.69E-05	0.00000275
8	0.00005545	0.00000145
25	5.24E-05	0.00000308
44	5.22E-05	0.00000013
13	5.19E-05	3.6E-07
43	5.12E-05	6.6E-07
9	0.00005045	7.7E-07
39	4.84E-05	0.00000203
4	0.00004685	0.00000157
14	4.47E-05	0.00000211
22	3.25E-05	0.00001224
33	3.04E-05	0.00000206
23	0.00002689	0.00000355
31	2.38E-05	0.00000311
1	0.00002209	0.00000169
21	0.00001237	0.00000972
32	7.80E-133	0.00001237
7	0	7.8049E-133

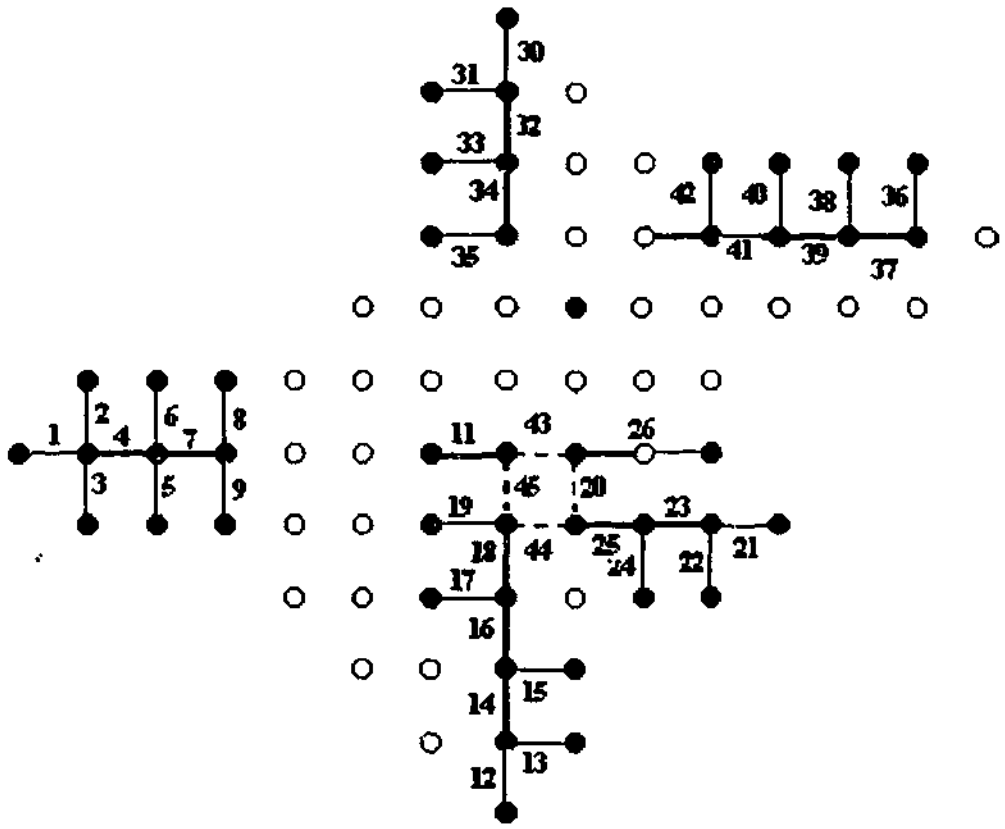


Figure 4.18: Four clusters separated

4.4.3 Skeleton Building and Cluster Separation using a Real Data Set

We use a subset of the animal data set used in chapter 3, to demonstrate the skeleton building and cluster separation process. Only 28 of the 99 animals (from the animal data) are selected based on the following criteria.

1. Generally well known and familiar animals.

2. Belongs to four main groups (insects, fish, mammals and birds).

3. Some sub-groupings exist inside the main groups (for example, meat mammals).

The set of animals selected are, lion, cheetah, wolf, puma, leopard, lynx, a lope, deer, elephant, buffalo, giraffe, seal, dolphin, pike, herring, carp, piranha, bee, wasp, fly, gnat, pheasant, wren, sparrow, lark, dove, duck and penguin.

The initial GSOM generated with the data is shown in Figure 4.19. It can be visualised that the GSOM has spread out mainly in four directions which correspond to the four *types* of animals in the data set. Figure 4.20 shows the data

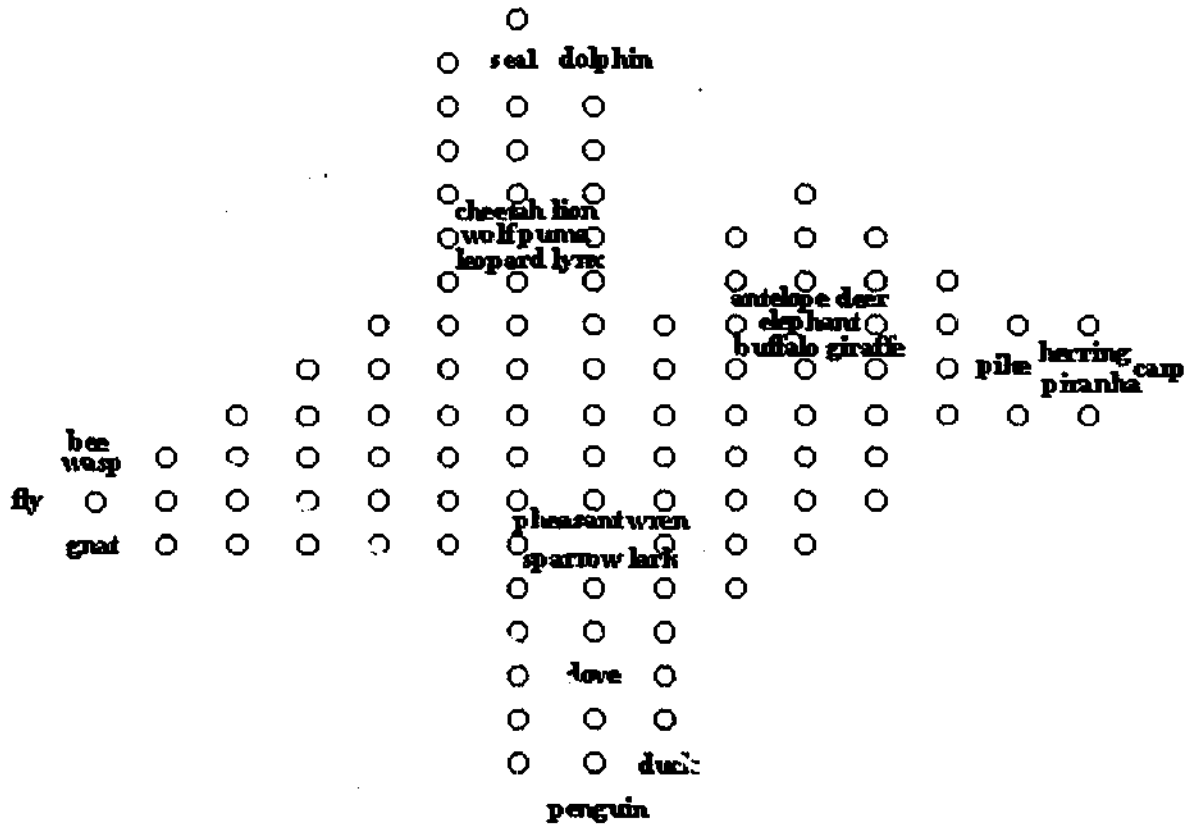


Figure 4.19: The GSOM for the 28 animals, with SF=0.25

skeleton mapped on to the GSOM and the POS's spreading outwards from the

initial square map. It can be seen from the skeleton that the main groups have spread out in four *directions* in the map. In fact, we can see that the GSOM has been generated by spreading out in these directions. It is interesting to see that the non-meat eating mammals have been mapped onto a different POS from the meat eaters, but it can also be seen that both POS's *move* in the same direction. It can also be identified that the sub-groups inside the main groups have been mapped to the same POS in most cases. Table 4.3 shows the distances for the path segments identified in the figure 4.20.

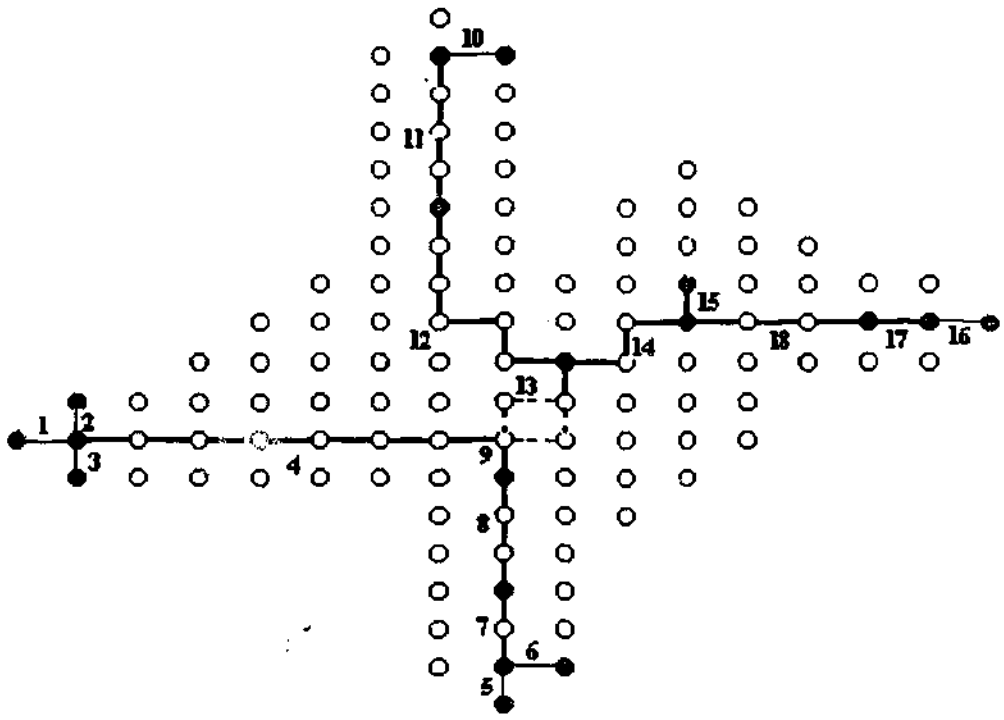


Figure 4.20: Data Skeleton for the animal data

Table 4.3 shows the distances for the path segments identified in the figure 4.20. The data analyst can now remove segments starting from the top of the table until the clusters formed is satisfactory or meaningful (which will depend on the ap-

plication at hand). It can be seen that segment 18 has a large distance compared to the others. Since the GSOM produces a topology preserving mapping this can be interpreted as the fish being considered as the most different cluster (group) from the other animals. The insects (wasp, bee, gnat and fly) are separated next by removing segment 4. Segments 13 can be removed next and this separates the four main groups. Now further segment removal will separate the sub clusters inside the main clusters. Figure 4.21 shows the main groups separated in the animal data set.

Table 4.3: Path segments for animal data set

Segment No.	$\sum_{i=1}^{Dim} (w_{i,A} - w_{i,B})^2$	Difference
18	7.43525391	
4	4.31007765	3.12517626
13	3.76652902	0.54354863
11	2.76706805	0.99946097
7	1.56629396	1.20077409
12	1.32260106	0.2436929
8	0.7473245	0.57527656
14	0.49138419	0.25594031
5	0.27485321	0.21653098
3	0.19083773	0.08401548
10	0.15102813	0.0398096
16	0.14247555	0.00855258
2	0.12936505	0.0131105
6	0.08565904	0.04370601
17	0.07553993	0.01011911
1	0.03097862	0.04456131
9	3.45E-06	0.03097517
15	0.00000001	0.00000344

Therefore it can be seen from these experiments that the segment removal method separates the clusters and provides the analyst with independence in selecting (or deciding) the level of clustering.

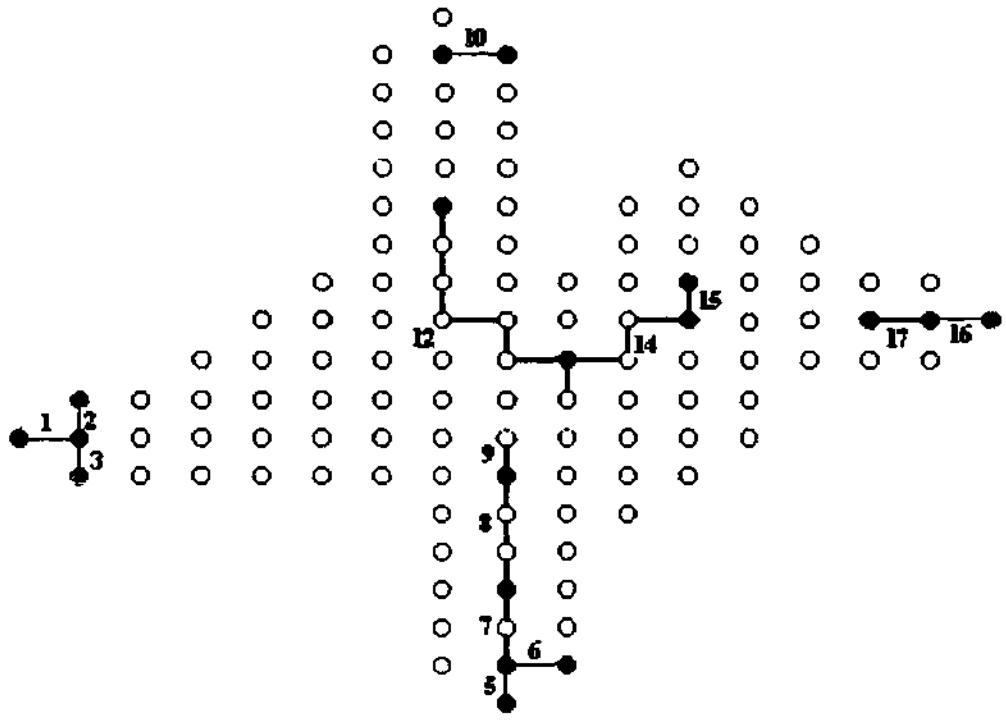


Figure 4.21: Clusters separated in the animal data

4.5 Summary

In this chapter a novel method of cluster identification from the GSOM was described. The motivation for developing this method is to enhance the traditional usage of feature maps from just visualisation tools for data mining to more automated method of cluster identification. Since this method made use of the *growing* nature of the GSOM to build the data skeleton, we can also present this

method as a *value added* over traditional SOM's.

The other advantage of this method are

1. The ability of the data analyst to be involved in the cluster separation even though the process is automated. The analyst can decide when to stop the path segment removal depending on the level of clustering required. On the other hand the analyst can let the segment removing process run from start to end, which will provide an *incremental picture* of the level of clustering present in the data.

2. It has been proved that the SOM does not provide complete topology preservation. Therefore the visual separation of clusters may provide a distorted view of the actual *difference* in the clusters. With the proposed method the actual differences in weights are also considered, which combined with the visual separation provide better understanding of the clusters.

3. The data skeleton provides the foundation for building a conceptual model for the input data set. Such a model can then be used for mining of rules and monitoring for changes. Chapter 6 describes such a model building using the data skeleton.

Therefore we conclude that this chapter has described a method of building a structure called the data skeleton and proposed a method of automated cluster identification using the GSOM. These methods enhance the suitability of the

GSOM as a data mining tool compared to the SOM, which is currently used mainly as a visualisation technique. In the next chapter we present a further usage of the GSOM by manipulating the spread factor as a control measure for hierarchical clustering.

Chapter 5

Optimising GSOM Growth and Hierarchical Clustering

5.1 Introduction

The previous chapter described a novel method of identifying clusters from the GSOM. Initially a data skeleton is created from the GSOM and the skeleton is separated according to the clusters by removing the path segments. The *growing* nature of the GSOM make it possible to identify the paths of spread, which are then used to build the data skeleton. Therefore the data skeleton building is *value added* to the feature maps due to its dynamic generation compared to conventional SOM.

A indicator called the spread factor (SF) is derived in chapter 3, and has been

proposed as a parameter which provides the data analyst with control over the spread of the GSOM. The ability to control the spread (size) of the feature map is unique to the GSOM and could be manipulated by the data analyst to obtain a progressive clustering of a data set at different levels. The SF can take real values in the range of 0..1 and is independent of the dimensionality of the data. For example, if the map generated at $SF = 0.3$ on a 5 dimensional data and another map with $SF = 0.3$ on a 10 dimensional data has *similar levels of spread*, then this provides the analyst with the possibility of visualising groupings across data sets for comparison. The spread of the GSOM can be increased by using a higher SF value. Such spreading out can continue until the analyst is satisfied with the *level* of clustering achieved or in the extreme case, each node is identified as a separate cluster.

For a data analyst using the SOM, the only option available for achieving such a spreading out effect is by increasing the network size (i.e. increasing the length and width parameters of the map). In this chapter we highlight the difference between increasing the size of a SOM and the spreading out a GSOM using high SF values. The GSOM using the SF indicator is described as a method for representing more accurate inter and intra cluster relationships, while the SOM can provide a distorted map unless very accurate *knowledge* about the data pre-exist.

The use of the SF indicator also facilitates the hierarchical analysis of data. Such

a hierarchy will consist of a *small* GSOM with a low spread factor at the top of the hierarchy and gradually larger GSOMs with higher SFs at the lower levels. Therefore the total data set is mapped at the top of the hierarchy and only the *interesting* clusters can be further spread out at the lower levels thus providing a divide and conquer method for data analysis. Such hierarchical clustering of data and separately analysing interesting clusters is advantages in real life data mining applications when the volume of data is one of the critical problems.

The focus of this chapter is the discussion of the usefulness of having a SF indicator for controlling the GSOM, and the opportunities and advantages it provides, specially with regard to data mining applications. In section 5.2 we describe the spread factor in detail and it's usage as a control measure of the GSOM. Section 5.3 discusses the difference between the size increase in traditional SOMs and its limitations compared with the spreading out of the GSOMs with increasing SF values. Section 5.4 describes the hierarchical feature map generation using the different values of SF. Section 5.5 provides experimental results and section 5.6 summarise the contributions of this chapter.

5.2 Spread Factor as a Control Measure for Optimising the GSOM

The notion of spread factor (SF) in GSOM was introduced and defined in chapter 3. In this section we analyse the usage of SF in detail and discusses the advantages and opportunities presented to a data mining analyst using the GSOM with varying spread factor.

5.2.1 Controlling the Spread of the GSOM

The traditional SOM does not provide any measure for identifying the size of the map with regard to data set or amount of spread required. Therefore the data analyst using the SOM for data mining has to use previous experience to create the initial grid by defining the size as rows and columns of nodes. Once the feature map has been generated, if the spread of clusters achieved is unsatisfactory, a larger grid has to be initialised and re-trained [Ses94]. Self generating feature maps has been suggested as a solution to this problem. Several previous attempts at developing such dynamic feature maps are described in chapter 2. Although these models claimed *better* cluster identification, none had a feature such as the SF to control or indicate the *amount of spread* of a feature map.

The identification of a cluster will always depend on the data set being used and also the *need* of the application. For example, in certain instances, the analyst

may only be interested in identifying the *most significant* clusters or groupings. Alternatively, the analyst may become interested in a more detailed separation of clusters. Therefore the clusters identified will depend on the *level of significance* required by the analyst at a certain instance in time. Such a level of significance is even more apparent in data mining applications. Since the data analyst is not aware of the clusters in the data, it is generally required to study the clusters obtained at several levels of spread (or detail) to obtain an understanding of the data. Attempting to obtain such different sized maps with the SOM can result in a distorted view since we attempt to force a data set into a map with a pre-defined fixed structure as discussed in chapter 2. The previous dynamic SOMs attempt to solve limitations of the fixed grid structure but do not address the problem of identifying a level of significance for cluster identification. In other words, they do not recognise the need and advantages of identifying clusters at different levels of spread. Thus these previous models assume a fixed number of clusters and usually define a threshold for cluster separation. Cluster analysis using different threshold values do not provide the flexibility of using the spread factor, as discussed below.

As mentioned in the earlier chapters, the GSOM was specifically developed for requirements of a data mining analyst. Therefore we not only consider the different *shapes* of structures in the data sets, but also consider the number of clusters as a variable which depends on the *level of significance* required by the data analyst.

The spread factor is our solution to this issue, which provides the analyst with a measure for controlling the map spread as required.

5.2.2 The Spread Factor

The derivation of the formula for the SF is also presented in section 3. The spread factor can be described as an indicator of the spread of a GSOM which has the following characteristics.

1. it takes real value in the range 0 to 1.

2. it has to be provided as a parameter to the GSOM at the start of training.

3. it is used to calculate the *growth threshold* (GT) according to the formula

$$GT = -D \times \ln(SF)$$

where D is the dimensionality of the data.

As described in chapter 3, the error of a certain node has to exceed the growth threshold for new nodes to be generated in the GSOM. A high GT will allow the nodes to accommodate large error values thus producing a *smaller* map and vice versa. The error accumulated in the nodes is dependent on the dimensionality of the input data, according to the formula

$$TE_i = \sum_{H_i} \sum_{j=1}^{D} (x_{i,j} - w_j)^2$$

where $x_{i,j}$ is the j^{th} attribute of the i^{th} input, w_j is the j^{th} attribute of the corresponding weight vector of the winning node, H_i is the number of hits (number

of times the node is a winner) of node i, as discussed in chapter 3. The data analyst needs to examine maps of varying spread on a data set has to consider the dimensionality of the data when calculating the GT. The SF is introduced as a solution whereby the values of SF, $0 \leq SF \leq 1$, is used to calculate the required GT using the relevant dimensions of the data. The analyst can refer to the *amount of spread* of a GSOM by it's SF value, independent of the dimensionality of the data. Such ability becomes very useful when comparing data sets with different dimensionality or cluster analysis using subsets of the same data only on selected attributes. In such instances the comparison of the maps with different dimensionality can be justified if they have been generated using the same SF.

Therefore the main purpose of the spread factor is to serve as an indicator for the amount of spread of the GSOM. In this context we define *spread* a· quite different from *growth* in the GSOM. The growth refers to the addition of new nodes to represent *new regions* of the input data space. Spread refers to a *zooming up* effect on the existing map thus making any sub-groupings *appear* within the main clusters. With the traditional SOM, the data analyst has to change the size of the network (grid) to achieve a different spread of clusters. The spread achieved by changing the network size differs from the spread of map by increasing the SF as discussed in the next section.

5.3 Changing the Grid Size in SOM vs Changing the Spread Factor in GSOM

In the previous section we have highlighted the need for a data analyst to train maps of different sizes before arriving at a satisfactory level of clustering. It is also mentioned in chapter 2 that with the SOM this is achieved by changing the size of the initial grid. Since the final map has to be a two dimension, the SOM is always initialised as a square or rectangle with the *length* and *width* defined by the analyst. With the GSOM a similar effect is achieved by generating the map with different SF's. In this section we compare the two methods of increasing the size of SOM and GSOM to demonstrate the advantages of the GSOM controlled by the spread factor.

5.3.1 Changing Size and Shape of the SOM for Better Clustering

Figure 5.1 shows the diagram of a SOM with four clusters A, B, C and D which can be used to explain the spread of clusters due to the change of grid size in a SOM. As shown in Figure 5.1(a) the SOM has a grid of length and width X and Y respectively. The intra cluster distances are x and y as shown in Figure 5.1(a). In Figure 5.1(b) a SOM has been generated on the same data but the length of the grid has been increased (to $Y' > Y$) while the width has been maintained at the previous value ($X = X'$). The intra cluster distances in Figure 5.1(b) are

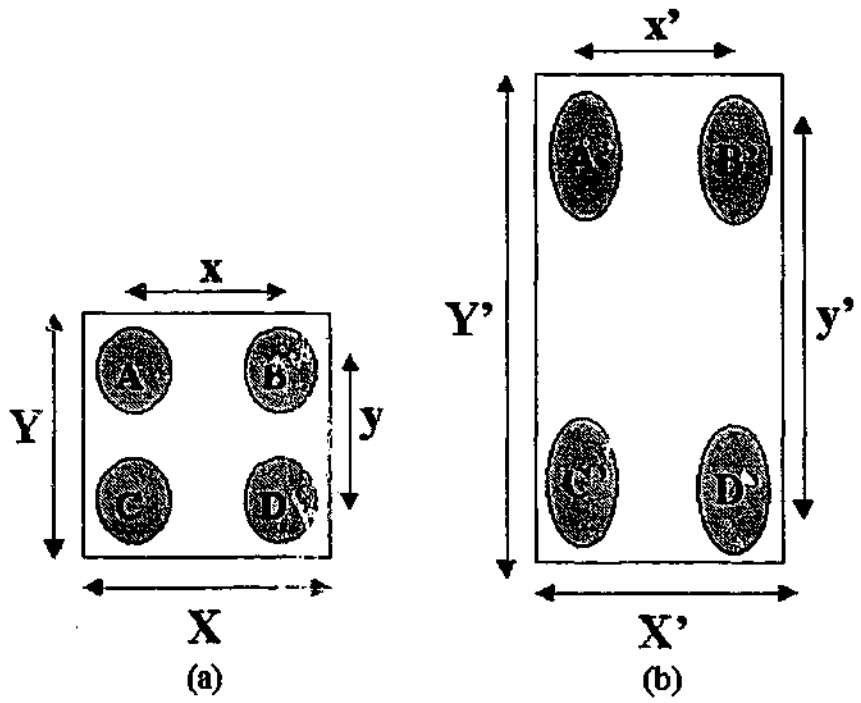


Figure 5.1: The shift of the clusters on a feature map due to the shape and size x' and y'. It can be seen that inter cluster distances in y direction has changed in such a way that the cluster positions have been *forced* into maintaining the proportions of the SOM grid. The clusters themselves have been *dragged out* in y direction due to the intra cluster distances also being forced by the grid. Therefore in Figure 5.1 $X : Y \approx x : y$ and $X' : Y' \approx x' : y'$. This phenomena can be considered in an intuitive manner as follows [Koh95]:

Since we consider two dimensional maps, the inter and intra cluster distances in the map can be separately identified in the X and Y directions. We simply visualise the spreading out effect of the SOM as the inter and intra cluster distances in the X and Y directions proportionally being adjusted to fit in with the width

and length of the SOM .

The same effect has been described by Kohonen [Koh95] as a limitation of the SOM called the *oblique orientation*. This limitation has been observed and demonstrated experimentally with a two dimensional grid by Kohonen [Koh95], and we use the same experiment to indicate the unsuitability of the SOM for data mining.

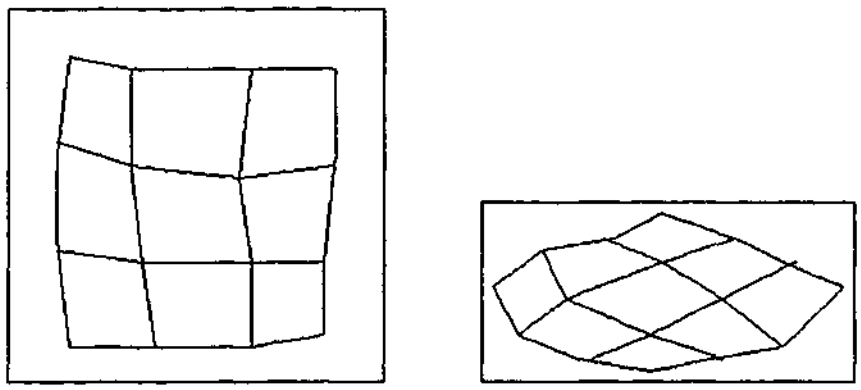


Figure 5.2: Oblique orientation of a SOM

Figure 5.2(a) shows a 4×4 SOM for a set of artificial data selected from a uniformly distributed 2 dimensional square region. The attribute values x, y in the data are selected such that $x : y = 4 : 4$ and as such the grid in Figure 5.2(a) is well spread out providing an optimal map. In figure 5.2(b) the input value attributes $x : y \neq 4 : 4$ while the input data *demands* a grid of $4 : 4$ or similar proportions. As such it has resulted in a distorted map with a crushed effect. Kohonen has described oblique orientation as resulting from significant differ-

ences in variance of the components (attributes) of the input data. Therefore the grid size of the feature map has to be initialised to match the values of the data attributes or dimensions to obtain a *properly* spread out map. For example, consider a two dimensional data set where the attribute values have the proportion $x : y$. In such an instance a two dimensional grid can be initialised with $n \times m$ nodes where $n : m = x : y$. Such a feature map will produce an optimal spread of clusters maintaining the proportionality in the data. But in many data mining applications the data analyst is not aware of the data attribute proportions. Also the data mostly is of very high dimensions, and as such it becomes impossible to decide a suitable two dimensional grid structure and shape. Therefore initialing with an optimal grid for SOM becomes a non-feasible solution.

Kohonen has suggested a solution to this problem by introducing *adaptive tensorial weights* in calculating the distance for identifying the winning nodes in the SOM during training. The formula for distance calculation is

$$d^2[x(t), w_i(t)] = \sum_{j=1}^{N} \psi_{i,j}^2 [\xi_j(t) - \mu_{i,j}(t)]^2 \tag{5.1}$$

where ξ_j are the attributes (dimensions) of input x, the $\mu_{i,j}$ are the attributes of w_i and $\psi_{i,j}$ is the weight of the j^{th} attribute associated with node i respectively. The values of $W_{i,j}$ are estimated recursively during the unsupervised learning process [Koh95]. The resulting adjustment has been demonstrated using artificial data sets in Figure 5.3.

The variance of the input data along the vertical dimension (attribute) versus

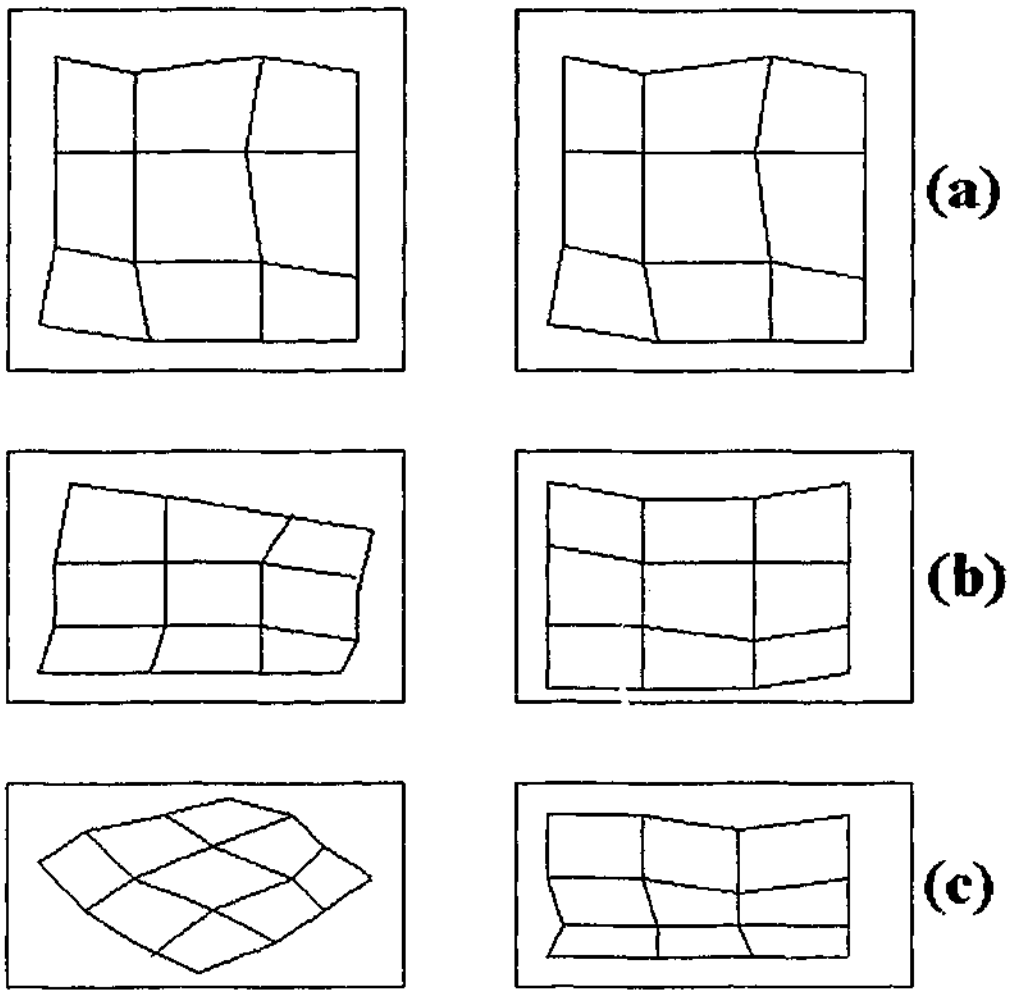


Figure 5.3: Solving Oblique orientation with tensorial weights (from [Koh95])

the horizontal one is varied ($1:1, 1:2, 1:3$, and $1:4$ in Figure 5.3(a), 5.3(b), 5.3(c), 5.3(d) respectively). The results of the unweighted map is shown on the left and the weighted map on the right in Figure 5.3. It can be seen that the optimal grid size for the map for Figure 5.3(d) would have been 2×8. Therefore, the pre-selected grid size results in a distortion of the map, and that the tensorial

weights method *adjusts* the map such that it is forced into the grid proportions.

We interpret the *oblique orientation* as an occurrence due to the map attempting to fit in with a pre-defined network, and resulting in a distorted structure. Tensorial weights method attempts to reduce the oblique orientation but still keeping within the network *borders* thus forcing the shape of the network on the data. This is opposite to the ideal solution since it is the data which should dictate the size and shape of the grid. By changing the size of the grid in the SOM, the map is forced to fit in with a new network size and shape. If the data attributes are not proportionate (in the x and y directions) to the network grid, a distorted final map can occur. Since the SOM is generally used as a visualisation tool in data mining, the inter and intra cluster distances will be shown distorted by the network size and shape. Such a distorted view can be a major disadvantage in data mining applications where the analyst is attempting to obtain an initial idea about the data structure and distribution using visualisation.

5.3.2 Controlling the Spread of a GSOM with the Spread Factor

In GSOM, the map is spread out by using different SF values. According to the formula 3.28 presented in chapter 3, a *low* SF value will result in a *higher* growth threshold (GT). In such a case a node will accommodate a higher error value

before it initiates a growth. Therefore we can state the spreading out effect (or new node generation) of GSOM as follows :

The criteria for new node generation from node i in the GSOM is

$$E_{i,tot} \geq GT \tag{5.2}$$

where $E_{i,tot}$ is the total accumulated error of node i and GT is the growth threshold. The $E_{i,tot}$ is expressed as (equation 3.20)

$$E_{i,tot} = \sum_{H_i} \sum_{j=1}^{D} (x_j(t) - w_j(t))^2 \tag{5.3}$$

If we denote a low SF and high SF values by SF_{low} and SF_{high} respectively, and $\Longrightarrow$ denotes *implies*, then

$$SF_{low} \Longrightarrow GT_{high} \tag{5.4}$$

$$SF_{high} \Longrightarrow GT_{low} \tag{5.5}$$

Therefore from equations 5.3, 5.4 and 5.5 we can say that:

when the $SF = SF_{low}$, node i will generate new nodes when

$$E_{i,tot} = \underbrace{\sum_{H_i} \sum_{j=1}^{D} (x_j(t) - w_j(t))^2}_{R_l} \geq GT_{high} \tag{5.6}$$

Similarly, when the $SF = SF_{high}$, node i will generate new nodes when

$$E_{i,tot} = \underbrace{\sum_{H_i} \sum_{k=1}^{D} (x_k(t) - w_k(t))^2}_{R_s} \geq GT_{low} \tag{5.7}$$

where $x_j \in R_l$ and $x_k \in R_s$ are two regions in the input data space.

It can be seen that region R_l represents a larger number of hits and accommodates a larger variance in the input space. Similarly region R_s represents a smaller number of hits and a smaller variance. Thus R_l represents a larger portion of the input space and R_s represents a smaller portion. Therefore we can infer that in the case of a low SF value, node i represents a larger region of the input space and with a high SF value, node i represents a smaller region. By generalising i to be any node in the GSOM it can be concluded that, with a small SF value, the nodes in the GSOM represents larger portions of the input space and with a high SF value, the nodes represent smaller portions. Therefore using the same input data, a low SF value will produce a smaller representative map and a high SF value will produce a larger representative map.

From the above discussion, the SF value dictates the amount of input to be represented by a node in a GSOM. Now if we visualise the map in x, y directions, similar to that of SOM, it can be seen that the same growth criteria, dictated by the SF applies to both directions. In other words, a node would be grown in a certain direction only if there are sufficient inputs in that *direction* considering the SF value.

In the case of the SOM, the spread of the map is pre-determined by the data

analyst and the input data in a certain direction is forced to fit into the available number of nodes. As such, unless the analyst has a method of (or knowledge of) assessing the *proper* number of nodes, a distorted map can occur. With the GSOM, the input values dictate the number of nodes and the spread factor provides, a global control of the spread of the nodes. Therefore the GSOM does not result in the oblique orientation or distorted view of the data as in the case of SOM.

5.3.3 The Use of the Spread Factor for Data Analysis

As described above, the ability to define a spread factor provides the data analyst with control over the spread of the GSOM. We claim the following as advantages due to the spread factor for data analysis using the GSOM.

1. The spread factor can be used to generate several maps with *increasing* SF values. The analyst can thus observe the clusters in the map and the effect of increasing value of SF on the clusters. Such a *set of maps* will provide the analyst with an initial, unbiased view of the distribution of input data. For example, it will be possible to see whether the data is uniformly distributed or clustered or whether the clusters are of uniform density or non-uniform. The series of maps with increasing SF values provide a more informative picture than observing a single map. The visualisation is also more accurate than attempting to visualise several SOMs with increasing grid sizes since,

as shown in section 5.2, the SOMs will provide a distorted picture if the grid size and shape does not match the *shape* of the data distribution.

2. Once clusters are identified and one or more GSOMs have been selected as *suitably* clustered by the analyst, clusters can further be spread out to identify the sub-groupings. When the data analyst is interested only in a subset of the data, then such relevant clusters can be selected for further spreading out. Such cluster identification and analysis can be carried out hierarchically and will be discussed in detail in section 5.4.

3. It is sometimes useful for a data analyst to observe the clusters in the data using only a partial set of the attributes (clustering on partial dimensions). The SF is independent of the dimensionality of the data and as such can be used to generate maps which can be compared if generated using the same SF. Such analysis will be useful in identifying non-contributing attributes (to the clustering) or attributes which dominate in a cluster. The usefulness of such analysis for data mining will be discussed in chapter 6.

5.4 Hierarchical Clustering of the GSOM

In chapter 4 we presented a method for automating the cluster identification process from the GSOM. Therefore the data analyst has the ability to use either visualisation or the data skeleton to identify the clusters. In section 5.2 we described the concept of the spread factor in detail and introduced the value added

to the GSOM with the ability to generate higher or lower spread maps, as required by the analyst. So far we have been studying maps generated on a certain fixed SF value and as such at a single *level* of clustering. This section highlights the usefulness of clustering of the same data set on several SF values and thus creating a hierarchy of GSOMs and sub-GSOMs [Ala00b].

In section 5.4.1 hierarchical clustering is defined and the advantages of such cluster identification is discussed. Section 5.4.2 describes how hierarchical clustering can be achieved using the spread factor as a control measure, and section 5.4.3 presents the algorithm for hierarchical clustering of GSOMs.

5.4.1 The Advantages and Need for Hierarchical Clustering

A hierarchy is a set $S_H = S_h : h \in H$ of subsets $S_h \subseteq I, h \in H$ called clusters and satisfy the following conditions.

1. $I \in S_H$

2. For any $S_1, S_2 \in S_H$, either they are non-overlapping ($S_1 \cap S_2 = \emptyset$) or one of them includes the other ($S_1 \subseteq S_2$ or $S_2 \subseteq S_1$), all of which can be expressed as $S_1 \cap S_2 \in \emptyset, S_1, S_2$.

Using the definition of a hierarchy given above we can present the GSOM generated on the given data set as the root of the hierarchy. Once the clusters have

been identified further spreading out of these clusters can build the *hierarchical tree* of GSOMs. Such hierarchical clustering of a data set has several advantages, specially in data mining applications.

1. Hierarchy corresponds to the way the human mind handles complex situations. Therefore a data analyst will be *comfortable* with hierarchically analysing a data set. The SOM or the GSOM is generally used in data mining to obtain an initial unbiased view of the data set. Therefore we can present this situation as a data analyst attempting to understand the *structure* present in a set of data, about which no current knowledge exists. In such a situation it is better to initially study an abstract view of the data and then gradually look at more detailed views. With a hierarchical set of feature maps, the root will contain a *small* map which may disclose the most significant groupings. Using the above definition we can denote these as the clusters $S_1, S_2, \ldots, S_N$ where N is the number of clusters identified by the data analyst from the root GSOM. These clusters can further be expanded into sub clusters of clusters $S_{11}, S_{12}, \ldots, S_{21}, \ldots$ where $S_{11}, S_{12}, \ldots \subseteq S_1$ and $S_{21}, \ldots \subseteq S_2$, ...which are not *seen* at the root level. Thus, analyst can initially concentrate on the *first level* of clusters and attempt to understand the overall picture and obtain an abstract understanding about the data. The analysis of the lower levels are thus conducted with an understanding of the overall picture.

2. One of the significant problems faced in data mining applications is the

large volumes of data that has to be manipulated. Such large volumes not only require vast computing resources, but will make it difficult and complex for the analyst to understand the data or recognize any patterns. Using hierarchical GSOMs, the analyst will initially study smaller (less spread) maps and will identify the clusters separately. Further detailed analysis can be conducted on the individual clusters $S1, S_2 \ldots S_N$ (where $S_1 \cap S_2 \cap \ldots \cap S_N = \emptyset$) as separate data sets. Such separate analysis does not result in data loss since the upper level of the hierarchy maintains a *record* of the *positions* of each cluster and their relationships with others. During such analysis it might also become apparent that

(a) A certain cluster may not of interest to the current application. Further analysis can thus be terminated on this cluster.

(b) A cluster might be found to have sufficiently spread, where further sub clustering is not possible or of no interest. Further spreading out can also be terminated in such cases.

In both the above instances a section of the data set is found which does not require further processing (because interesting properties with respect to the application has already been found), and can thus save time and resources. Without hierarchical clustering such clusters would also have been spread out unnecessarily.

Hence, hierarchical clustering can help in understanding complex data structure and also can result in faster and less complex data processing.

5.4.2 Hierarchical Clustering Using the Spread Factor

In chapter 3, we derived an equation to calculate the growth threshold (GT) for a GSOM from a given spread factor (SF). The work on dynamic feature maps in the past has concentrated on obtaining an accurate topographical mapping of the data. But for knowledge discovery applications it is also very important for the data analyst to have some control of the growth (or spread) of the map itself. Since feature maps are generally used to gain an initial idea of the data in data mining applications, it may not be necessary to achieve high accuracy in cluster identification and a controllable growth may become even more important. Therefore the spread factor becomes useful for the data analyst in the following instances.

1. Since the spread factor (SF) takes values in the range 0 to 1, where 0 is the least spread and 1 is the maximum spread, the data analyst can specify the amount of spread required. Generally for knowledge discovery applications, if no previous knowledge of the data exists, it would be reasonable to use a low spread factor at the beginning (say between 0 and 0.3). This will produce a GSOM which may highlight the most significant clusters. From these initial clusters the analyst can decide whether it is necessary to study a further spread out version of the data. Else the analyst can select the

regions of the map (or clusters) that are of interest, and generate GSOMs on the selected regions using a larger SF value. This will allow the analyst to do a finer analysis on the subsets of data of interest, which have now been separated from the total data set. Figure 5.4 shows how this type of analysis can produce an incremental hierarchical clustering of the data.

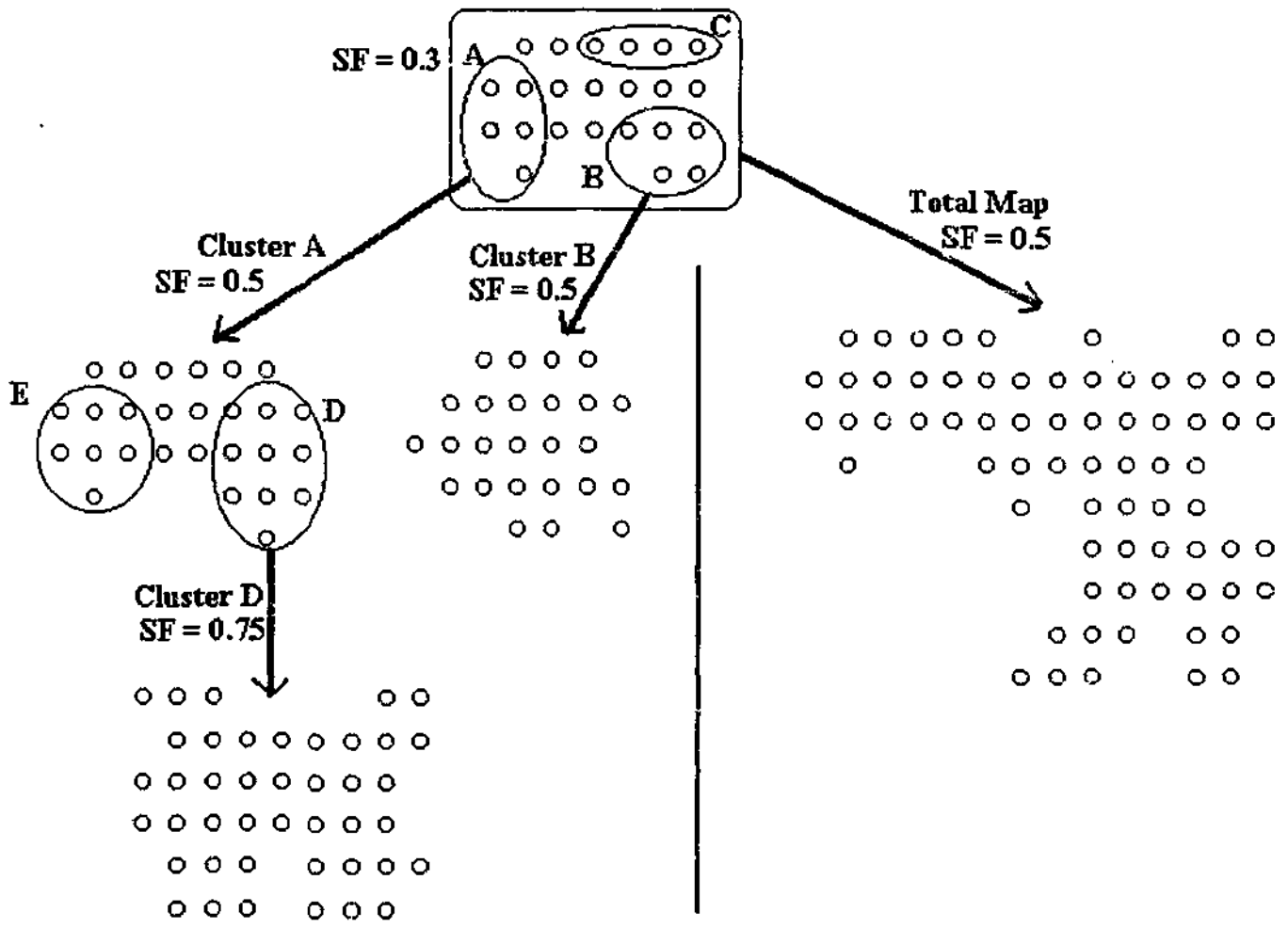


Figure 5.4: The hierarchical clustering of a data set with increasing spread factor (SF) values

2. During cluster analysis, sometimes it is necessary (and useful) for an analyst to study the effect of removing some of the attributes (dimensions) on the

existing cluster structure. This might be useful in confirming opinions on non-contributing attributes on the clusters. The spread factor facilitates such further analysis since it is independent of the dimensionality of the data. This is a very useful feature since the growth threshold (which is derived using the spread factor) depends on the dimensionality of the data.

3. When comparing several GSOMs of different data sets it would be useful to have a measure for denoting the spread of the maps. Since the dimensions of the data sets could be different, the spread factor (independent of dimensionality) can be used to identify the maps, across different data sets. This also opens up very useful opportunities for an organisation that wants to automate this process of hierarchical clustering. The system can be configured to start clustering with an initial value of $SF = a$ and gradually continue until $SF = x$, where $a < x$.

Figure 5.5 shows different options available to the data analyst with the GSOM, due to the control achieved by the SF. The figure shows the initial GSOM generated with a smaller SF (in the figure $SF = 0.3$). The clusters are then identified and expanded at a higher level of SF (in the figure this is 0.5). At each level, the data analyst has the choice of expanding the whole map or expand some of the clusters with their full set of attributes or expand using only a selected sub-set of the attributes. In most cases the data analyst will use a combination of these methods to obtain an understanding of the data itself. In the next section

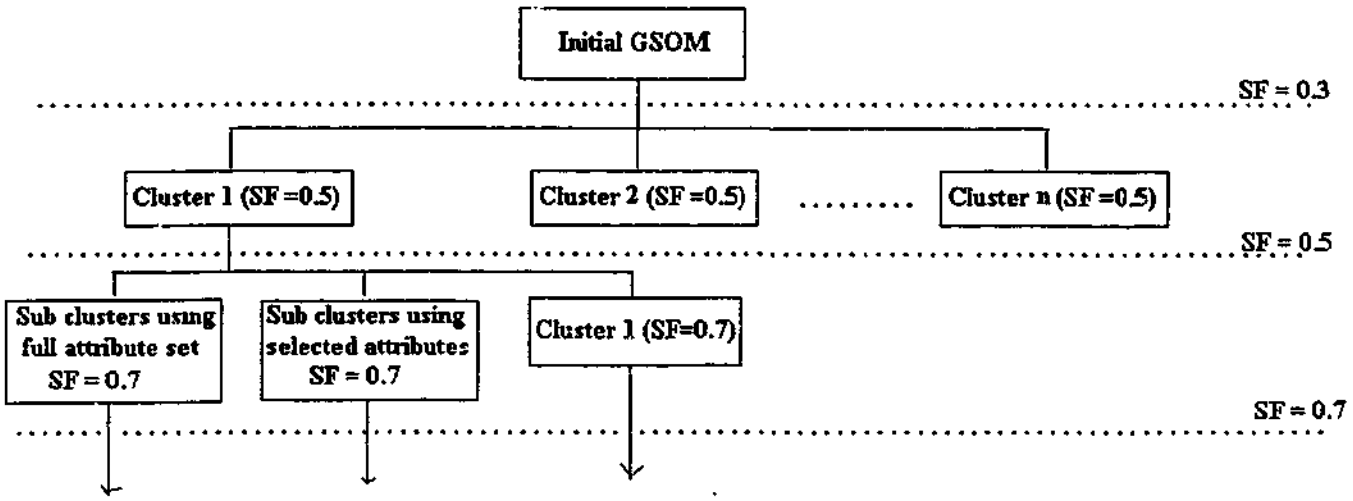


Figure 5.5: The different options available to the data analyst using GSOM for hierarchical clustering of a data set

we formalise this concepts by developing an algorithm for hierarchical clustering using the GSOM.

5.4.3 The Algorithm for Implementing Hierarchical Clustering on GSOMs

The hierarchical clustering of a data set using the GSOM can be expressed as the following algorithm.

1. Generate an initial GSOM on the total data set S with spread factor SF'. Let this map be denoted as $GMAP_{S,SF'}$.

2. Identify the clusters S_i where $\bigcup_{i=1}^{N} S_i \Longrightarrow S$ and $S_l \bigcap S_m = \emptyset \;\; \forall S_l, S_m \subset S$, and $l, m \in [1..N]$. For the cluster identification process, the traditional

visualisation or the data skeleton method presented in chapter 4 can be used.

3. Identify the clusters $IC = [S_j | j \in 1..N]$ and $|IC| = N' \leq N$, which are of interest for the current application.

4. From the clusters of interest IC, identify and remove any non-contributing attributes, where non-contributing attributes are those attributes that do not contribute to the clustering significantly.

5. Identify clusters S_t with special attributes of interest D', where $D' < D$ (D is the dimension of the original data), which may justify consideration for partial dimension analysis.

6. Generate GSOMs $GMAP_{S_t,SF''}, SF' > SF''$ for cluster S_t selected in (4) and (5) above.

7. Repeat from step (3) until the data set is separated into sufficiently *pure* clusters.

5.5 Experimental Results of Using the SF indicator on GSOMs

In this chapter we describe the usefulness of the spread factor and its usage as a control measure in hierarchical clustering of the GSOM. In this section, we use two real data sets to demonstrate the functionality of the spread factor. Initially,

an experiment is performed to demonstrate the effect of *high* and *low* SF values in spreading out of a GSOM. A second experiment is then presented to demonstrate the hierarchical clustering ability of the SF. Finally a third experiment is described, where a data set of high dimension is used to highlight the potential of the GSOM using the SF, for real applications. The data set used for the first two experiments is the animal database (Appendix A), consisting of 99 animals [Bla98], which has been used in chapters 3 and 4. For the third experiment, a human genetics data set of 42 dimensions is used [Cav94].

5.5.1 The Spread of the GSOM with increasing SF values

The purpose of the first experiment is to demonstrate the effect of the spread factor on the GSOM. Therefore we have selected 50 animals from the animal database, and generated 2 GSOMs as shown in Figures 5.6 and 5.7. A low SF value of (0.1) is used for the GSOM in Figure 5.6 and a high SF value of (0.85) for the GSOM in Figure 5.6.

As stated above, Figure 5.6 shows the GSOM of the animal data with a low (0.1) spread factor. It can be seen from this figure that the different types of animals have been grouped together and also similar sub-groups have been mapped closer to each other. Since we have used a very low SF value to demonstrate the difference in spread with a very high spread factor (in Figure 5.7), the clusters in Figure 5.6 are not clearly visibe. A SF value of around 0.3 would have given a better visual picture of the clusters. But it is still possible to see in Figure 5.6,

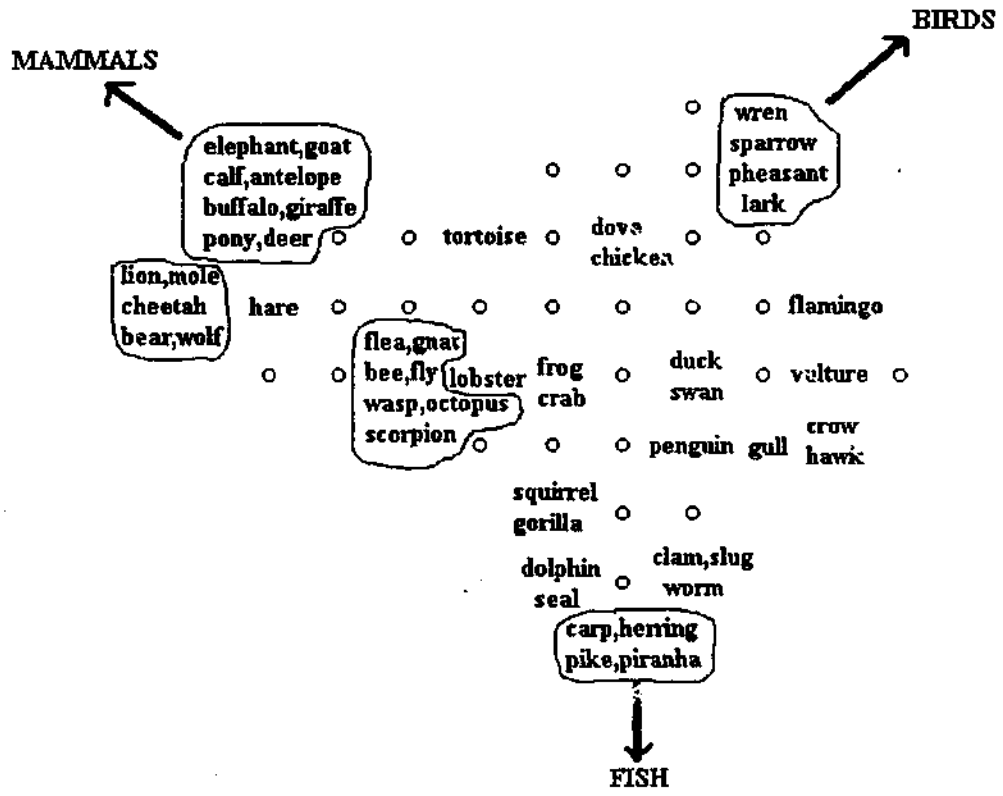


Figure 5.6: The GSOM for the animal data set with SF = 0.1

that there are 3 to 4 main groupings in the data. One of the main advantages of the GSOM over the SOM can highlighted by this figure. The GSOM indicates the groupings in the data by it's shape even when generated with a low SF value. Figure 5.6 has branched out in three directions and this indicates that there are three main groupings in the data (mammals, birds and fish). The *insects* have been grouped together with some other animals but does not show significantly separated at this low level of SF due to a lesser number of insects present in the data set.

Figure 5.7 shows the same data set mapped with a higher (0.85) spread factor.

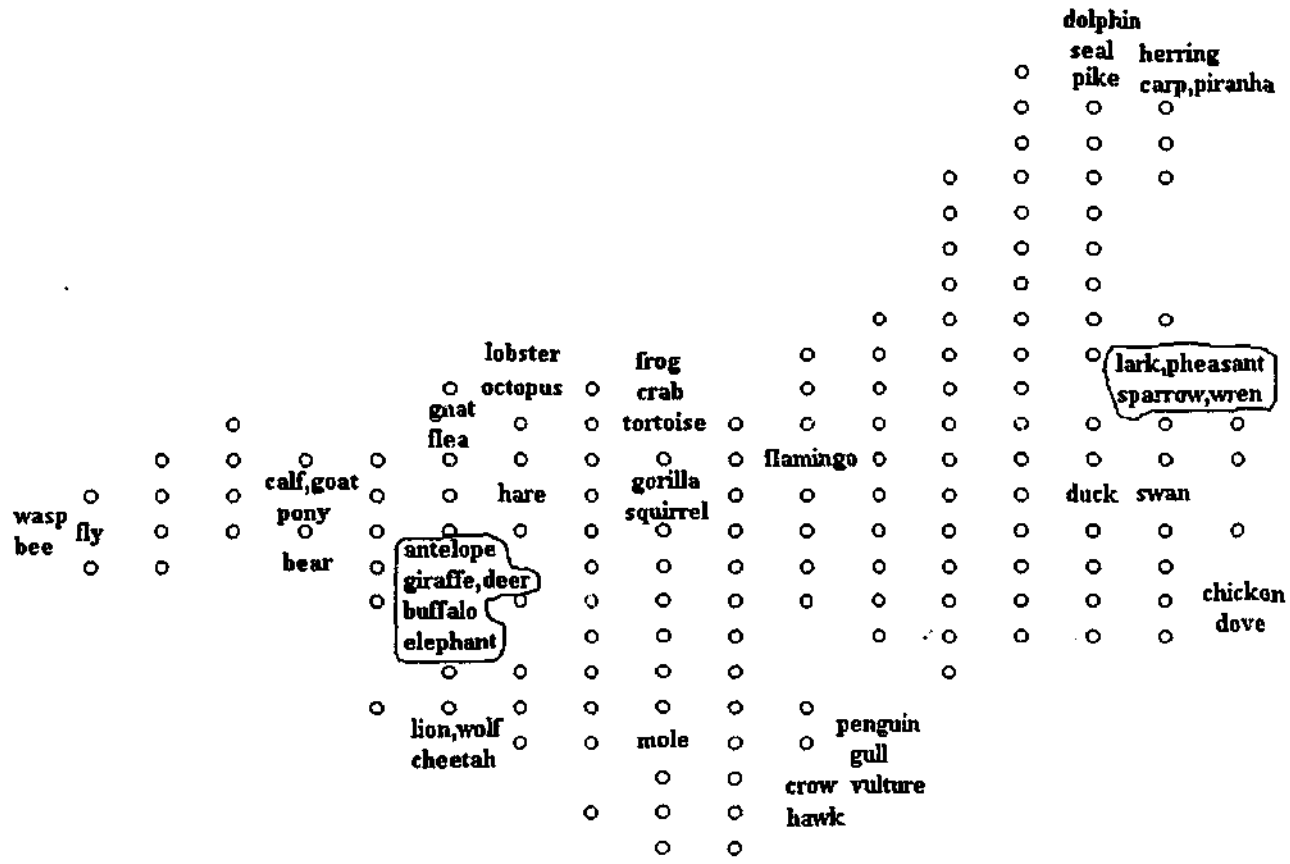


Figure 5.7: The GSOM for the animal data set with SF = 0.85

It is possible to see the clusters clearly, as they are now spread out further. The clusters for *birds, mammals, insects* and *fish* have been well separated. Since figure 5.7 is generated with a higher SF value even the sub-groupings have appeared in this map. The *predatory birds* have been separated into a separate sub-cluster from other birds. The other sub-groups of birds can be identified as *airborne* (lark, pheasant, sparrow) and *non-airborne* (chicken, dove) and *aquatic* (duck, swan). The flamingo has been completely separated due to it being the only *large* bird in the selected data. *The mammals* have been separated into *predators* and *non-predators* and the non-predators have been separated into *wild* and *do-*

mestic sub-groups.

With this experiment it can be seen that the SF controls the spread of the GSOM. An interesting observation from this experiment is, the way the GSOM can *branch out* to represent the data set. Due to this flexible shape of the network, the GSOM can represent a set of data with a fewer number of nodes (at the equal value of spread factor) compared to SOM. This has been proved experimentally by extensive testing on different data sets, the results of which were reported in our previous work [Ala98e],[Ala98a],[Ala99b], [Ala00a]. This becomes a significant advantage when training a network with a very large data set, since the reduction in the number of nodes will result in a reduction in processing time and also less computing resources are required to manage the smaller map.

5.5.2 Hierarchical Clustering of Interesting Clusters

Although we could expand (spread out) the complete map with a higher SF value for further analysis as shown in Figure 5.7, it would sometimes be advantages to select and expand only the areas of interest as shown in Figure 5.5. The data analyst can therefore continue hierarchically clustering the selected data until a satisfactory level of clarity is achieved. Figure 5.8 shows the hierarchical clustering of the same data set used in the previous experiment. The upper right corner of Figure 5.8 shows a section of the GSOM from Figure 5.6. We have assumed that the analyst is interested in two specific clusters (as shown with circles) and

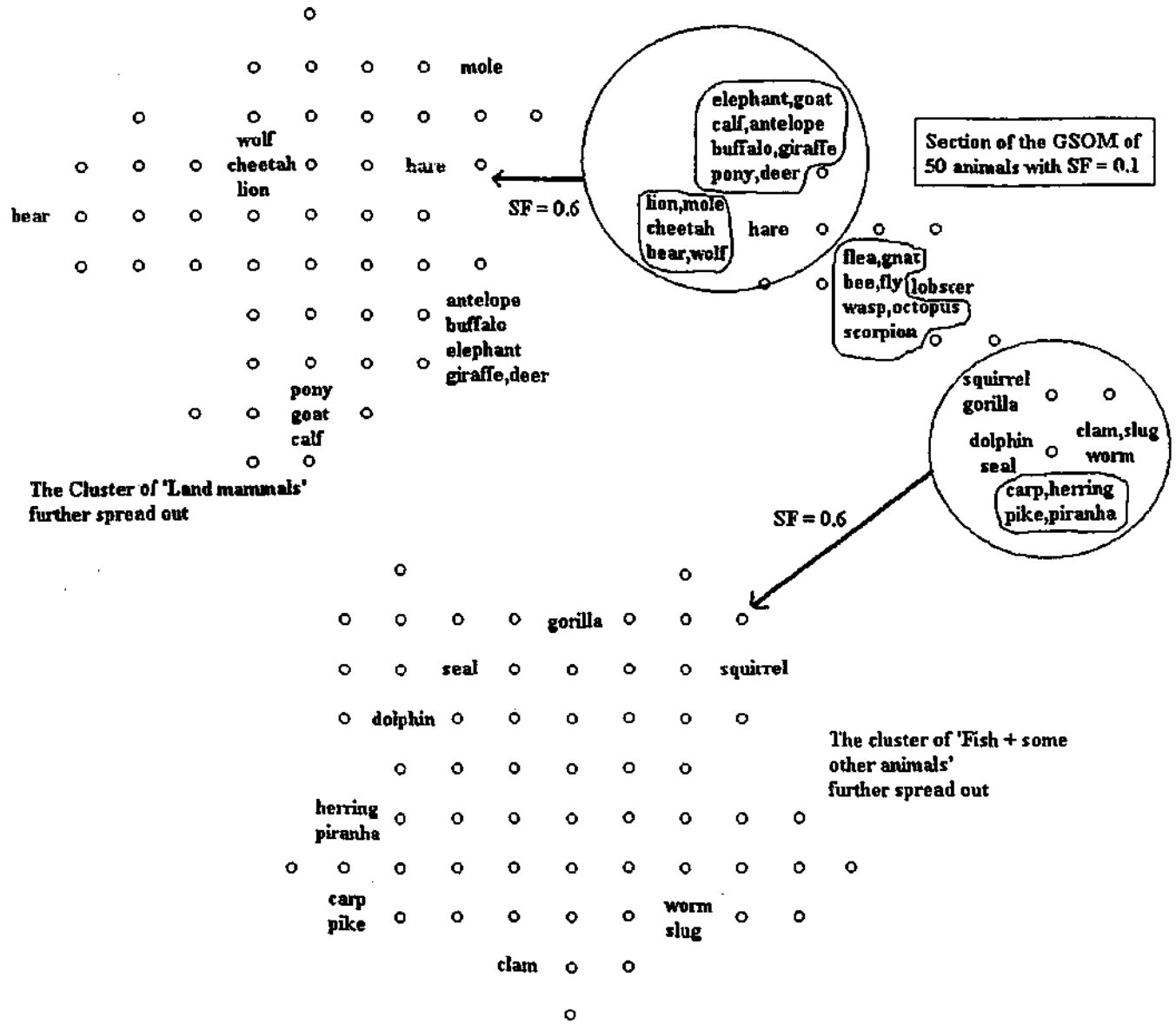


Figure 5.8: The mammals cluster spread out with SF = 0.6

requires a more detailed view of the clusters. Two separated GSOMs have been generated for the selected clusters, using a higher (0.6) SF value. The sub clustering inside the selected cluster is now clearly visible. The non-domestic and non-predatory mammals have been mapped together because they have the same attribute values. The predators (wolf, cheetah, lion) and the domestic mammals

(pony, goat, calf) have been separated. The reasons for the other separations can also be investigated. For example, it might be important to find why *bear* has been separated from the other predatory mammals. On further analysis it was found that the *bear* is the only predatory mammal *without a tail* in this data set. The shape of the GSOM, by branching out, clearly brought the *bear* to our attention. In a more realistic data mining application we might identify interesting outliers with this type of analysis. The *mole* and the *hare* have been separated from their respective groups (non-domestic, non-predatory mammals and predatory mammals) due to their smaller size.

Such analysis can also be carried out on the other cluster. The fish has been separated from the other animals which were clustered close to fish. The reasons for such closeness would become apparent with further analysis of the *new spread* out GSOM. Since the data are already well separated, it would not be useful to further spread out the current clusters. But in a more complex data set it might be necessary to use several levels of GSOMs with increasing SF values to gradually obtain a better understanding of the data set.

5.5.3 The GSOM for High Dimensional Human Genetic Data Set

In this section we generate a GSOM for a more realistic data set with 42 dimensions. We used the simpler animal data set for our earlier experiments since it

was better suited to explain the functionality of the GSOM and the effect of the spread factor. The purpose of this experiment is to demonstrate the ability of the GSOM to map a much higher dimensional and complex set of data. The human genetics data set consists of the genetic distances of 42 populations of the world which has been calculated using gene frequencies [Cav94]. The genetic information is derived from blood samples taken from individuals of a population inhabiting particular areas. The presence of certain genes have been identified in those blood samples and these genes have been used to describe the individual. With sufficiently large samples, the individual gene frequencies have been used to calculate an average for each population. The data set has been generated by selecting 42 populations from an initial 1950 populations. The *genetic distance* between the populations has been calculated with a measure called the F_{st}, which has specifically derived to measure distances between populations. The F_{st} calculation uses a form of normalization to account for frequencies that are not normally distributed. Special terms have also been added to correct any sampling errors. Thus F_{st} has been described as a better measure of genetic distance than the Euclidean distance [Cav94].

We have used a SF=0.5 to generate the GSOM for the genetics data and this mapping is shown in Figure 5.9. It can be seen from this figure that the map is spread mostly according to the geographic localities. But some interesting deviations could be seen, such as the *Africans* being separated into two sub-groups A

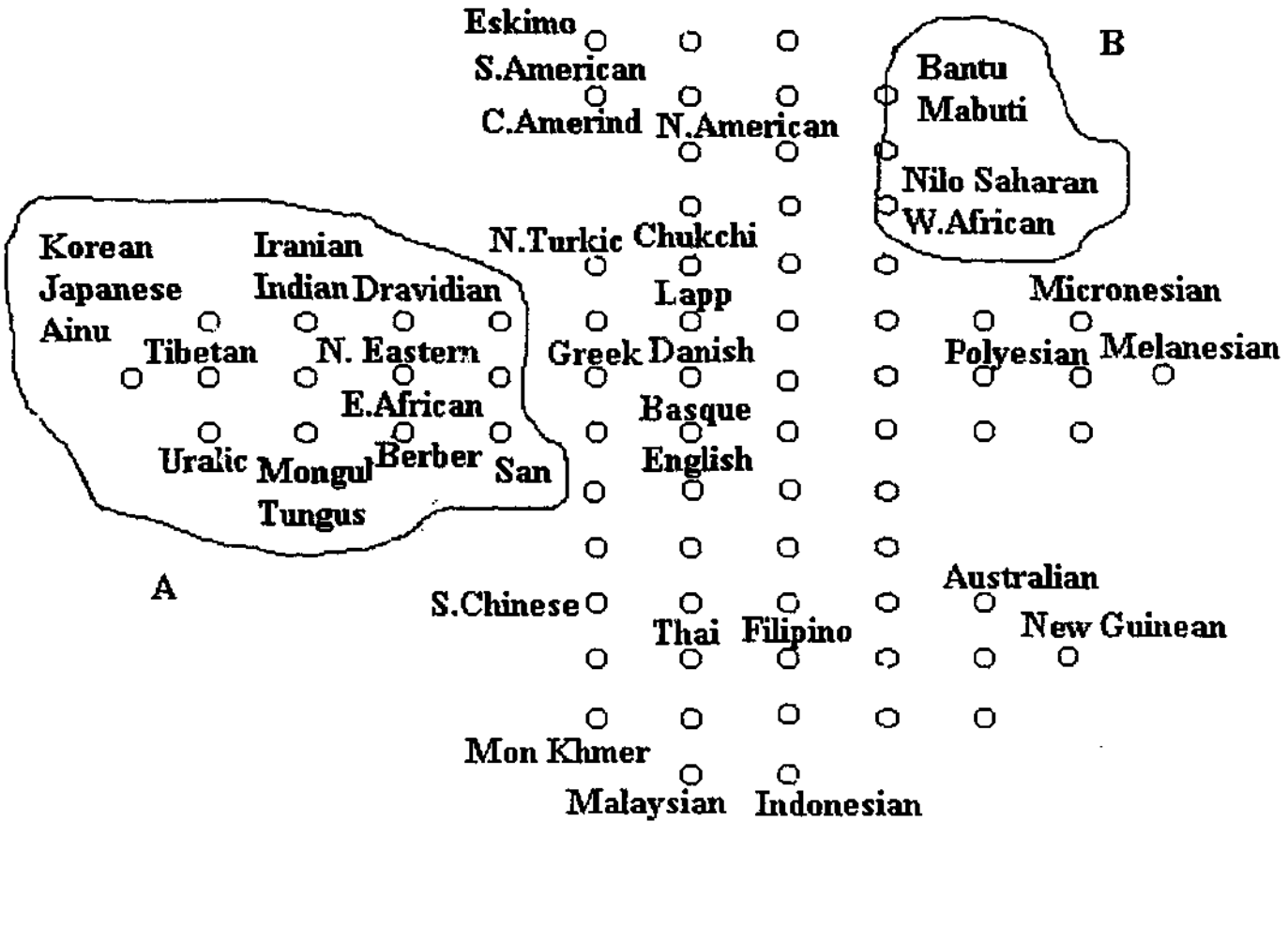


Figure 5.9: The Map of the Human genetics Data

and B.

Figure 5.10 shows cluster A further spread out for clearer viewing or detailed analysis. The *left branch* of the GSOM (cluster A) in Figure 5.9 was picked for further spread since it showed a clustering of populations which were generally thought to be *different*. The resulting GSOM with a SF=0.6 is shown in Figure 5.10.

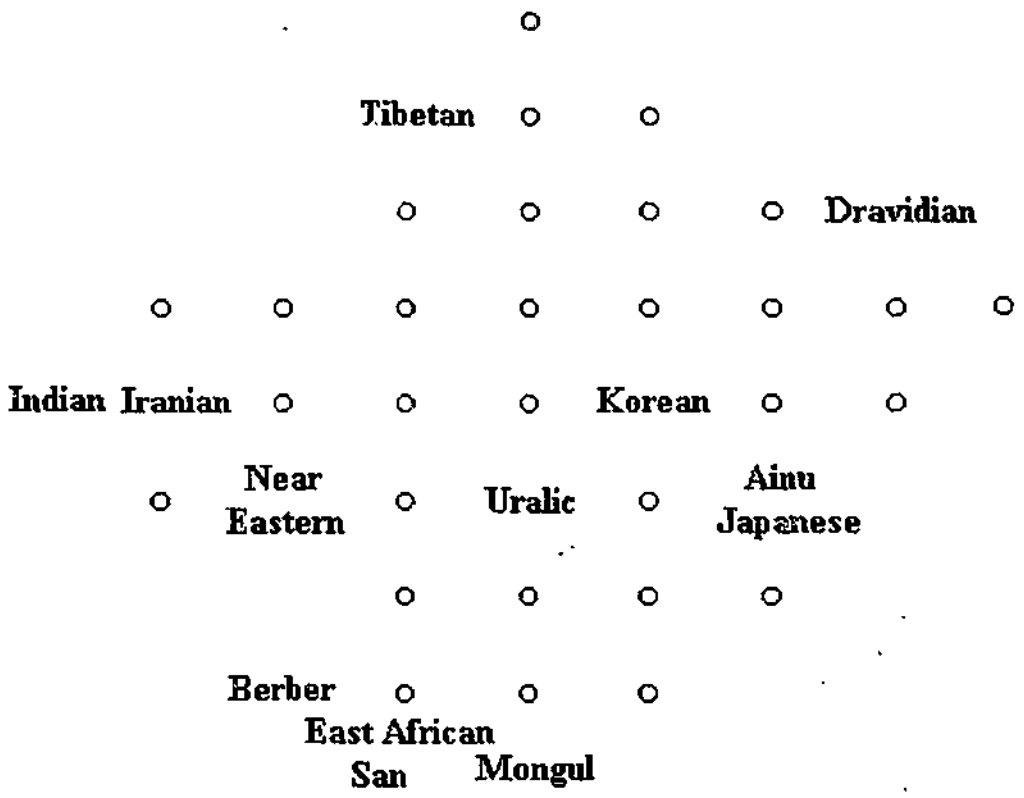


Figure 5.10: Further expansion of the genetics data

Figure 5.10 shows that the two Indian populations (Indian and Dravidian) are further apart. The African populations have been mapped close together and do not seem to have any significant sub groups among them at this level.

5.6 Summary

The main focus of this chapter is the use of spread factor (SF) in controlling the growth of GSOM, and its value in data mining applications. The SF is provided as a parameter at the beginning by the analyst to indicate the *amount of spread* required, and is independent of the dimensionality of the data. Therefore the

analyst has the freedom to decide on the amount of map spread for a specific application.

A data analyst using the traditional SOM achieves different spread of the map by changing the grid size. We have shown that such grid size change can cause distortion in the map, and considering that the SOM is used as a visualisation tool, becomes a significant disadvantage. Therefore, by varying the SF in GSOM provides the data analyst with a novel way of spreading the feature maps without such distortion as in SOM. This chapter has also discussed a way of achieving hierarchical clustering using the SF. Such hierarchical clustering is a significant advantage when using large data sets, by enabling the analysis to be carried out separately on clusters. Hence the SF is a unique feature of the GSOM which enhances it's value as a data mining tool.

In the next chapter we discuss a method of extending the GSOM by developing a conceptual layer to facilitate data mining by rule extraction from the GSOM clusters. The conceptual layer is also used to develop a method for monitoring change and movement in data over time.

Chapter 6

A Conceptual Data Model of the GSOM for Data Mining

6.1 Introduction

The SOM has been described as a visualisation tool for data mining applications [Wes98], [Deb98]. The visualisation is achieved by observing the two dimensional clusters of the multi dimensional input data set and identifying the inter and intra cluster proximities and distances. Once such clusters are identified, the data analyst generally develop hypothesis on clusters and the data and can use other methods such as statistics to test such hypothesis. In other instances, attribute values of the clusters can be analysed to identify the *useful* clusters in the data set for further analysis.

Therefore the feature maps generated by the SOM can be used in the preliminary stage of data mining process where the analyst uses such a map to obtain an *initial idea* about the *nature* of the data. The current usage of feature maps in data mining has been mainly for obtaining such an initial unbiased segmentation of the data. Although this is a useful function, the feature maps offer the potential of providing more informative information regarding the data. Developing the feature maps to obtain such additional information can be thought of as a step in the direction of developing the feature maps as a complete data mining tool.

Another limitation of feature maps is that they do not have a fixed shape for a given set of input data. This becomes apparent when the same set of data records are presented to the network in changed order (say, sorted by different dimensions or attributes). Although the map would produce a similar clustering, the positioning of the clusters would be different. The factors which contribute to such changed positioning in SOMs are the initial weights of the nodes and the order of data record presentation to the network, while in the GSOM only the order of records presented to the network, since the node weights are initialised according to the input data values.

Therefore comparing two maps becomes a non-trivial operation as even similar clusters may appear in different positions and shapes. Due to this problem, usage of feature maps for detecting *changes* in data is restricted. A solution to

this problem is the development of a conceptual model of the clusters in the data such that the model is dependent upon the inter cluster distance proportions and independent of the actual positions of the clusters in the map.

Chapters 4 and 5 of this thesis has introduced extensions to the GSOM model. These extensions provide value added to the original feature maps with the ability to automatically separate the clusters and allows hierarchical analysis of the clusters using the spread factor. In this chapter, the data skeleton proposed in chapter 4 is made use of to develop a conceptual model of the clusters present in the data. It is shown that such a model is useful in detecting *changes* in data which is an important function in data mining. A method is also proposed to obtain rules describing the data using the clusters of the GSOM. Therefore work described in this chapter can be considered as a shift of the usage of feature maps from an initial data *probing* method for data mining to a tool that can be used to extract rules from a set of data.

Section 6.2 discuss the development of a conceptual model called the Attribute Cluster Relationship (ACR) model of the GSOM. In section 6.3 the ACR model is used to extract several types of rules useful for the data mining analyst. Section 6.4 identifies the usefulness of data shift monitoring for data mining and introduce a method for data shift identification using the GSOM-ACR model. Section 6.5 discusses the experimental results on our test data set (animal data)

to demonstrate the rule extraction and data shift monitoring methods. Section 6.6 provides a summary of the chapter.

6.2 A Conceptual Model of the GSOM

The need of a conceptual model of feature maps is highlighted in the previous section. It is also discussed in chapters 4 and 5 that a feature map generated by the GSOM can be considered as a representation of the clusters in the data at a certain spread factor. Therefore we can state that a GSOM feature map $GMAP_{SF_1}$ represents the clusters $Cl_1, Cl_2, \ldots Cl_k$ of a data set S_1 at a specified spread factor SF_1,which can be expressed as

$$GMAP_{SF_1} = Cl_1 + Cl_2 + \ldots + Cl_k \tag{6.1}$$

where + represents union and $Cl_i \bigcap Cl_j = \emptyset, \quad \forall Cl_i, Cl_j \subset S_1$. For a spread factor SF_2, where $SF_2 > SF_1$, and for the same data set S_1, the GSOM feature map $GMAP_{SF_2}$ will be

$$GMAP_{SF_2} = \underbrace{Cl_{11} + Cl_{12} + \ldots + Cl_{1l_1}}_{Cl_1} + \underbrace{Cl_{21} + Cl_{22} + \ldots Cl_{2l_2}}_{Cl_2} + \ldots + \underbrace{Cl_{k1} + Cl_{k2} + \ldots + Cl_{kl_k}}_{Cl_k}$$

where

$Cl_{11} + Cl_{12} + \ldots + Cl_{1l_1} \Longrightarrow Cl_1$

$Cl_{21} + Cl_{22} + \ldots + Cl_{2l_2} \Longrightarrow Cl_2$

$\vdots$

$Cl_{k1} + Cl_{k2} + \ldots + Cl_{kl_k} \Longrightarrow Cl_k$

and $l_1 + l_2 + \ldots + l_k \geq k$

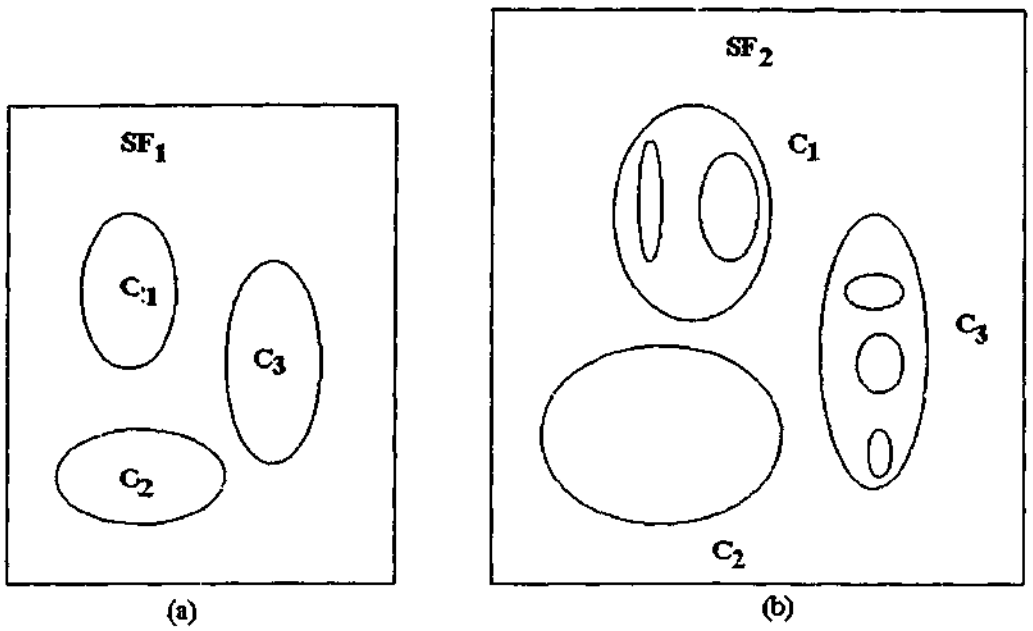


Figure 6.1: The spreading out of clusters with different SF values

Such spreading out of clusters can be illustrated by a diagram as shown in Figure 6.1. With spread factor SF_1, let us assume that there are three clusters as in Figure 6.1(a). These clusters have been further spread out with the spread factor SF_2, $(SF_2 > SF_1)$ and some sub-clusters in C_1 and C_3 can now be visualised in Figure 6.1(b). Cluster C_1 shows two sub clusters and C_3 shows three sub clusters. C_2 does not show such sub groupings and this will occur when the data points inside the cluster has a uniform distribution. It can also be seen that the original clustering is still apparent in the map.

As such the clusters appearing in a map depends on the spread factor, with

$$N_{SF_1} \leq N_{SF_2} \tag{6.2}$$

where N_{SF_1} and N_{SF_2} are the number of clusters for a given set of data for spread factor values SF_1 and SF_2 respectively and $SF_1 \leq SF_2$. The equality in equation 6.2 is satisfied when the increase in spread factor is not sufficient to spread out the clusters or when the intra cluster data points are uniformly distributed. Since the number of clusters in a feature map depends on the spread factor used, we now define a cluster indicator $CI_{i,SF'}$ to represent a cluster i of a GSOM for a spread factor SF' as

$$CI_{i,SF'} = ((\mu_{A_{i,1}}, SD_{i,1}, Max_{i,1}, Min_{i,1}), (\mu_{A_{i,2}}, SD_{i,2}, Max_{i,2}, Min_{i,2}), \ldots,$$
$$(\mu_{A_{i,D}}, SD_{i,D}, Max_{i,D}, Min_{i,D}))$$

where $\mu_{A_{i,k}}, k = 1..D$ represents the average attribute (dimension) value for attribute k in cluster i, $(i = 1..N_{SF'})$. SD_i is the standard deviation of the intra cluster data distribution in cluster Cl_i and $Max_{i,j}$ and $Min_{i,j}$ are the maximum and minimum values for each attribute j for Cl_i, and D is the dimension of the data set.

The average weight value is calculated as follows :

$$\mu_{A_{i,j}} = \frac{\sum_{k=1}^{n} A_{i,j,k}}{n} \tag{6.3}$$

where n is the number of nodes assigned to cluster i, and $A_{i,j,k}$ is the value of j^{th}

attribute ($j = [1..D]$) of node $k, k = [1..n]$, in cluster i.

Once such a set of cluster identifiers have been selected, they can be considered as a set of conceptual identifiers for the clusters in the map, for a specific spread factor. Such a set of identifiers are called conceptual cluster identifier since they represent the clusters without considering their positions in the feature map. Once the cluster identifiers have been calculated it is necessary to join them together with the attributes into a model which can facilitate the automated analysis of inter and intra cluster relations. The development of such a model is described in the next section.

6.2.1 The Attribute Cluster Relationship (ACR) Model

The ACR model is developed as a method of implementing the conceptual cluster identifiers. The model has been generated by developing two layers of nodes on top of the GSOM and adding connections between the layers as shown in Figure 6.2. There are three layers in the ACR model and they are

1. the physical layer or the GSOM
2. the cluster summary layer
3. the attribute layer

The GSOM clusters are linked to cluster summary nodes, which represents the identifier for the respective cluster. Each cluster summary node is linked to the

attribute layer nodes which enables the rule generation and this will be described in section 6.3. The GSOM represents the clusters at physical level, and as such can represent the clusters at different shapes and positions in different maps. The cluster summary layer and the attribute layer with the links in between them represents the conceptual view of the clusters which will only change due to a change in the attribute values of the data.

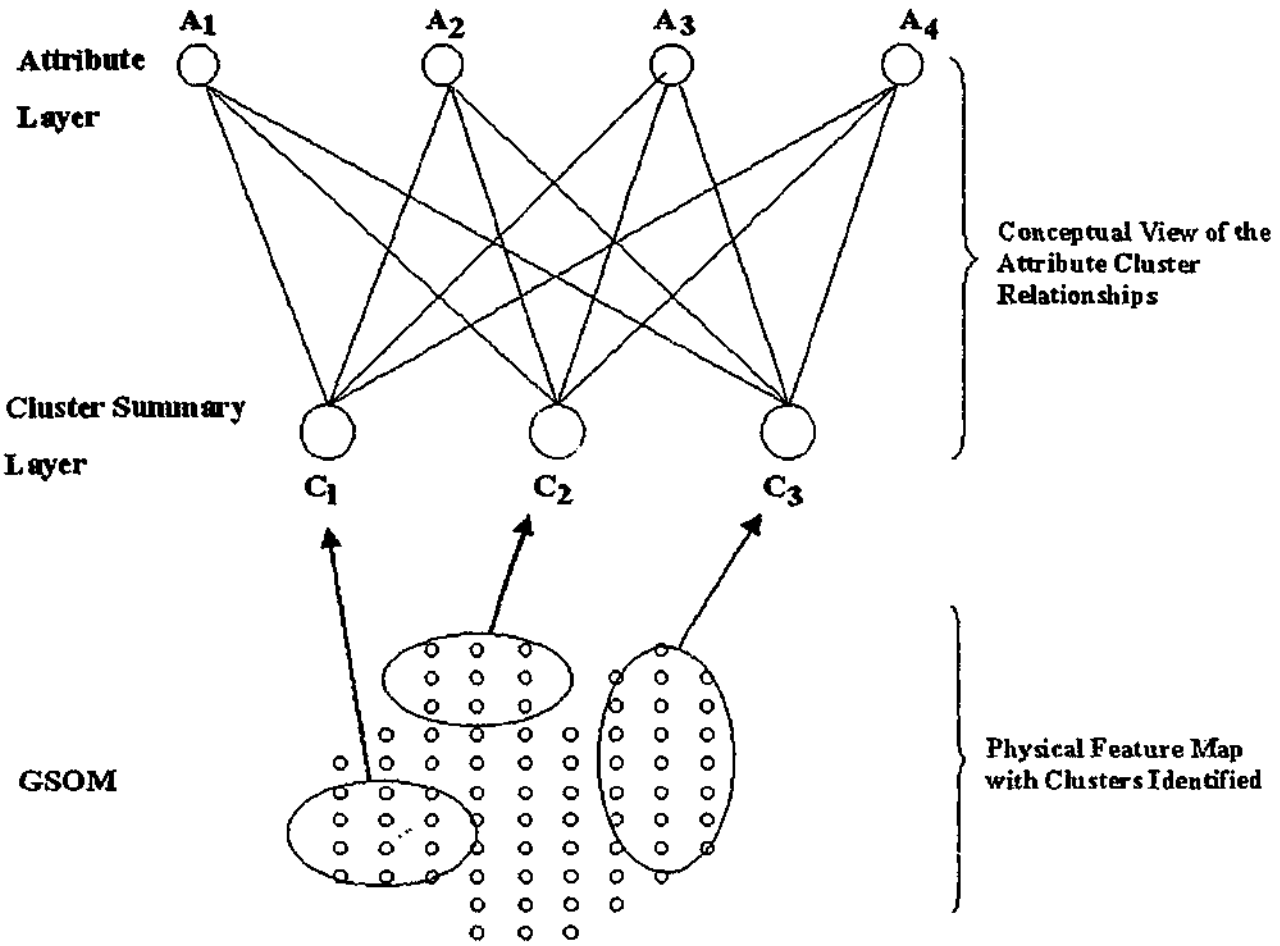


Figure 6.2: The ACR model developed using the GSOM with 3 clusters from a 4 dimensional data set

Cluster Summary Layer

In Figure 6.2 the cluster summary layer is built on top of the GSOM. This layer consists of a set of nodes where each node represents a cluster in the GSOM. Therefore the map has to be built and the clusters separated before the cluster summary layer can be built. The nodes in this layer act as identifiers for the clusters. Once the clusters are identified from the GSOM, the nodes in each separate cluster are linked together. A set of cluster summary layer nodes are generated and each cluster is linked to a summary node. The attribute values of the clusters are calculated by averaging over all the nodes in the cluster according to equation 6.3. The standard deviation and the maximum and minimum values are also found across the nodes in each cluster for each attribute. These values are then stored in the attribute cluster links between the two layers. Therefore the summary nodes are representation of the respective cluster which not only provide information about the average attribute values, but also can be used to analyse the distribution of the cluster.

Attribute Layer

The attribute layer consists of a set of nodes where each node represents one dimension (attribute) in the data set. In this model the attributes layer is generated after the cluster summary layer, and the nodes in the two layers are fully connected. The attribute layer nodes do not store any information and only serve as initiating points for querying or rule generation from the map. The rule gen-

eration method is described in section 6.3.

The steps in ACR model generation can be stated more formally as follows.

1. Generate a GSOM $GMAP_{SF'}$ for a given set of data S_1, with a spread factor SF'.

2. Build the data skeleton and use the cluster separation method suggested in chapter 4. Define cluster identifier Cl_i for each of the clusters separated.

3. For each cluster $Cl_i, i = 1..k$, generate a cluster summary node Cls_i.

4. Generate the attribute layer nodes $Cla_j, \quad j = 1..D$, where D is the dimension of the data.

5. Add the links between the two layers. Calculate the attribute average values according to the equation 6.3 and assign the values as $\mu_{A_{i,j}} \Longrightarrow$ link (Cls_i, Cla_j). The link (Cls_i, Cla_j) is named as m_{ij} (and will be referred to as such in the rest of this chapter).

6. Identify any attributes which have a similar presence in all the clusters as *non-contributing attributes*. For all non-contributing attributes A_{ij} make $m_{ij} = -1, \quad$ for every node i in the cluster summary layer.

7. Define a threshold of non-significance T_{ns} for attributes which have a low mean presence in certain clusters. For such non-significant attributes $A_{j'}$ in clusters $Cl_{i'}$, assign $m_{ij} = -1$.

6.3 Rule Extraction from the Extended GSOM

One of the main limitations of using artificial neural networks for data mining is that they do not provide *explainable* results. This contrasts with tools such as Decision Trees, where explainable rules are provided. Such rules are quite important for many data mining tasks where explainability is one of the most important requirements.

Therefore neural networks are generally used for applications where it is sufficient to obtain accuracy of classification or prediction, and as such results can be made use of without the need of explainability. For example, if a direct mail firm develops a model that can accurately predict which group of a set of prospective customers are most likely to respond to a solicitation, but they may not care how or why the model works [Ber97b]. There are other applications and situations where the ability to explain the reason for a decision is crucial. An example is the approval of a credit application where it would be more acceptable both to the loan officer and the credit applicant to hear a reason for the denial of a loan.

There has been several attempts at extending and enhancing current neural network models such that rules can be extracted from them [Avn95], [Cha91], [DeC96], [Fu 98]. Such work has mainly concentrated on enhancing the neural networks with the supervised learning paradigm with rule extraction capabilities. We have used the conceptual model of the GSOM as a method for extracting rules

which provides explainabilty to the unsupervised clusters obtained [Ala98c]. This is achieved by obtaining a set of *descriptions* of the clusters, using the attributes of the data set. Therefore, instead of the traditional manual analysis, the proposed model can be used to extract rules regarding the clusters.

A rule can be generally considered as having the form :

IF *condition* THEN *result*

where the *condition* has to be satisfied for the *result* to be true. There can be different types of rules that can be useful to a data mining analyst. We only consider the following types of rule generation from the GSOM ACR model.

1. Cluster description rules.

2. Query by attribute rules.

In the rest of this chapter, when we refer to an attribute A_j belonging to a cluster i, we refer to the average attribute value $\mu_{A_{i,j}}$ for the cluster, calculated as per the equation 6.3.

6.3.1 Cluster Description Rules

Traditionally the feature maps have been used to identify clusters in a set of data. Once the clusters are selected, the *presence* of the attribute values in the clusters are used to *describe* each cluster. Such descriptions are then used to obtain an initial understanding about the data. The ACR model proposed above

can be considered as a method for automating such traditional cluster analysis, and rules obtained are called the cluster description rules. The initiation point for these rules are the cluster summary nodes and the attribute links of these nodes are *searched* to obtain the most significant attributes for generating the rules. A cluster description rule is of the form

$$IF \bigcap_{i=1}^{n_k} A_i \; THEN \; Cluster = Cl_k \tag{6.4}$$

where Cl_k is the k^{th} Cluster, A_i is the i^{th} attribute, n_k, $n_k \leq D$, is the number of significant attributes in cluster Cl_k. The significance of an attribute is decided by

$$SD_{A_{ij}} \geq T_{ij} \quad 0 \leq T_i \leq 1 \tag{6.5}$$

where $SD_{A_{ij}}$ is the standard deviation of j^{th} attribute in cluster Cl_i and $T_{i,j}$ is a threshold value for attribute A_j in cluster Cl_i. The threshold $T_{i,j}$ is determined for the cluster depending on the needs of the application. Therefore if the analyst is only interested in any attribute which has a *very high* significance for the cluster, then this threshold may be set to a low value. For example, if the attributes A_1, A_3 and A_{10} are considered significant for cluster Cl_5, the rule generated will be :

$$IF \; (A_1 \; AND \; A_2 \; AND \; A_{10}) \; THEN \; Cluster = Cl_5$$

The cluster description rules provide descriptions about the clusters using the unsupervised groupings and as such are useful when the analyst does not have any previous knowledge regarding the data. The data mining analyst may further

use these rules to obtain an unbiased opinion about data regarding which some previous knowledge exists. Comparing the existing knowledge with the rules may provide unforeseen and interesting facts.

6.3.2 Query by Attribute Rules

Although the cluster description rules satisfy the traditional usage of feature maps, the same clusters can further be used to provide additional use for the data mining analyst. The query by attribute rules provide a method by which the data analyst can search for patterns of interest using some existing knowledge or topic of interest. With the cluster description rules, the rule generation was initiated from the cluster summary nodes. With the query by attribute method, the rules are generated starting from an attribute layer node. The initiating points (or query attribute) for these rules can be selected in the following circumstances.

1. The data analyst has certain previous knowledge, and as such would like to obtain more information regarding one or more attributes.

2. The cluster description rules may focus interest on certain attributes, which can be analysed using query by attribute rules.

3. One or both of the above circumstances may result in the data analyst developing hypothesis regarding the relationship between attributes and clusters. Such hypothesis may be confirmed or rejected using query by attribute rules.

The query by attribute . be categorised into two types and they are discussed in the following . .ons.

Type 1

The first type of rules provide relationships between attributes and clusters. These rules are different from the cluster description rules since the initiating point is one more attributes. For example, these types of rules can provide answers for queries such as, *Are there any clusters which are dominated by a high value of attributes* A_1 *and* A_4 *?* As such the condition contains one or more attributes with a relevant threshold value. The threshold can be adjusted by the analyst depending on the needs of the application. Therefore these rules can be described as :

$$IF \bigcap_{p=1}^{n_p} (A_p \Theta T_p) \; THEN \; Cluster = \bigcup_{j=1}^{k'} Cl_j \qquad j \in [1, 2, \ldots k] \qquad (6.6)$$

where n_p is the number of attributes of interest, k' is the number of relevant clusters and Θ is a relational operator. Figure 6.3 is an example of the initiation and generation of the type 1 query by attribute rules. Figure 6.3 shows an ACR model with a query initiated to identify clusters which have a high value of attribute A_2. The data analyst has to specify the threshold value for defining *high* A_2 values. A sorted list of μ_{A_2} values (for the different clusters) can be used in deciding a threshold value T_{A_2}. The rule generated using the ACR model in Figure 6.3, for $T_{A_2} = 0.75$ and Θ *being* $>$ is,

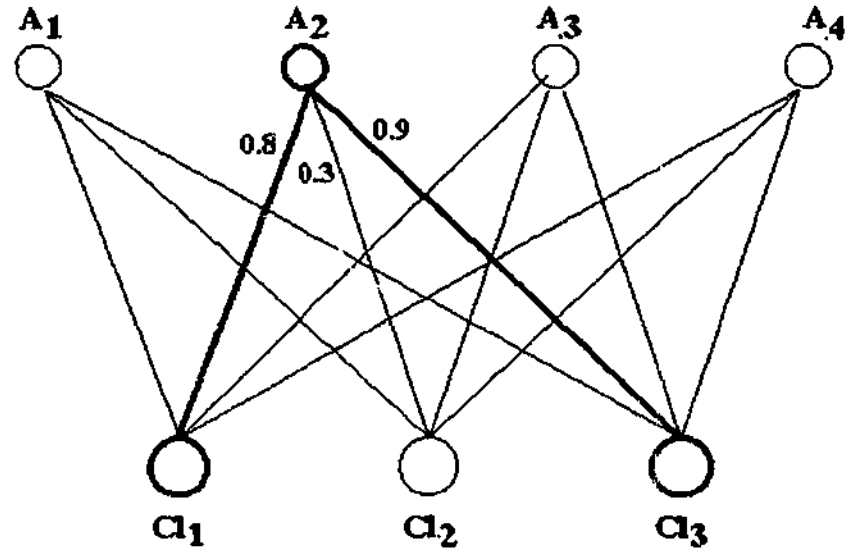


Figure 6.3: Query by attribute rule generation 1

$$IF\ (A_2 > 0.75)\ THEN\ Clusters = Cl_1\ OR\ Cl_3$$

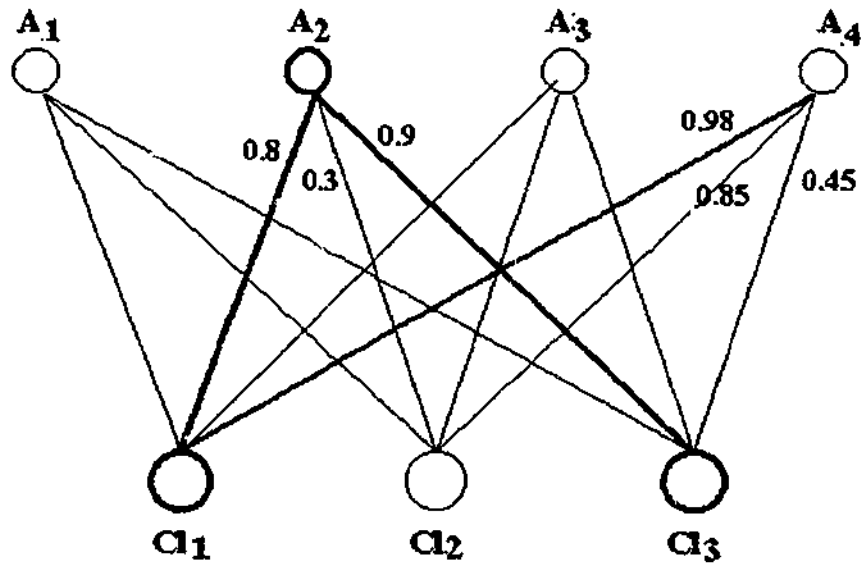


Figure 6.4: Query by attribute rule generation 2

The ACR model in figure 6.4 shows the generation of a rule for identifying clusters with a high value of A_2 and A_4. If $T_{A_2} = 0.75$ and $T_{A_4} = 0.95$ the rule will be,

$$IF\ ((A_2 > 0.75)\ AND\ (A_4 > 0.95))\ THEN\ Clusters = Cl_1$$

Type 2

Generally in cluster analysis, the analyst obtains information regarding relationships between clusters or between attributes and clusters. But it may also be possible to obtain certain relationships or patterns that exist between attributes. Such relationships may be global, i.e. general to the whole data set, or localised, (exist only for certain clusters). Knowledge of such relationships may prove to be useful for a data mining analyst. The second type of rules provide relationships between attributes. Therefore the analyst can provide the attribute or attributes of interest with threshold values and obtain relationships with other attributes in the data set. In other words the condition and result in this rule contains attributes. For example, the analyst can query the ACR model to find attributes which have a *high* relationship with attribute A_1. For instance, such a query may produce the result that attributes A_3 and A_4 have high relationships with A_1, then a rule can be generated as :

$$IF\ (A_1 > T_1)\ THEN\ ((A_3 = 8.3)\ AND\ (A_4 = 9.2))\ IN\ Cluster = Cl_3$$

Since such rules can be used to obtain rules with multiple attributes in the condition, they can be generalised as :

$$IF \bigcap_{p=1}^{n_p} (A_p \Theta T_p) \; THEN \bigcup_{i=1}^{k'} (\bigcap_{j=1}^{n_{p'}} (A_j \Theta T_j) \; IN \; Cluster = Cl_i) \qquad (6.7)$$

where n_p is the number of attributes of interest in the query, $n_{p'}$ is the number of significant attributes in the result for a given cluster, and k' is the number of significant clusters.

6.4 Identification of Change or Movement in Data

As described in the previous sections, feature maps have traditionally been used to identify clusters or groupings in data sets. With the ACR model proposed in section 6.2 it is possible to compare and identify *differences* in separate sets of data. The following sections describe the implementation and the usefulness of such ability for data mining applications. Section 6.4.1 describes the types (categories) of differences that can occur in data sets. Section 6.4.2 discusses the advantage of monitoring change and movement in data sets, and section 6.4.3 proposes a method of comparing GSOMs, and defines a measure for comparing feature maps.

6.4.1 Categorisation of the Types of Comparisons of Feature Maps

In this section we present the concept of comparing different feature maps. We have identified several *types of differences* that can occur in feature maps. For the purpose of describing the method of *identifying differences* we categorise such differences in data as shown in Figure 6.5.

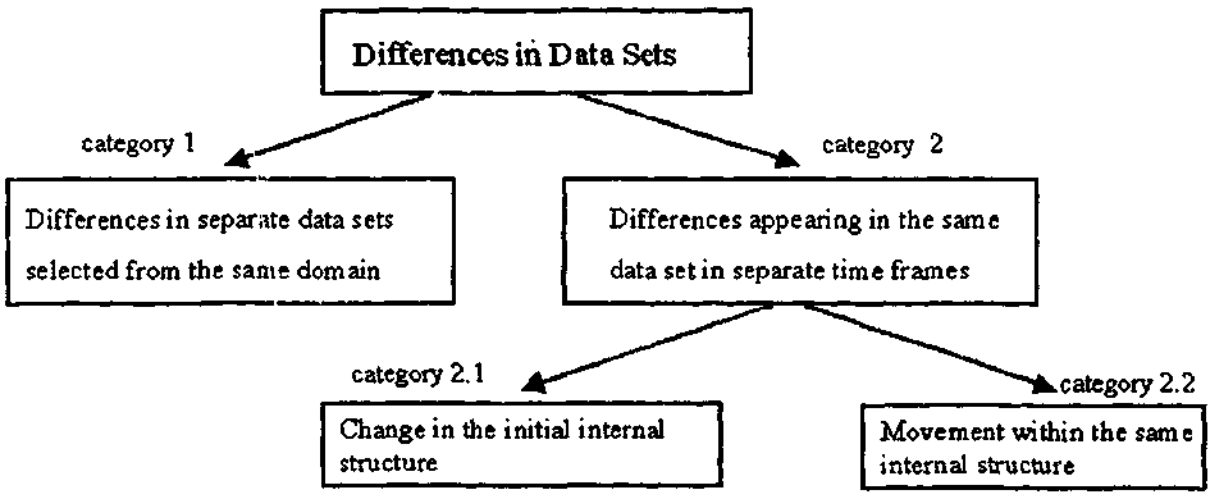


Figure 6.5: Categorisation of the type of differences in data

Category 1 consists of different data sets selected from the same domain. Therefore the attributes of the data could be the same although the attribute values can be different. Category 2 is the same data set but analysed at different points in time. Therefore the attribute set and the records are the same initially. During a certain period of time, if modifications are done to the data records, the data set

may become different from the initial set. As shown in Figure 6.5 we categorise such differences as :

1. Category 2.1 → change in data structure

2. Category 2.2 → movement of data while possessing the same structure

The categories 2.1 and 2.2 are further discussed in this chapter as useful methods for data mining. Therefore a formal definition of *movement* and *change* in data are presented below. Considering a data set $S(t)$ with k clusters Cl_i at time t where $\sum_{i=1}^{k} Cl_i \Longrightarrow S(t)$, and $\bigcap_{i=1}^{k} Cl_i = \emptyset$ categories 2.1 and 2.2 can thus be defined more formally as

Definition 6.1: Change in structure

Due to additions and modifications of attribute values the groupings existing in a data set may change in time. Therefore we say that the data has *changed* if, at a time t' where $t' > t$, and k and k' are the number of clusters (at similar spread factor) before and after the addition of records respectively, if

$$S(t') \Longrightarrow \sum_{i=1}^{k'} Cl_i(t) \qquad where \quad k' \neq k \tag{6.8}$$

In other words, the number of clusters has been increased or decreased due to some changes in the attribute values.

Definition 6.2: Movement in data

Due to the additions and modifications of attribute values the internal *content* of the clusters, *the intra cluster relationships* may change, although the actual

number of clustering remain the same. For a given data set $S(t)$, this property can be described as:

$$S(t') \rightarrow \sum_{i=1}^{N} Cl_i(t') \tag{6.9}$$

and $\exists\ A_{ij}(t')$ such that $A_{ij}(t') \neq A_{ij}(t)$ for some cluster Cl_i.

6.4.2 The Need and Advantages of Identifying Change and Movement in Data

According to the above categorisation of the type of differences in data, category 1 simply means a difference between two or more data sets. Since the data are from the same domain (same sets of attributes), they can be directly compared. Such analysis provide insight into the relationships between the functional groupings and the natural groupings of the data. For example, in a customer database, if the data sets have been functionally separated by region, such regions can be separately mapped. Comparing such maps will provide information as to the similarity/differences of functional clusters to natural clusters by region. Such analysis may provide information which can be used to optimise current functional groupings.

Category 2.1 provides information about the *change* in a data set over time. As defined in the previous section the changes will be an increase or decrease in the number of clusters in the map, at the same level of spread. Identification of such change is useful in many data mining applications. For example, in a survey

for marketing potential and trend analysis, these would suggest the possibility of the emergence of a new category of customers, with special preferences and needs.

Category 2.2 will identify any *movement* within the existing clusters. It may also be that the additions and modifications to the data has in time resulted in the same number of, but different clusters being generated. Monitoring such movement will provide an organisation with the advantage of identifying certain shifts in customer buying patterns. Therefore such movement (category 2.2) may be the initial stage of a change (category 2.1) and early identification of such movements towards the change may provide valuable competitive edge for the organisation.

6.4.3 Monitoring Movement and Change in Data with GSOMs

The GSOM with the ACR model provides an automated method of identifying movement and change in data. The summarised and conceptual view of the map provided by the ACR model makes it practical to implement such map comparison. The method of identifying change and movement in data using the GSOM and ACR model is described in this section.

Figure 6.6 shows two GSOMs and the respective ACR models for a data set S at two instances t_1 and t_2 in time. It can be seen from the maps that significant

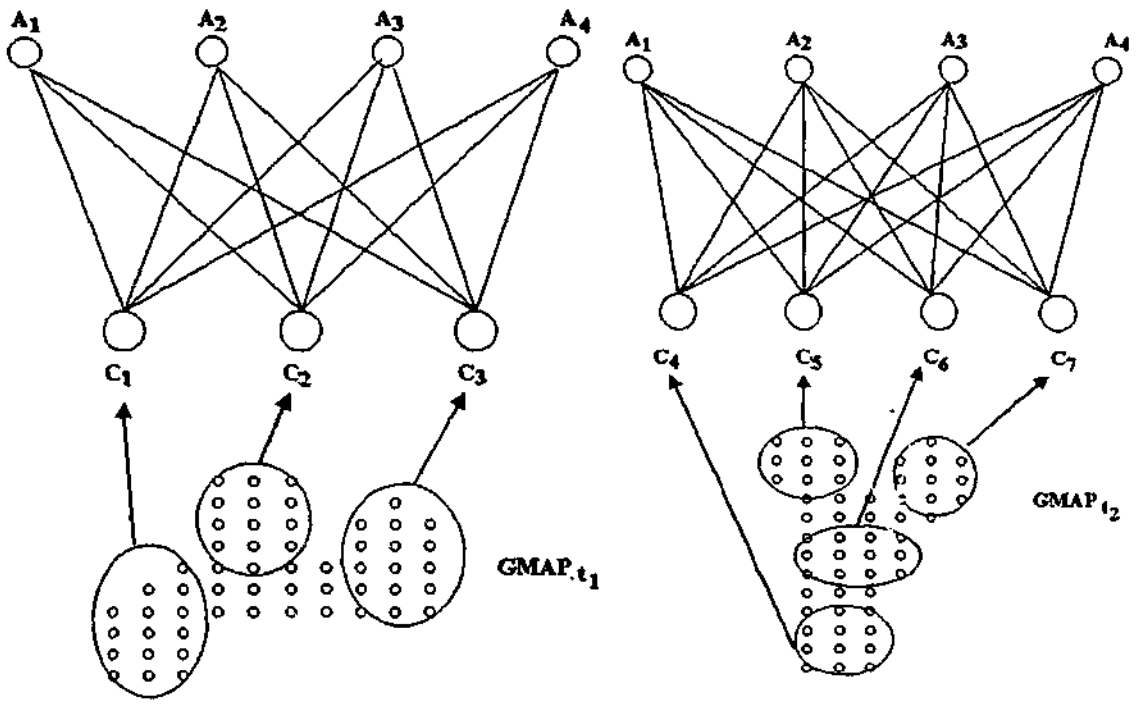


Figure 6.6: Identification of change in data with the ACR model

change has occurred in the data as shown by the increase in the number of clusters. This fact itself does not provide much useful information and the data analyst has to identify the type of change that has occurred and if possible the reason for such a change. In most instances, the reason can be identified from external facts once the type of change is recognized. There are several possibilities for the type of change that has occurred and some of them are

1. An additional cluster has been added while the earlier three clusters have remained the same.

2. The grouping has changed completely. That is, the four clusters in $GMAP_{t_2}$ of Figure 6.6 are different from the three clusters in $GMAP_{t_1}$.

3. Some clusters may have remained the same while others might have changed,

thus creating additional clusters. For example, C_1 and C_2 can be the same as C_4 and C_5 while C_3 has changed. That is, C_3 could have been split into two clusters to generate clusters C_6 and C_7. Another possibility is that C_3 does not exist with the current data and instead the two clusters C_6 and C_7 have been generated due to the new data.

We propose a method to identify the differences in GSOMs by comparing the clusters in the ACR model. First the following terms are defined considering two GSOMs $GMAP_1$ and $GMAP_2$ on which ACR models have been created. These terms are then used in the description of the proposed method.

Definition 6.3: Cluster Error ((ERR_{Cl})

A measure called the cluster error (ERR_{Cl}) between two clusters in two GSOMs is defined as :

$$ERR_{Cl}(Cl_j(GMAP_1), Cl_k(GMAP_2)) = \sum_{i=1}^{D} |A_i(Cl_j) - A_i(Cl_k)| \qquad (6.10)$$

where Cl_j and Cl_k are two clusters belonging to $GMAP_1$ and $GMAP_2$ respectively and $A_i(Cl_j)$, $A_i(Cl_k)$ are the i^{th} attribute of clusters Cl_j and Cl_k.

The ERR_{Cl} value is calculated using the ACR models for the GSOMs. During the calculation of the ERR_{Cl}, if a certain attribute which was considered non-significant to the cluster $Cl_j(GMAP_1)$ is considered as a significant attribute for $Cl_k(GMAP_2)$, then the two clusters are not considered to be similar. Therefore we define a term *significant non-similarity* as shown below.

Definition 6.4: Significant-Non-Similarity

If $\exists\ A_i(Cl_j), A_j(Cl_k)$ such that $A_j(Cl_k) = -1$ and $A_i(Cl_j) \neq -1$ then Cl_j and Cl_k are significantly non-similar. The -1 above is considered since the non-significant and non-contributing attribute cluster links (m_{ij}) are assigned -1 values as described in section 6.2.

Definition 6.5: Cluster Similarity

We define two clusters Cl_j and Cl_k as similar when the following conditions are satisfied.

1. Does not satisfy the significant non similarity condition.

2. $ERR_{Cl}(Cl_j, Cl_k) \leq T_{CE}$ where T_{CE} is the threshold of cluster similarity and has to be provided by the data analyst depending on the level of similarity required. If complete similarity is required, then $T_{CE} = 0$.

We can now derive the range of values for T_{CE} as follows.

Since $0 \leq A_i \leq 1 \quad \forall i = 1 \ldots D$, using equation 6.10 we can say

$$0 \leq ERR_{Cl} \leq D \tag{6.11}$$

Since the average ERR_{Cl} value is $D/2$, if $ERR_{Cl} \geq D/2$ the two clusters are more *different* according to attribute values than they are equal. Since we need the threshold value to identify cluster similarity, the maximum value for such a threshold can be $D/2$. Therefore

$$0 < T_{CE} < D/2 \tag{6.12}$$

Definition 6.6: Measure of Similarity Indicator

Since the similarity between two clusters depends on the T_{CE} value we define a new indicator called the *measure of similarity* which will indicate the *amount* of similarity when two clusters are considered to be similar. The measure of similarity indicator (I_s) is thus calculated as the fraction of the actual cluster error to the maximum tolerable error for two clusters to be considered similar.

$$I_s = 1 - \frac{ERR_{Cl}(Cl_j, Cl_k)}{Max(T_{CE})} \tag{6.13}$$

By substituting from equation 6.12 ,

$$I_s = 1 - \frac{ERR_{Cl}(Cl_j, Cl_k)}{D/2} \tag{6.14}$$

Considering two GSOMs $GMAP_1$ and $GMAP_2$, the cluster comparison algorithm can now be presented as :

1. Calculate $ERR_{Cl}(Cli, Clj) \quad \forall Cl_i \in GMAP_1$ with all $Cl_j \in GMAP_2$.

2. For each $Cl_i \in GMAP_1$, find $ERR_{Cl}(Cl_i, Cl_p), Cl_p \in GMAP_2$, such that

$$ERR_{Cl}(Cl_i, Cl_p) \leq ERR_{Cl}(Cl_i, Cl_j) \quad \forall Clj \in GMAP_2, \ p \neq j$$

3. Ensure that the clusters Cl_i, Cl_p satisfy the cluster similarity condition.

4. Assign Cl_p to Cl_i (as similar clusters) with the *amount of similarity* calculated as the measure of similarity value

$$1 - \frac{ERR_{Cl}(Cl_i, Cl_p)}{D/2}$$

5. Identify cluster Cl_i in $GMAP_1$ and Cl_j in $GMAP_2$ which have not been assigned to a cluster in the other GSOM.

The cluster comparison algorithm will provide a measure of the similarity of the clusters. If all the clusters in the two maps being compared have high measure of similarity values, then the maps are considered equal. The amount of similarity, or difference to be tolerated, will depend on the applications need. If there is one or more clusters in a map (say $GMAP_1$) which do not find a similar cluster in the other map (say $GMAP_2$), the two maps are considered different. The advantage of this comparison algorithm is not only for comparing feature maps for their similarity, but as a data monitoring method. For example, feature maps generated on a transaction data set at different time intervals may identify movement in the clusters or attribute values. The movement may start as a small value initially (the two maps have high similarity measure) and gradually increase (reduction of the similarity measure) over time. Such movement may indicate an important trend that can be made use of, or the starting of a deviation (problem) which needs immediate corrective action. As such monitoring the data by using the comparison method can be beneficial for data monitoring in a data mining system.

6.5 Experimental Results

The animal data set (Appendix A) used in the previous chapters is also used in this chapter to demonstrate the effect of the ACR model. Subsets of the animal

data set are selected for the experiments such that the rule extraction and the data change or movement identification can be demonstrated.

6.5.1 Rule Extraction from the GSOM Using the ACR model

For this experiment, 25 animals out of the 99 are selected from the animal data and these are, lion, wolf, cheetah, bear, mole, carp, tuna, pike, piranha, herring, chicken, pheasant, sparrow, lark, wren, gnat, flea, bee, fly, wasp, goat, calf, antelope, elephant and buffalo. The animals for this experiment has been selected such that they equally represent the groups insects, birds, non-meat eating mammals, meat eating mammals and fish. Some sub groupings exist within those main groups and these are shown in Figures 6.7 and 6.8. Figure 6.7 shows the GSOM for the 25 selected animals with SF=0.6. The clusters are shown with manually drawn boundary lines for easy visualisation.

Table 6.1 shows the mean ($mu_{A_{ij}}$) and standard deviation values for the attributes in the five clusters identified in Figure 6.7. Since we have used the animal data with mostly binary value attributes, and also since the number of data records considered are small, the standard deviation values are quite high overall. Therefore, when we refer to high or low standard deviation values, high or low values, with respect to those in the table have to be selected.

From the table it can be seen that some attributes in the clusters have high stan-

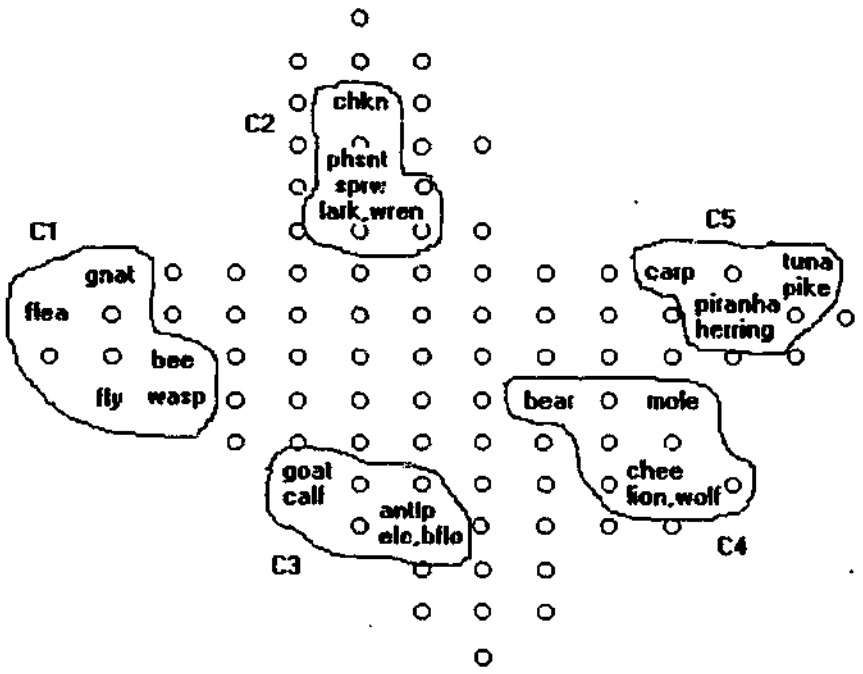


Figure 6.7: $GMAP_1$ - GSOM for the 25 animals with SF=0.6 with clusters separated by removing path segment with the weight difference=2.7671

dard deviation values. The high standard deviation values show that for that particular attribute, the input values are highly dispersed from the mean. As such the current average attribute value does not provide a representative value for the cluster. A solution to this problem would be the identification of the subgroups within the clusters and re-calculating the mean and standard deviation values. If the standard deviation values decrease compared with the previous values, then the subgrouping has made an improvement to the clustering. The analyst has to decide whether the increase in the work load of handling the larger number of clusters is sufficient compensation for the increase in accuracy.

It is also possible to generate rules from the current clusters, if the analyst is

Table 6.1: Average attribute values (Avg) and standard deviations (Std) for clusters in Figure 6.7

	C1		C2		C3		C4		C5	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
has hair	0.6	0.548	0	0	1	0	1	0	0	0
feathered	0	0	1	0	0	0	0	0	0	0
lay-eggs	1	0	1	0	0	0	0	0	1	0
feeds-milk	0	0	0	0	1	0	1	0	0	0
airborne	0.8	0.448	1	0	0	0	0	0	0	0
aquatic	0	0	0	0	0	0	0	0	1	0
predator	0	0	0	0	0	0	1	0	1	0
toothed	0	0	0	0	1	0	1	0	1	0
has-backbone	0	0	1	0	1	0	1	0	1	0
breathes	1	0	1	0	1	0	1	0	0	0
venomous	0.4	0.548	0	0	0	0	0	0	0	0
has-fins	0	0	0	0	0	0	0	0	1	0
no-of-legs	1	0	0.33	0	0.67	0	0.67	0	0	0
has-tail	0	0	1	0	1	0	0.8	0.448	1	0
domestic	0.2	0.448	0.2	0.448	0.4	0.548	0	0	0	0
big	0	0	0	0	1	0	0.8	0.448	0.4	0.548

satisfied with the level of clustering. In such a situation the analyst can select the attributes with high standard deviation values as *non-significant* for the groups. For the GSOM of Figure 6.7 attributes with the standard deviation values greater than 0.4 can be selected as non-significant, and as such the attributes *has hair, airborne, venomous* and *domestic* are non-significant for cluster 1. It is now possible to obtain the cluster description rule for cluster 1 from the model as :

IF (lay eggs, breathes, has legs) AND

NOT (have feathers, feed-milk, aquatic, predator,

toothed, has backbone, has fins, has tail, is big)

THEN Cluster = Ci

Similar rules can be generated for all the other clusters. These rules provide a description of the cluster such that the analyst can now label the cluster with an appropriate name according to the rule.

It is also possible to consider a situation where the analyst has some un-confirmed knowledge and/or has developed a hypothesis on some aspects of the data. The query by attribute rules discussed in section 6.3 provides a method of querying the data such that the uncertain knowledge or an hypothesis can be confirmed. For example, the analyst may *feel* that if an animal lays eggs, then it does not feed milk. Such a hypothesis can be confirmed by generating a query by attribute rule of the form :

1. IF lays-eggs THEN Cluster = ?

2. IF feed-milk THEN Cluster = ?

3. IF (lays-eggs AND feed-milk) THEN Cluster = ?

(1) and (2) above will confirm the existence of egg laying and milk drinking animals in the data while (3) will provide proof of any existence of animals who

satisfy both conditions. Therefore the results for the query using the ACR model of the GSOM in Figure 6.7 is shown below.

1. Cluster = 1, Cluster = 2, Cluster = 5.

2. Cluster = 3, Cluster = 4.

3. Cluster = NIL.

As such the hypothesis can be confirmed on the given data set. If the analyst feels that further sub clustering is required to obtain more *pure* clusters, this can be achieved by removing the next largest path segment in the data skeleton as described in Chapter 4. The resulting clusters are highlighted and labeled in Figure 6.8. Mean ($mu_{A_{ij}}$) and standard deviation values can now be calculated as shown in tables 6.2 and 6.3.

The standard deviation values have been reduced due to the finer clustering and as such it will be possible to obtain more accurate rules from the clusters. Therefore the data analyst has the opportunity of deciding on the level of clustering necessary for the application.

6.5.2 Identifying the Shift in Data values

The same set of data used in the previous experiment is used in this section to demonstrate the data shift identification with the ACR model. The GSOM of Figure 6.7 is considered as the map of the initial data and some additional data

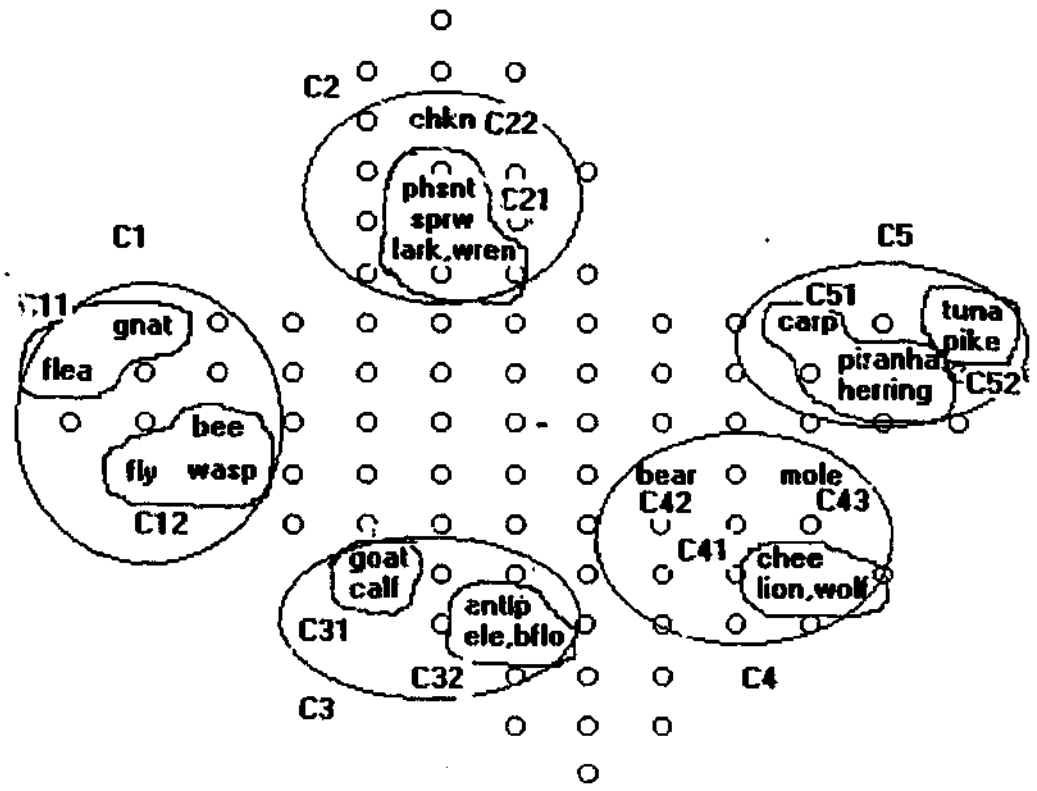


Figure 6.8: GSOM for the 25 animals with SF=0.6 with clusters separated by removing path segment with weight difference=1.5663

records (given below) are added to simulate a new data set for identification of any significant shifts in data.

Experiment 1 : Adding new data without the change in clusters

In the first experiment eight new animals are added to the earlier twenty five. The new animals are deer, giraffe, leopard, lynx, parakeet, pony, puma and reindeer. These animals were selected such that they belong to groups (clusters) that already exist in the initial data. As such the addition of these new records should not make any difference to the cluster summary nodes in the ACR model. Fig-

Table 6.2: Average attribute values (Av) and standard deviations (SD) for clusters in Figure 6.8

	C1.1		C1.2		C2.1		C2.2		C3.1		C3.2	
	Av	SD	Av	SD	Av	SD	Av	SD	Av	SD	Av	SD
has hair	0	0	1	0	0	0	0	0	1	0	1	0
feathered	0	0	0	0	1	0	1	0	0	0	0	0
lay-eggs	1	0	1	0	1	0	1	0	0	0	0	0
feeds-milk	0	0	0	0	0	0	0	0	1	0	1	0
airborne	0.5	0.71	1	0	1	0	1	0	0	0	0	0
aquatic	0	0	0	0	0	0	0	0	0	0	0	0
predator	0	0	0	0	0	0	0	0	0	0	0	0
toothed	0	0	0	0	0	0	0	0	1	0	1	0
has-backbone	0	0	0	0	1	0	1	0	1	0	1	0
breathes	1	0	1	0	1	0	1	0	1	0	1	0
venomous	0	0	0.67	0.58	0	0	0	0	0	0	0	0
has-fins	0	0	0	0	0	0	0	0	0	0	0	0
no-of-legs	1	0	1	0	0.33	0	0.33	0	0.67	0	0.67	0
has-tail	0	0	0	0	1	0	1	0	1	0	1	0
domestic	0	0	0.33	0.58	0	0	1	0	1	0	0	0
big	0	0	0	0	0	0	0	0	1	0	1	0

ure 6.9 shows the GSOM for the new data set with the additional 8 animals. Five clusters C1', C2'...C5' have been identified from $GMAP_2$. Table 6.4 shows cluster error values (calculated according to equation 6.10) between the GSOMs $GMAP_1$ (Figure 6.7) and $GMAP_2$ (Figure 6.9). Table 6.5 shows the clusters that have been identified as similar from $GMAP_1$ and $GMAP_2$ with the respective measures of similarity indicator values (I_s). It can be seen that the five clusters have not changed significantly due to the introduction of new data.

Table 6.3: Average attribute values (Av) and standard deviations (SD) for clusters in Figure 6.8 contd..

	C4.1		C4.2		C4.3		C5.1		C5.2	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
has hair	1	0	1	0	1	0	0	0	0	0
feathered	0	0	0	0	0	0	0	0	0	0
lay-eggs	0	0	0	0	0	0	1	0	1	0
feeds-milk	1	0	1	0	1	0	0	0	0	0
airborne	0	0	0	0	0	0	0	0	0	0
aquatic	0	0	0	0	0	0	1	0	1	0
predator	1	0	1	0	1	0	1	0	1	0
toothed	1	0	1	0	1	0	1	0	1	0
has-backbone	1	0	1	0	1	0	1	0	1	0
breathes	1	0	1	0	1	0	0	0	0	0
venomous	0	0	0	0	0	0	0	0	0	0
has-fins	0	0	0	0	0	0	1	0	1	0
no-of-legs	0.67	0	0.67	0	0.67	0	0	0	0	0
has-tail	1	0	0	0	1	0	1	0	1	0
domestic	0	0	0	0	0	0	0	0	0	0
big	1	0	1	0	0	0	0	0	1	0

Experiment 2 : Adding new data with different groupings

In the second experiment six different *birds* are added to the original 25 animals. The newly added birds are crow, duck, gull, hawk, swan, vulture. These birds are different from the birds that exist in the original 25 animal data set as some of them are *predators* and others being *aquatic* birds whereas these sub categories did not exist in the birds included with the initial 25 animals. The purpose of selecting these additional data is to demonstrate the data movement identification with the ACR model. Figure 6.10 shows the clusters C1', C2' .. C5' identified

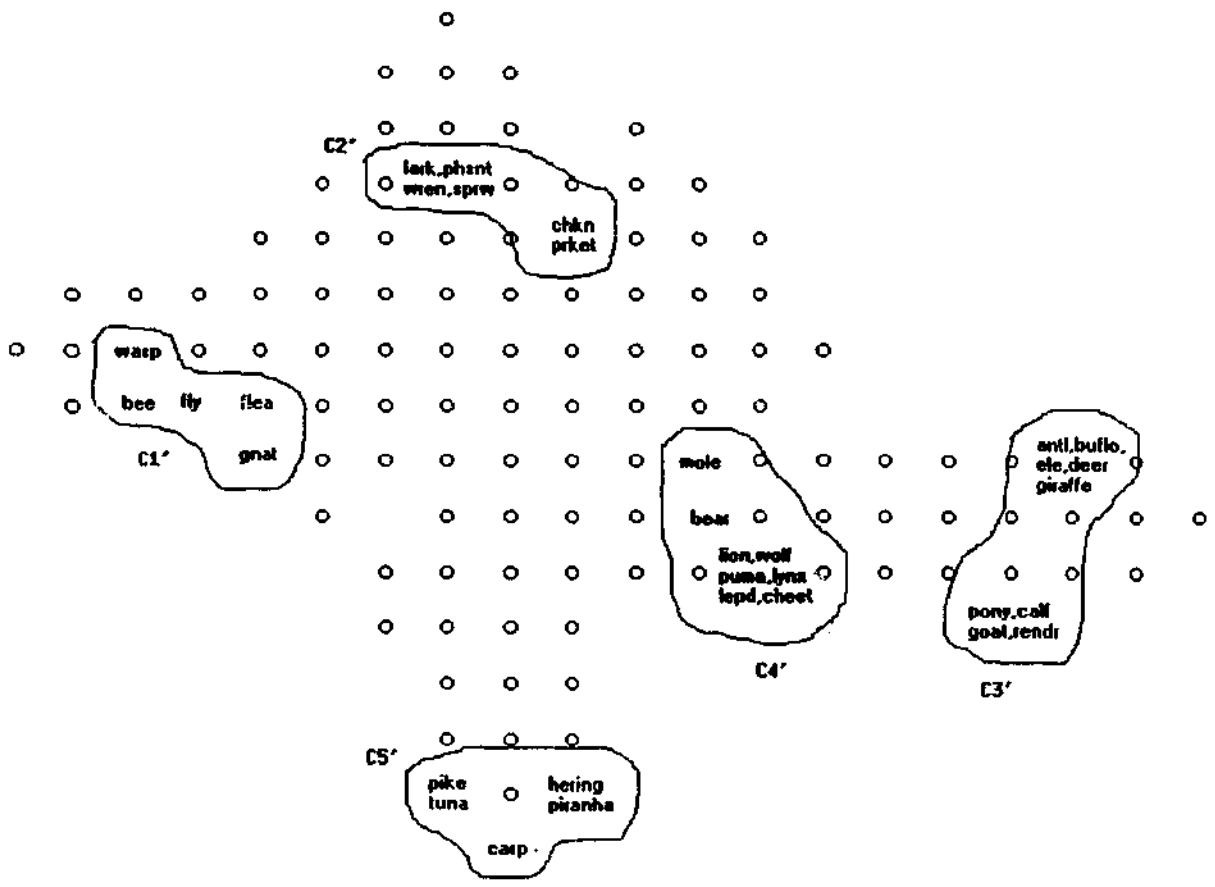


Figure 6.9: $GMAP_2$ - GSOM for the 33 animals with SF=0.6

with the new 31 animal data set.

The cluster error values between $GMAP_1$ and $GMAP_3$ are shown in table 6.6, and the best matching clusters are given in table 6.7, with the respective similarity indicator (I_s) values. From the I_s values, it can be seen that although clusters C2 and C2' are considered as similar there is a noticeable movement in the cluster. As such the data analyst can focus attention on this cluster to identify the reason for such change.

Table 6.4: The cluster error values calculated for clusters in Figure 6.9 with clusters in Figure 6.7.

	C1	C2	C3	C4	C5
C1'	0	4.87	8.13	8.73	10.4
C2'	5.00	0.13	7.40	8.67	8.07
C3'	8.17	7.58	0.04	1.84	8.71
C4'	8.88	8.54	1.65	0.15	7.27
C5'	10.4	7.93	8.67	7.27	0

Table 6.5: The measure of similarity indicator values for the similar clusters between $GMAP_1$ and $GMAP_2$

$GMAP_1$	$GMAP_2$	I_s
C1	C1'	1
C2	C2'	0.983
C3	C3'	0.995
C4	C4'	0.981
C5	C5'	1

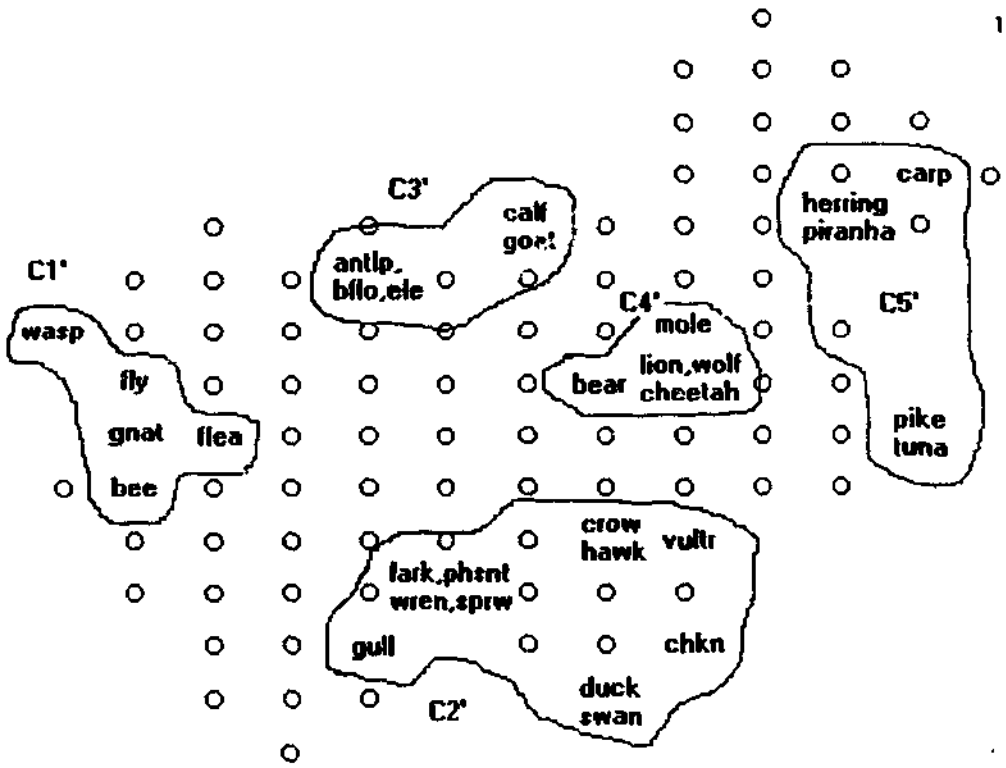


Figure 6.10: $GMAP_3$ - GSOM for the 31 animals with SF=0.6

6.6 Summary

In this chapter we have extended the GSOM by developing two layers of nodes on the output layer (map). The purpose of this extension is to exploit the advantages of using GSOM for data mining applications. Analysis of clusters obtained from SOMs has been traditionally carried out manually by data analysts. In this chapter we have proposed a model which can automate such a process and provide the analyst with a set of rules describing the data. Using the proposed model it is now possible for the data analyst to initiate queries on the map (of

Table 6.6: The cluster error values calculated for clusters in figure 6.9 with clusters in Figure 6.10.

	C1	C2	C3	C4	C5
C1'	0	4.87	8.13	8.73	10.4
C2'	5.79	0.93	8.10	8.15	7.01
C3'	8.13	7.56	0	1.8	8.67
C4'	8.73	8.56	1.8	0	7.27
C5'	10.4	7.93	8.67	7.27	0

Table 6.7: The measure of similarity indicator values for the similar clusters between $GMAP_1$ and $GMAP_3$

$GMAP_1$	$GMAP_3$	I_s
C1	C1'	1
C2	C2'	0.88
C3	C3'	1
C4	C4'	1
C5	C5'	1

the data) to confirm any hypothesis or un-confirmed knowledge.

Since the GSOM has been developed on the same unsupervised competitive learning concepts as the SOM, it inherits certain limitations that were inherent in the SOM. One such main limitation is the possibility of *different* maps for the same data when the data is presented in different order. Therefore it becomes impossible to accurately compare two maps to identify any changes that has occurred in the data. Since identifying changes or movement in data is one of the most useful functions in data mining, we have used the additional layers to enhance the GSOM with such ability.

The main contributions of this chapter can be summarised as follows :

1. Identification of the inability of comparing feature maps due to their differing shapes and placement of clusters, and introduce such limitation as a significant disadvantage for data mining.

2. Developed the Attribute Cluster Relationship (ACR) model as a solution to the problem in (1), by developing a conceptual layer on top of the GSOM.

3. Introduced a method of automating cluster analysis using feature map by developing a rule extraction method on the GSOM using the ACR model.

4. Identification of data shift monitoring as an important function in data

mining and the development of a method for such shift monitoring using the GSOM.

Chapter 7

Fuzzy GSOM-ACR model

7.1 Introduction

In 1965, Lotfi A. Zadeh [Zad65] proposed the theory of fuzzy logic, which is close to the way of human thinking. The main idea of fuzzy logic is to describe human thinking and reasoning with a mathematical framework. The main advantage of fuzzy logic is its *human like* reasoning capability, which makes it possible to describe a system using simple *if then* relations. Therefore it is possible to obtain a simpler *human understandable* solution, to a problem, reasonably quickly. In many applications, knowledge that describes desired system behaviour, or other useful information, is contained in data sets. Therefore the fuzzy system designer has to derive the *if then* rules from the data sets manually. A problem with such manual derivation is that it can be difficult and even impossible when the data

sets are large and complex.

When data sets contain knowledge or useful information, neural networks have the advantage because they can be *trained* using such data. A major limitation of neural networks is the lack of an easy way to interpret the knowledge learned during training. Therefore it can be seen that neural networks can learn from data while fuzzy logic solutions are easy to interpret and verify. Interest has been focussed on the combination of these two techniques for achieving the best of both worlds, which has resulted in the development of neuro fuzzy systems [Ber97a], [Kar96].

In chapter 6 we proposed the (Attribute Cluster Relationship) ACR model which can be used to generate rules regarding the data by using the clusters from the GSOM. The rules generated by the ACR models contain *crisp* values and as such may not represent a realistic view of actual situations. In this chapter we propose a method of extending the GSOM-ACR model such that *fuzzy* rules can be generated from the ACR model. We begin by interpreting the clusters identified with the GSOM as *fuzzy clusters*, similar to the fuzzy c-means method [Bez92]. The advantage of our method is that the number of clusters do not have to be pre-defined as in traditional fuzzy c-means, and as such our method is more suitable for data mining applications [Ala99a]. This extended fuzzy ACR model can be developed into a supervised data classification system and this idea is de-

scribed in chapter 8 under future work.

Section 7.2 of this chapter presents the basic theory of fuzzy sets and fuzzy logic and section 7.3 discuss the advantages of a fuzzy extension to ACR model by comparing the rules generated by the initial ACR with the possible fuzzy versions. Section 7.4 provides the functional equivalence between the GSOM clusters and fuzzy rules. Such functional equivalence is used in section 7.5 to describe the proposed extended fuzzy ACR model. Section 7.6 provides a summary for the chapter.

7.2 Fuzzy Set Theory

In this section we describe the basic fuzzy theory as a base for presenting the proposed fuzzy ACR model.

7.2.1 Fuzzy Sets

Fuzzy set theory is a generalisation of the traditional set theory. A classical *crisp* set is a collection of its members [Kec95]. The objects x of the universe X either are members (belongs to the set) or are not members of a given set A. The membership status of an object x can be expressed by a membership function $\mu_A(x)$, which is defined as

$$\mu_A(x) = 1, \qquad if x \in A$$

$$\mu_A(x) = 0, \qquad if x \notin A$$

The difference between fuzzy sets and crisp sets is that fuzzy sets can also have intermediate membership values [Yag94], [McN94]. In other words, the membership function $\mu_A(x)$ can have any value between one and zero:

$$\mu_A : X \rightarrow [0, 1] \tag{7.1}$$

Hence the object x can belong partially to a given fuzzy set A.

Classical set theory defines operations like intersection, union and complement between different sets. In fuzzy set theory, membership values are given by membership functions. Therefore, it is natural that fuzzy set operations are also defined by membership functions. The following basic operations are described in Zadeh's original paper [Zad65].

The *intersection* of two fuzzy sets A and B is defined as

$$\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\} \tag{7.2}$$

The *union* is defined as

$$\mu_{A \cap B}(x) = max\{\mu_A(x), \mu_B(x)\} \tag{7.3}$$

and the *complement* is defined as

$$\neg\mu_A(x) = 1 - \mu_A(x) \tag{7.4}$$

Often, the intersection operation is called *t-norm* (triangular norm), and the union is called the *t-conorm* (or s-norm). The minimum and the maximum operations are the most widely used.

7.2.2 Fuzzy Logic

Fuzzy set theory can easily be applied to logic. In fuzzy logic, logical sentences can have any truth value between zero and one, where zero corresponds to absolute false and one to absolute truth. Basically, the truth values of fuzzy logic are equivalent to the membership values of fuzzy set theory. Fuzzy logic is based on *linguistic variables*, which are characterised by a quintuple [Ngu97], [Dri96]:

$$\{x, T(x), U, G, M\} \tag{7.5}$$

where x is the name of variable, $T(x)$ is a set of names of *linguistic values* of x, G is a syntactic rule for generating the names of values of x, and M is a semantic rule which assigns the membership value $\mu_M(x)$ to the variable x in a fuzzy set of universe U.

Liniguistic variables are the building blocks of fuzzy logic. Different logical operations can be used to build logical sentences from them. There are several methods available and we describe one of these below.

The *intersection (and)* operation can be defined as

$$\mu_A(x) \wedge \mu_B(y) = min\{\mu_A(x), \mu_B(y)\} \tag{7.6}$$

The *union (or) operation* can be defined as

$$\mu_A(x) \vee \mu_B(y) = max\{\mu_A(x), \mu_B(y)\} \tag{7.7}$$

The *complement* can be defined as

$$\neg\mu_A(x) = 1 - \mu_A(x) \tag{7.8}$$

Other definitions of the logical operations can be used, but the above defined are the most common ones.

A fuzzy logic system is composed of several *if-then* fuzzy rules as :

if x is A_1 **and** y is B_1 **then** z is C_1,

$\vdots$

if x is A_n **and** y is B_n **then** z is C_n,

where (x is A_i) and (y is B_i) are the antecedent parts, and (z is C_i) are the conclusions. Based on the definition of the linguistic variable, each condition (x is A_i) can be interpreted as the membership value $\mu_{A_i}(x)$ of the variable x in the fuzzy set A_i.

Inference methods are used to determine the output of the fuzzy rules. *Max - min* composition is the most common inference method. We will also be using this method in section 7.5 for proposing our method. First the firing strengths of

the fuzzy rules (αi) are computed by combining the membership values together according to

$$\alpha_i = \mu_A(x) \wedge \mu_B(y) = min\{\mu_A(x), \mu_B(y)\} \tag{7.9}$$

where $i = 1, 2, \ldots, n$.

Next, the conclusion of the fuzzy rules ($\mu_{Ci'}(z)$) are computed according to

$$\mu_{Ci'}(z) = \alpha_i \wedge \mu_{Ci}(z) = min\{\alpha_1, \mu_{Ci}(z)\} \tag{7.10}$$

Then the conclusions are combined together according to

$$\begin{aligned} \mu_{Ci}(z) &= \mu_{C1'}(z) \vee \mu_{C2'}(z) \vee \ldots \vee \mu_{Cn'}(z) \\ &= max\{\mu_{C1'}(z), \mu_{C2'}(z), \ldots, \mu_{Cn'}(z)\} \end{aligned} \tag{7.11}$$

The outputs from the fuzzy rules form a combined fuzzy set C'. The fuzzy set C' is finally transformed into a crisp output value. This is called *defuzzification* of the output. *Center of area* (of membership function) is the mostly used defuzzification method, which can be derived as

$$z^* = \frac{\int_Z z \cdot \mu_{C'}(z) dz}{\int_Z \mu_{C'}(z) dz} \tag{7.12}$$

7.3 The Need and Advantages of Fuzzy Interpretation

The ACR model is described in chapter 6. This model is then used to generate rules from the GSOM. As explained in chapter 6, the rule generation method

is developed for interpreting the knowledge *learnt* by the neural network, thus removing the conventional *black box* limitation.

We have proposed two different types of rules for the ACR model. All these rule types have been generated on the assumption that the clusters are *crisp*, by defining threshold values for rule generation. The assumption of crisp clusters result in the following limitations.

1. Such crisp clusters will need to have uniform density of input data distribution inside the cluster and zero density outside. It is generally more realistic to assume higher density closer to the cluster center with decreasing density towards the borders.

2. Providing crisp values as the threshold may result in unfair *cutting off* of some useful information. For example, forcing the threshold for *tall men* at six foot two inches may unfairly judge six foot one inch as medium or short.

3. Human beings find it easier to deal with fuzzy values than crisp values. For example, using query by attribute rules, it is more *human like* to look for *hot, humid* climate than to decide on threshold values such as hot $\Longrightarrow \geq 35$ centigrades, and humid $\Longrightarrow \geq 80\%$.

Examples

In this section we describe the advantages of fuzzy rules over similar crisp rules. The 3 types of rules generated using the crisp ACR model is considered as exam-

ples and their fuzzy counterparts (with the proposed fuzzy model) are presented and advantages discussed in this section.

Example 1

Cluster description rule is described in chapter 6. Such a rule can be presented as

$$IF\ (A_1 > C')\ AND\ (A_2 < C'')\ THEN\ Cluster = Cl_k$$

where C' and C'' are singletons representing average values of the attributes A_1 and A_2. By defining fuzzy membership functions for the clusters, the above rule can be presented as

$$IF\ (A_1\ is\ U_{i,1})\ AND\ (A_2\ is\ U_{i,2})\ THEN\ Cluster = Cl_i$$

where U_1 and U_2 are the respective membership functions of attributes A_1 and A_2 in cluster i. The advantage of the fuzzy rule is that, instead of giving a single crisp value for the attribute values in the cluster, a membership function is given, thus providing a more realistic interpretation.

Example 2

A type 1 query by attribute according to chapter 6 can be,

What are the clusters where attribute A_1 is reported as large.

With the crisp method described in chapter 6 and assuming that threshold for large = 0.8, we search for $(A_1 > 0.8)$ and may obtain the rule as

$$IF\ (A_1 > 0.8)\ THEN\ Cluster = Cl_i$$

With the fuzzy interpretation, *large* is meaningful to the system and the results of the same query can be of the form

$$IF A_1 \ is \ large \ THEN \ Clusters = Cl_i \ OR \ Cl_j$$

By not using a crisp threshold, we do not leave out some clusters which do not satisfy the threshold condition, but still can be considered as *good candidates* for our requirements. As such, fuzzy querying relieves the data analyst from the burden of having to *know* or *decide* on exact (crisp) thresholds (which is difficult or even impossible in many situations in data mining applications).

Example 3

As described in chapter 6, a type 2 query by attribute can be expressed as

Provide the related attributes for small values of A_1.

With the crisp method the resulting rule can be expressed as

$$IF \ (A_1 < 0.3) \ THEN \ (A_3 = 9.2) \ AND \ (A_4 = 2.3) \ IN \ Cluster = Cl_i$$

where *small* has been expressed as a threshold value < 0.3 by the user (analyst). With the fuzzy extension, the result can be of the form

$$IF \ A_1 \ is \ small \ THEN \ (A_3 \ is \ large) \ (A_4 \ is \ small) \ IN \ Cluster = Cli_i$$

The fuzzy output provides a more generalised view, which is more human understandable than the crisp threshold values.

It can be seen that the fuzzy rules provide an abstract and *human understandable* interpretation of the information stored in the GSOM clusters. The *crisp* and *fuzzy* methods can even be used to complement each other by the data analyst. The next section provides the proof of functional equivalence between fuzzy rules and GSOM clusters and section 7.5 uses such equivalence to propose the Fuzzy ACR model.

7.4 Functional Equivalence Between GSOM Clusters and Fuzzy Rules

As a first step in developing a Fuzzy ACR model, we have to prove that the clusters of GSOMs can be interpreted as fuzzy rules. We use a method proposed by Halgamuge et. al. [Hal95] as a base for our work, to prove the functional equivalence of the cluster summary nodes of the ACR model to fuzzy rules.

Nearest prototype classifiers such as fuzzy c - means algorithm can be used to produce a set of prototype vectors from an input data set [Bez93]. Once the prototypes are found, they can be used to define a classical nearest prototype classifier to classify the feature vectors $x \in C_i$ as

$$|x - w| \leq |x - w_j| \quad where\ 1 \leq j \leq k \tag{7.13}$$

where k is the number of nearest prototypes, C_i is one of the l class labels and the weight w is the nearest prototype from all the vectors w_j. Since the cluster

summary nodes in the ACR model represent the k clusters identified from the GSOM, they can be described as nearest prototypes for the k clusters. As such, for developing the fuzzy ACR model, we need to interpret the cluster summary nodes as fuzzy rules.

The inference process is to decide to which class C_i, (where $1 \leq x \leq k$) an input vector x belongs to. It has to be decided whether $x \in C_i$, i.e. Cluster C_i (cluster summary node in the case of the ACR model), has to be found such that the reference average weight vector $w_j \in C_i$ shows the least deviation (minimum distortion) with x, considering average weight vector values of all the cluster summary nodes in the ACR model. Using equation 7.13, such w_j can be interpreted as,

$$\begin{aligned} &\Rightarrow \quad Min|x - w_j| \quad for \quad 1 \leq j \leq k \\ &\Rightarrow \quad Max(-|x - w_j|) \quad for \quad 1 \leq j \leq k \\ &\Rightarrow \quad Max(e^{-|x-w_j|}) \quad for \quad 1 \leq j \leq k \end{aligned} \tag{7.14}$$

For Euclidean distance as the distance measure in equation 7.14,

$$\begin{aligned} &\Rightarrow \quad Max(e^{-\sum_{d=1}^{n}(x-w_{j,d})^2}) \quad for \quad 1 \leq j \leq k \\ &\Rightarrow \quad Max \prod_{d=1}^{D}(e^{-(x-w_{j,d})^2}) \quad for \quad 1 \leq j \leq k \end{aligned} \tag{7.15}$$

where $1 \leq d \leq D$ is an index for input dimensions.

Consider the product as T-norm (intersection), and singletons as consequent

membership functions from the fuzzy system point of view. Also the defuzzification can be avoided since we are considering classification. We can represent a fuzzy rule for this case as

decide whether $x \in C_i$

$$\Rightarrow \quad \mathrm{OR}(\mathrm{AND}(\mu(\mathrm{x_d})))$$

$$\Rightarrow \quad \mathrm{Max} \prod_j (\mu_{j,d}(\mathrm{x_d})) for 1 \leq j \leq k \tag{7.16}$$

By comparing (7.16) with (7.15) it can be seen that these equations are similar when

$$\mu_{j,d}(\mathrm{I_d}) \quad = \quad e^{-(\mathrm{I_d}-\mathrm{W_{j,d}})^2} \tag{7.17}$$

Therefore it can be concluded that classifier type of fuzzy systems with product inference and maximum composition are equivalent to the nearest prototype classifier with Euclidean distance, if the antecedent membership functions $\mu_{j,d}$ of each prototype neuron j is selected as a Gaussian function, or modified exponential function respectively.

7.5 Fuzzy ACR model

The (Attribute Cluster Relationship) ACR model is described in chapter 6, and its usefulness as a base for rule generation is demonstrated. In section 7.3 of this chapter we discussed the advantages of generating fuzzy rules compared with the traditional *crisp* rules. In this section we propose a method for extending the

initial ACR model to extract fuzzy rules, thus furthering its current usefulness. The main steps in extending the ACR model is as follows.

1. Define the cluster summary nodes in ACR model as fuzzy rules.

2. Develop membership functions for each of the data attributes (dimensions) for each cluster.

3. Project the membership functions to the respective attribute value ranges to develop *unsupervised* categorisations of data attributes.

4. Generate rules using the fuzzy membership functions and the categorisations such that they provide a more abstract method of representing the data.

The above steps are described in detail below.

7.5.1 Interpreting Cluster Summary Nodes as Fuzzy Rules

In section 7.4 we have shown that a nearest prototype vector is functionally equivalent to a fuzzy rule. In the ACR model, the cluster summary nodes represent the clusters in the data, and representative values such as the mean and the standard deviation for the cluster calculated for the summary node. Therefore we consider the summary node as the nearest prototype for the input vectors that are assigned to that particular cluster. Therefore the cluster summary nodes can be

considered as functionally equivalent to fuzzy rules and we represent such nodes with the following form of fuzzy rule.

$$if\ x_1\ is\ U_{i,1}\ and\ x_2\ is\ U_{i,2}\ and \ldots and\ x_n\ is\ U_{i,n}\ then\ y\ is\ a_i \qquad (7.18)$$

where $x = [x_1, x_2, \ldots, x_n]$ is the input and y is the output of the node identifying the cluster. Variables $U_{i,j}, j = 1..n$ are the fuzzy sets for the n dimensions of the i^{th} cluster. Note that the fuzzy sets are not shared by the fuzzy rules, but rather each fuzzy rule (cluster) has a set of membership functions (fuzzy sets) associated with each input.

Figure 7.1 shows the extended fuzzy ACR model. Each cluster summary node has now become a fuzzy rule (R_1, R_2, R_3), which contain a multi dimensional (equal to the dimensionality of the data) fuzzy membership function $\mu_{i,j}$ where i is the cluster (number) identifier, and $j = [1..D]$ represents the data dimensions. It is now necessary to develop the membership functions for each cluster, and such a method is proposed in the next section.

7.5.2 Development of the Membership Functions

Once the nearest prototypes (cluster summary nodes) have been identified and assigned with fuzzy rules, it is necessary to provide values for the respective fuzzy membership functions ($\mu_{i,j}$). There are several types of such functions and we will describe the generation of triangular type of membership functions.

Figure 7.2 shows a triangular membership function for an input attribute (di-

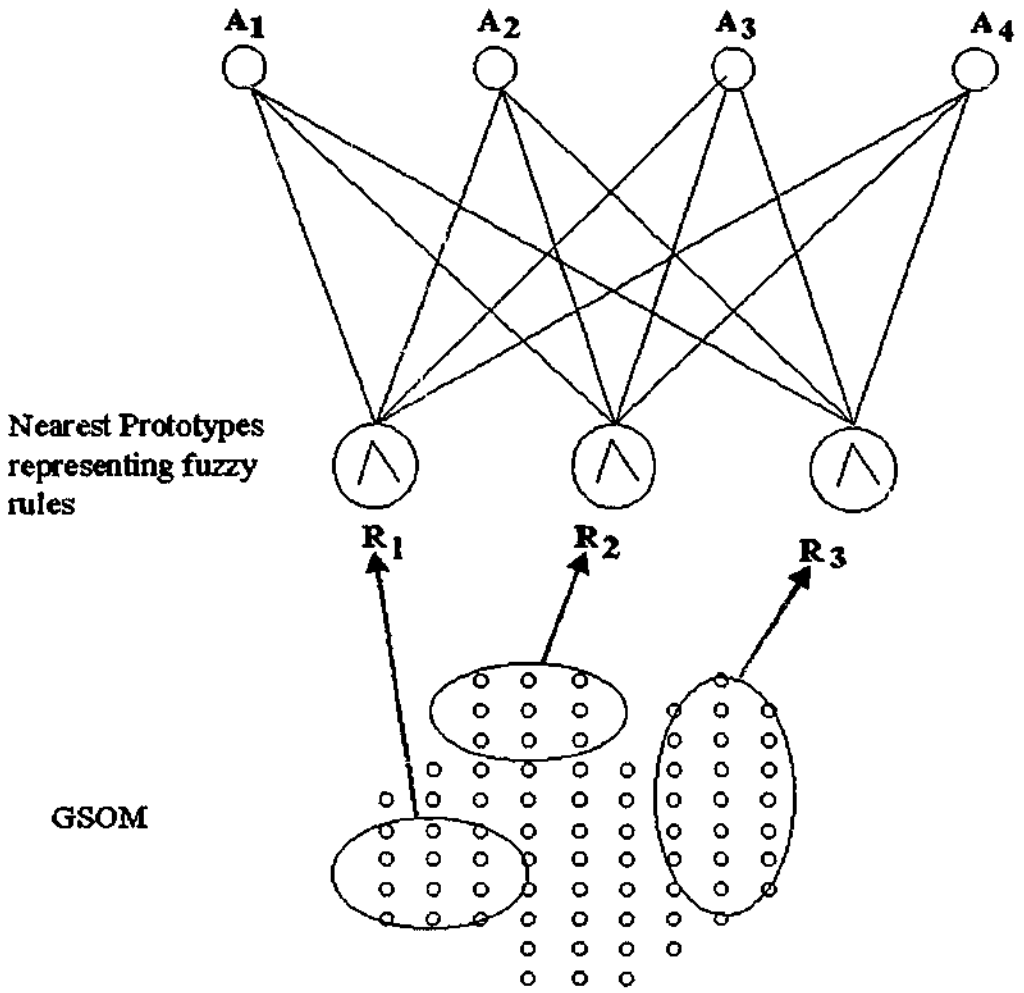


Figure 7.1: The Fuzzy ACR model

mension) j of cluster i. The membership value $\mu_{U_{i,j}}(x_j)$ of the input signal x_j in the fuzzy set $U_{i,j}$ is given by the triangular membership function in Figure 7.2. Variable $c_{i,j}$ is the center of the fuzzy set $U_{i,j}$ and variables $l_{i,j}$ and $r_{i,j}$ are the left and right spreads of the same set respectively. Since each cluster summary node represents a multi dimensional membership function (or a fuzzy rule), we will show them separately for easy visualisation (Figure 7.3). Evanthough the membership functions are simplified as triangles, the exact shape is determined

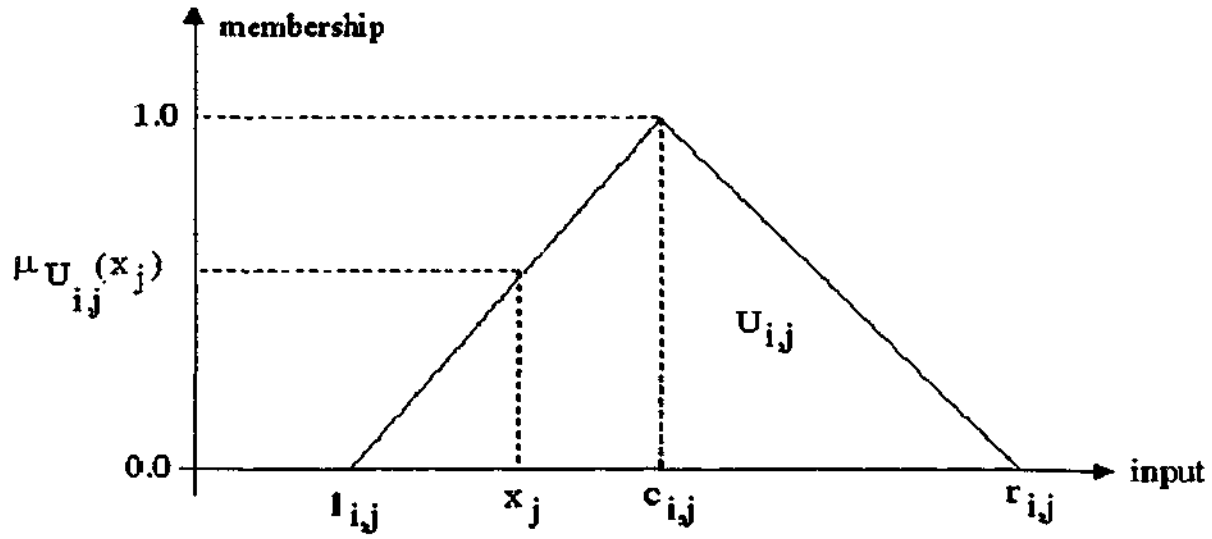


Figure 7.2: A triangular membership function

by the *distance* selected.

Each condition (x_j *is* $U_{i,j}$) in the fuzzy rule is interpreted as the membership value $\mu_{U_{i,j}}$. To obtain the membership values, the triangular membership function can be defined as follows.

$$\begin{aligned}
\mu_{U_{i,j}}(x_j) &= \frac{x_j - l_{i,j}}{c_{i,j} - l_{i,j}}, \quad if \;\; l_{i,j} \leq x_j \leq c_{i,j}, \\
\mu_{U_{i,j}}(x_j) &= \frac{x_j - r_{i,j}}{c_{i,j} - r_{i,j}}, \quad if \;\; c_{i,j} \leq x_j \leq r_{i,j}, \\
\mu_{U_{i,j}}(x_j) &= 0 \quad Otherwise
\end{aligned}$$

For the calculation of the membership values, $c_{i,j}, l_{i,j}$ and $r_{i,j}$ are assigned values as follows.

$c_{i,j}$ = mean value of attribute j in cluster i.

$l_{i,j}$ = minimum value of attribute j in cluster i.

$r_{i,j}$ = maximum value of attribute j in cluster i.

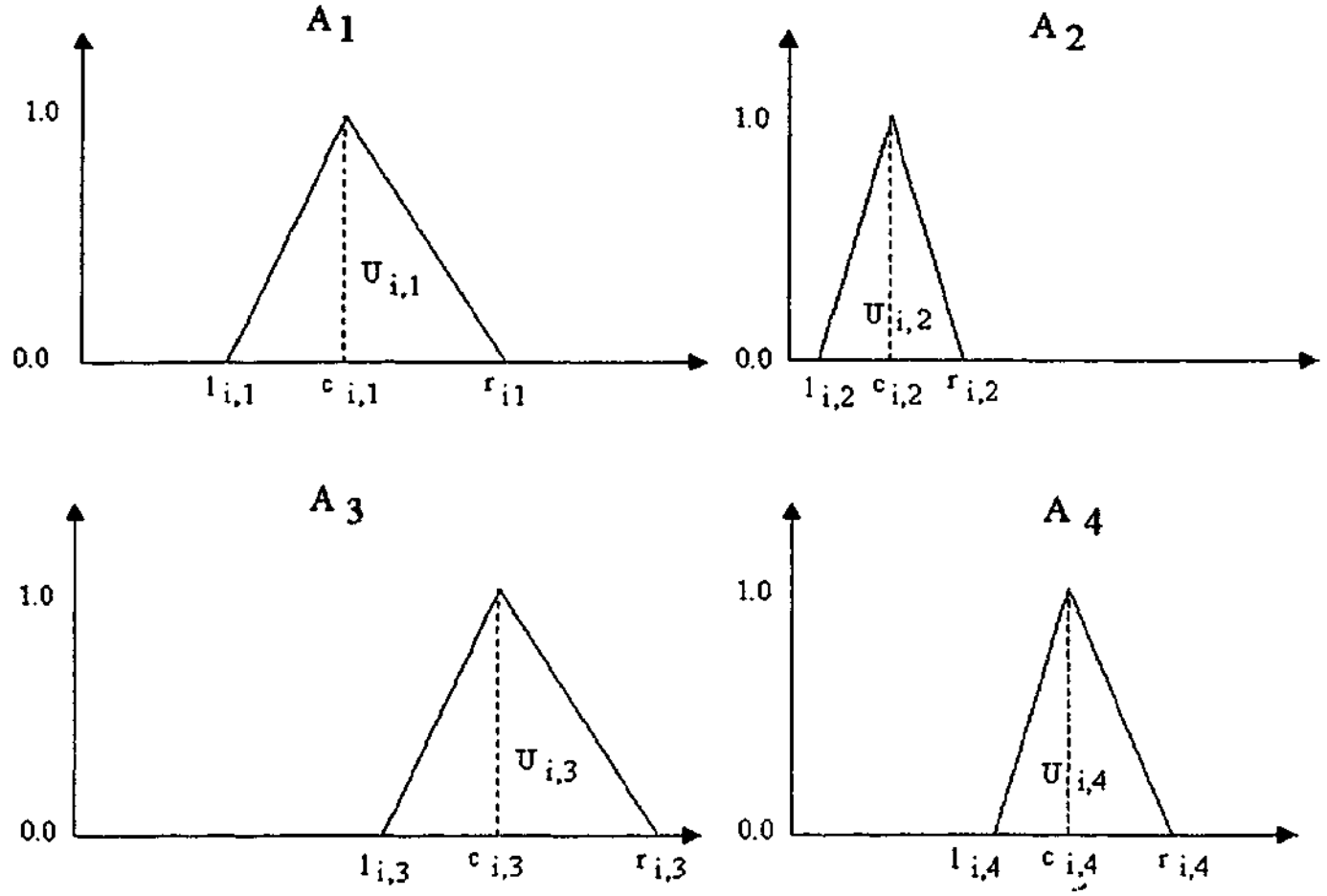


Figure 7.3: The membership functions for cluster i

Another method for calculating the $l_{i,j}, r_{i,j}$ values is to take a fixed distance from the cluster center in *plus* and *minus* directions [Vuo94].

7.5.3 Projecting Fuzzy Membership Values to the Attribute Value Range

Once the membership functions have been defined and calculated, these functions are used to generate the *categories* (or classes) for each input dimension. The advantage of generating such categories using the membership functions is

that such categories will be unbiased from external *opinions*. Such categorisation may also result in the identification of unknown aspects about the attributes.

For example, assume that attribute A_1 is conventionally categorised into small, medium and large. By generating the membership functions, it may be apparent that there are four identifiable categories in A_1. Therefore the analyst may decide to re-define the value range of attribute A_1 into four categories as, *small, medium large* and *very large*. The importance and *advantage* of the new categorisation is that it is built using the unbiased clusters from the GSOM. Therefore can be very useful in identifying unforeseen patterns in data.

Figure 7.4 shows the category projection of the membership functions of attribute A_1, for all the clusters with a significant representation of A_1. When such clearly

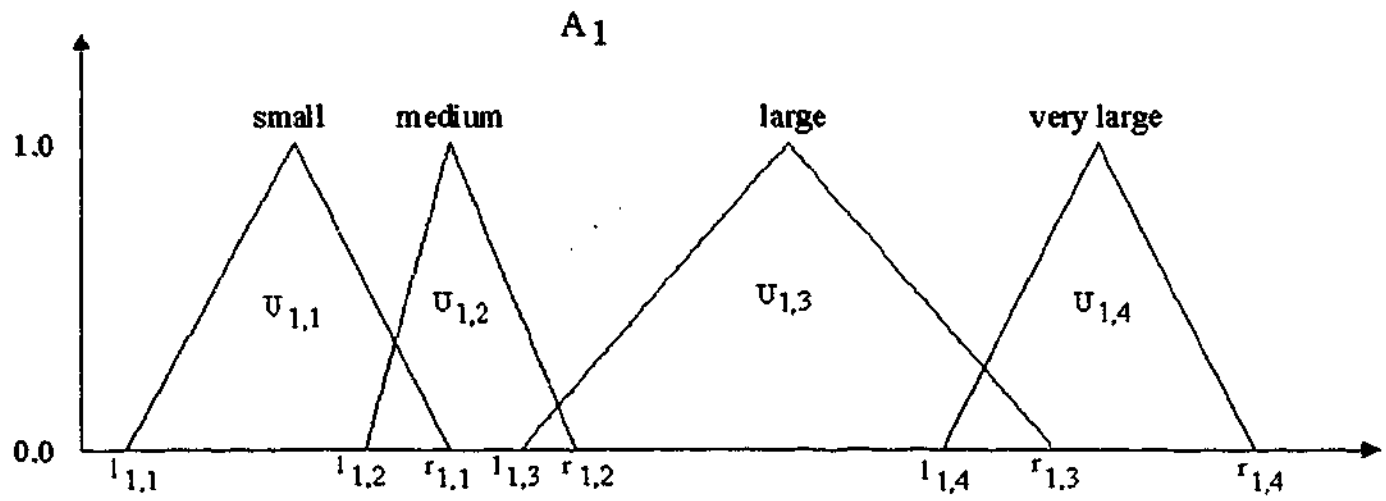


Figure 7.4: Projection of the fuzzy membership functions to identify categorisation

separate membership values are present the analyst can identify these as separate categories. It may also be possible that two or more membership values project on top of each other. Such a result will show that the same input value regions of attribute A_1 participate in two or more clusters. Therefore the study of such categories itself provide the data analyst with more information regarding the data. For example, if attribute A_k is represented in all the clusters in equal strength, then A_k is not *contributing* to the clustering.

Therefore the advantage of generating such categories are

1. It produces an unbiased categorisation of the attribute value range. Where categorisation is already present, it can be compared to confirm whether any unforeseen categories exist.

2. Provide an abstract fuzzy method of querying the data instead of using the crisp values as discussed previously in chapter 6.

7.6 Summary

Chapter 6 of this thesis presented an Attribute Cluster Relationship (ACR) model, which provides a base for generating rules describing the data. Such rule generation serves as a method for removing the *black box* limitation of conventional neural networks. The ACR model uses *crisp* values to represent knowledge regarding the data and their relationships, and as such the rules generated are

also crisp rules. Such rules contain specific values and therefore it is difficult to arrive at a more general opinion regarding the data. The concept is fuzzy logic and fuzzy sets provide a method of fuzzifying the *crisp* values such that the knowledge can be generalised and provide a more abstract view.

In this chapter we proposed a method for extending the ACR model using the concepts of fuzzy logic thus developing a Fuzzy ACR model. The new extended model can be used to obtain *fuzzy* versions of the cluster description rules and query by attribute rules, which can be considered as more generalised presentation of the data relationships. A main advantage of our model is that it builds the fuzzy rules from unsupervised clusters, and as such do not contain any pre-conceived bias in building the rules. The Fuzzy ACR model can be further extended into a fuzzy classifier system by considering the ACR model as two layers of a Radial Basis Function (RBF) network. We describe such possible extensions in the next chapter.

Chapter 8

Conclusion

The thesis has focused on developing a novel structure adapting neural network model which has special significance for data mining applications. Pre-defined fixed structure is a major limitation in traditional neural network models which is a barrier in the quest for developing *intelligent* artificial systems. The research work described is not only the development of a new neural network model with significant advantages for data mining, but also as a step towards breaking the *pre-definable structure* barrier in neural networks. In this final chapter we present a summary of the research achievement in the thesis work and suggests some areas and problems which have emerged from our work, with potential for further research and expansion.

8.1 Summary of Contributions

The major contributions in this thesis can be divided in to five parts and they are

1. the development of the Growing Self Organising Map,
2. data skeleton modeling and automated cluster separation from the GSOM,
3. hierarchical clustering using the GSOM,
4. development of a conceptual model for rule extraction and data movement identification,
5. fuzzy rule generation using the extended GSOM.

Contributions in each of these parts are described in the following subsections.

8.1.1 Growing Self Organising Map (GSOM)

The main contribution of this thesis is the development of the new neural network model called the GSOM. The GSOM encapsulates several new features to provide it with the incrementally generating nature while preserving the advantages of the traditional SOM. The new features introduced in the GSOM are

1. A new formula for calculating the rate of weight adaptation (learning rate) which considers the incrementally generating nature of the network. As such the number of nodes of a partially generated network is considered in the learning rate calculation.

2. A heuristic criteria for generating new nodes is introduced. The new node generation is restricted to the boundary of the network thus always maintaining a two dimensional map.

3. Initialisation of the weights of the newly generated nodes (during network growth), to *fit in* with the existing partially *organised* weights. The new weight initialisation method introduces already *ordered* weights and hence localised weight adaptation is sufficient for convergence.

4. A new method of *error distribution* from non-boundary nodes to maintain proportional representation of the data distribution with in the *area* of the GSOM.

5. A new parameter called the spread factor for providing the data analyst with control over the *level of spread* of a GSOM.

Due to these new features, the GSOM has the following advantages over the traditional feature maps.

1. Due to its self generating ability, network designer is relieved of the difficult (sometimes impossible) task of predefining an optimal network structure for a given data set.

2. A major limitation of traditional feature maps is the problem of *oblique orientation* which results in distorted representations of the input data structure. Oblique orientation is a result of pre-defined fixed structure, and as such does not occur with the GSOM.

3. The incremental network generation results in the *self organisation* of the shape of the network as well as the node weights. Therefore the GSOM has an *input driven* shape, which highlights the data clusters by the shape of the network itself, thus providing better visualisation.

4. The new weight initialisation method initialises *ordered* new weights, thus reducing the chance of *twisted maps* which is a major limitation of traditional SOMs. The ordered initial weights remove the requirement of an *ordering phase*, as in the randomly initialised SOM, and thus the GSOM achieves convergence with a lesser number of iterations compared to the SOM.

5. The spread factor provides the analyst with flexibility of generating representative maps at different levels of spread according to the needs of the application. With traditional SOM, such control can only be attempted by changing the size and shape of the two dimensional network of nodes. We have shown that this conventional method results in the data clusters being forced into the user defined network shape, thus providing a distorted picture.

6. It has been experimentally shown that the GSOM provides maps with similar spread using lesser number of nodes compared to the SOM. Hence lesser amount of computing resources are required for the GSOM.

8.1.2 Data Skeleton Modeling and Automated Cluster Separation

In applications using conventional feature maps, clusters are visually identified, which can result in errors and inaccuracies. Visual cluster identification can also become a problem when building automated systems. Therefore we have developed a method for automating the cluster identification process from the GSOM. The incrementally generating nature of the GSOM facilitates the building of a data skeleton (chapter 4). The data skeleton provides a visualisation of the *paths* along which the network *grew* thus providing further insights into the structure of the input data. The data skeleton is also used to develop a method of automated identification/separation of clusters, which is traditionally considered as a visualisation task.

8.1.3 Hierarchical Clustering using the GSOM

Hierarchical clustering is an useful tool in data analysis. In this thesis a method for hierarchical clustering of the GSOM is proposed using the control provided by the spread factor (chapter 5). Such hierarchical clustering can be used to obtain an initial *abstract overview* of the data, and further detailed analysis can be carried out only on selected *interesting* regions. Hierarchical clustering also facilitates working with large data sets since the analysis can be conducted separately on different regions of the data.

8.1.4 Development of a Conceptual Model for Rule Extraction and Data Movement Identification

Feature maps may develop clusters in different *positions* of the network according to the order of input presentation. Therefore it becomes difficult to compare maps to identify similarity or differences in clusters. We have developed a conceptual model of the GSOM called the Attribute Cluster Relationship (ACR) model which provides a conceptual and summarised view of the data (chapter 6). The ACR model further facilitates the following,

1. Extraction of descriptive rules of the GSOM clusters.

2. A new method for querying the database with *unconfirmed* knowledge or hypothesis and generate *query by attribute rules* for confirming the hypothesis.

3. Method for identifying *movement* and *change* in data which can provide a data analyst with information on new trends.

8.1.5 Fuzzy Rule Generation

A new method is proposed for extending the Attribute Cluster Relationship (ACR) model into a fuzzy rule generating system. The main advantage of such a system is the ability to interpret the clusters more *realistically* by interpreting the GSOM clusters as fuzzy rules and identifying fuzzy membership functions.

Such membership functions also provide the analyst with the ability to query the data at a more abstract level.

8.2 Future Research

Several interesting areas of future research has opened up from the work described in this thesis. Analysis and optimisation of the traditional SOM parameters is still being carried out. Similar work on the GSOM needs to be conducted to enhance its advantages. We have shown that the self generating structure and the use of spread factor results in a *better* representation of the data clusters. There is potential for a rigorous mathematical analysis of this phenomena to calculate the optimal parameters for better performance.

A very useful extension of the GSOM-ACR model will be an on-line data monitoring system which can automatically update and report the changes in the data. A method for incrementally updating the ACR model is needed to continuously monitor the movement in the data. Such a system has vast potential for commercial organisations since it can not only identify trends and changes in the data for competitive advantage but also provide *warnings* of potentially *dangerous* situations before their actual occurrence.

The proposed fuzzy extension to the ACR model can also be extended to derive a fuzzy classification system. The cluster summary nodes and their fuzzy mem-

bership functions can be considered as equivalent to the hidden layer of a Radial Basis Function (RBF) network [Hal95] or a FuNe1 neuro-fuzzy model developed by Halgamuge et. al. [Hal97]. These models are traditionally used for supervised fuzzy classification, and a hidden layer self generated with the GSOM can provide unbiased set of clusters for further supervised fine tuning.

Finally, the concept of adaptable structures provide the neural networks with higher level of intelligence, due to the less human dependence. All conventional neural network models can benefit from such intelligence, which has the potential to be the next major development in artificial neural network research.

Appendix A

The Animal Data Set

Animal Name	has hair	feathered	lay eggs	feed milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	has fins	legs	has tail	domestic	catsize	type
aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
buffalo	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
calf	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
carp	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	4
catfish	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
cavy	1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	1
cheetah	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
chicken	0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	2
chub	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4

Animal Name	has hair	feathered	lay eggs	feed milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	has fins	legs	has tail	domestic	catsize	type
clam	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	7
crab	0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	7
crayfish	0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	7
crow	0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	2
deer	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
dogfish	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	4
dolphin	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	1	1
dove	0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	2
duck	0	1	1	0	1	1	0	0	1	1	0	0	2	1	0	0	2
elephant	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
flamingo	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	1	2
flea	0	0	1	0	0	0	0	0	0	1	0	0	6	0	0	0	6
frog	0	0	1	0	0	1	1	1	1	1	1	0	4	0	0	0	5
fruitbat	1	0	0	1	1	0	0	1	1	1	0	0	2	1	0	0	1
giraffe	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
gnat	0	0	1	0	1	0	0	0	0	1	0	0	6	0	0	0	6
goat	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
gorilla	1	0	0	1	0	0	0	1	1	1	0	0	2	0	0	1	1
gull	0	1	1	0	1	1	1	0	1	1	0	0	2	1	0	0	2
haddock	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0	0	4
hamster	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	0	1
hare	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	0	1
hawk	0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	2
herring	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
honeybee	1	0	1	0	1	0	0	0	0	1	1	0	6	0	1	0	6
housefly	1	0	1	0	1	0	0	0	0	1	0	0	6	0	0	0	6

Animal Name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	type
kiwi	0	1	1	0	0	0	1	0	1	1	0	0	2	1	0	0	2
ladybird	0	0	1	0	1	0	1	0	0	1	0	0	6	0	0	0	6
lark	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2
leopard	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
lion	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
lobster	0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	7
lynx	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
mink	1	0	0	1	0	1	1	1	1	1	0	0	4	1	0	1	1
mole	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	0	1
mongoose	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
moth	1	0	1	0	1	0	0	0	0	1	0	0	6	0	0	0	6
newt	0	0	1	0	0	1	1	1	1	1	0	0	4	1	0	0	5
octopus	0	0	1	0	0	1	1	0	0	0	0	0	8	0	0	1	7
opossum	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	0	1
oryx	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
ostrich	0	1	1	0	0	0	0	0	1	1	0	0	2	1	0	1	2
parakeet	0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	2
penguin	0	1	1	0	0	1	1	0	1	1	0	0	2	1	0	1	2
pheasant	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2
pike	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	4
piranha	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
pitviper	0	0	1	0	0	0	1	1	1	1	1	0	0	1	0	0	3
platypus	1	0	1	1	0	1	1	0	1	1	0	0	4	1	0	1	1
polecat	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
pony	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
porpoise	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	1	1
puma	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
pussycat	1	0	0	1	0	0	1	1	1	1	0	0	4	1	1	1	1
raccoon	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
reindeer	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
rhea	0	1	1	0	0	0	1	0	1	1	0	0	2	1	0	1	2
scorpion	0	0	0	0	0	0	1	0	0	1	1	0	8	1	0	0	7

Animal Name	has hair	feathered	lay eggs	feed milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	has fins	legs	has tail	domestic	catsize	type
seahorse	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0	0	4
seal	1	0	0	1	0	1	1	1	1	1	0	1	0	0	0	1	1
sealion	1	0	0	1	0	1	1	1	1	1	0	1	2	1	0	1	1
seasnake	0	0	0	0	0	1	1	1	1	0	1	0	0	1	0	0	3
seawasp	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	7
skimmer	0	1	1	0	1	1	1	0	1	1	0	0	2	1	0	0	2
skua	0	1	1	0	1	1	1	0	1	1	0	0	2	1	0	0	2
slowworm	0	0	1	0	0	0	1	1	1	1	0	0	0	1	0	0	3
slug	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7
sole	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0	0	4
sparrow	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2
squirrel	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	0	1
starfish	0	0	1	0	0	1	1	0	0	0	0	0	5	0	0	0	7
stingray	0	0	1	0	0	1	1	1	1	0	1	1	0	1	0	1	4
swan	0	1	1	0	1	1	0	0	1	1	0	0	2	1	0	1	2
termite	0	0	1	0	0	0	0	0	0	1	0	0	6	0	0	0	6
toad	0	0	1	0	0	1	0	1	1	1	0	0	4	0	0	0	5
tortoise	0	0	1	0	0	0	0	0	1	1	0	0	4	1	0	1	3
tuatara	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	0	3
tuna	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	4
vampire	1	0	0	1	1	0	0	1	1	1	0	0	2	1	0	0	1
vole	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	0	1
vulture	0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	1	2
wallaby	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	1	1
wasp	1	0	1	0	1	0	0	0	0	1	1	0	6	0	0	0	6
wolf	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
worm	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7
wren	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2

Bibliography

[Ala98a] Alahakoon L. D. and Halgamuge S. K. Knowledge Discovery with Supervised and Unsupervised Self Evolving Neural Networks. In *Proceedings of the International Conference on Information-Intelligent Systems*, pages 907–910, 1998.

[Ala98b] Alahakoon L.D. and Srinivasan B. Improved Cluster Formation with a Structurally Adapting Neural Network. In *Proceedings of the International Conference on Information Technology*, pages 31–34, 1998.

[Ala98c] Alahakoon L.D., Halgamuge S. K. and Srinivasan B. A Self Growing Cluster Development Approach to Data Mining. In *Proceedings of the IEEE Conference on Systems Man and Cybernatics*, pages 2901–2906, 1998.

[Ala98d] Alahakoon L.D., Halgamuge S. K. and Srinivasan B. A Structure Adapting Feature Map for Optimal Cluster Representation. In *Proceedings of the International Conference on Neural Information Processing*, pages 809–812, 1998.

[Ala98e] Alahakoon L.D., Halgamuge S. K. and Srinivasan B. Unsupervised Self Evolving Neural Networks. In *Proceedings of the Ninth Australian Conference on Neural Networks*, pages 188–193, 1998.

[Ala99a] Alahakoon L.D., Halgamuge S. K. and Srinivasan B. Data Mining with Self Generating Neuro-Fuzzy Classifiers. In *Proceedings of The Eighth IEEE International Conference on Fuzzy Systems*, pages 1096–1101, 1999.

[Ala99b] Alahakoon L.D., Halgamuge S. K. and Srinivasan B. A Self Generating Neurai Architecture for Data Analysis. In *Proceedings of the International Joint Conference on Neural Networks*, 1999.

[Ala00a] Alahakoon L.D. and Halgamuge S. K. Data Mining with Self Evolving Neural Networks. In Hsu C., editor, *Advanced Signal Processing Technology*. World Scientific Publisher, To be published in 2000.

[Ala00b] Alahakoon L.D., Halgamuge S. K. and Srinivasan B. Dynamic Self Organising Maps with Controlled Growth for Knowledge Discovery. *IEEE Transactions on Neural Networks*, To be published in 2000.

[Ala00c] Alahakoon L.D., Halgamuge S. K. and Srinivasan B. Mining a Growing Feature Map by Data Skeleton Modeling. In Last M., editor, *Data Mining and Computational Intelligence*. Physica Verlag, To be published in 2000.

[Ame98] Amerijckx C., Verleysen M., Thissen P. and Legat J. Image Compression by Self-Organized Kohonen Map. *IEEE Transactions on Neural Networks*, 9(3):503–507, 1998.

[Ash43] Ash T. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[Ash94] Ash T. and Cottrell G. W. A Review of Learning Algorithms that Modify Network Topologies. Technical report, University of California, San Diego, 1994.

[Ash95a] Ash T. Dynamic Node Creation in Backpropagation Networks. *Connection Science*, 1:365–375, 1995.

[Ash95b] Ash T. and Cottrell G. Topology-Modifying Neural Network Algorithms. In Arbib M. A., editor, *The Hand Book of Brain Theory and Neural Networks*, pages 990–993. The MIT Press, 1995.

[Avn95] Avner S. Discovery of Comprehensible Symbolic Rules in a Neural Network. In *Proceedings of the International Symposium on Intelligence in Neural and Biological Systems*, pages 64–67, 1995.

[Bar94] Bartlett E. B. A Dynamic Node Architecture Scheme for Layered Neural Networks. *Journal of Artificial Neural Networks*, 1:229–245, 1994.

[Ber97a] Berkan R.C. and Trubatch S.L. *Fuzzy Systems Design Principles - Building Fuzzy IF THEN Rule Bases.* IEEE Press, 1997.

[Ber97b] Berry M. J. A. and Linoff G. *Data Mining Techniques.* Wiley Computer Publishing, 1997.

[Ber97c] Berson A. and Smith S. J. *Data Warehousing, Data Mining and OLAP.* McGraw Hill, 1997.

[Bez92] Bezdek J. C. and Pal S. *Fuzzy Models for Pattern Recognition : Methods that Search for Structures in Data.* IEEE Press, 1992.

[Bez93] Bezdek J. C. A Review of Probabilistic, Fuzzy and Neural Models for Pattern recognition. *Journal of Intelligent Fuzzy Systems*, 1(1):1–25, 1993.

[Big96] Bigus J. P. *Data Mining with Neural Networks.* McGraw Hill, 1996.

[Bim96] Bimbo A. del, Corridoni J. M. and Landi L. 3D Object Classification Using Multi Object Kohonen Networks. *Pattern Recognition*, 29(6):919–935, 1996.

[Bla95] Blackmore J. Visualising High Dimensional Structure with the Incremental Grid Growing Neural Network. Master's thesis, The University of Texas at Austin, 1995.

[Bla96] Blackmore J. and Miikkulainen R. Incremental Grid Growing: Encoding High Dimensional Structure into a Two Dimensional Feature Map. In Simpson P., editor, *IEEE Technology Update.* IEEE Press, 1996.

[Bla98] Blake C., Keogh E. and Merz C. J. University of California, Irvine, Dept. of Information and Computer Sciences. UCI Repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[Bou93] Bouton C. and Pages G. Self-Organisation of the One Dimensional Kohonen Algorithm with Non-Uniformely Distributed Stimuli. *Stochastic Processes and their Applications*, 47:249–274, 1993.

[Bro94] Brown G.D.A., Hulme C., Hyland P.D. and Mitchell I.J. Cell Suicide in the Developing Nervous System: A Functional Neural Network Model. *Cognitive Brain Research*, 2:71–75, 1994.

[Cab98] Cabena P., Hadjinian P., Stadler R., Verhees J. and Zanasi A. *Discovering Data Mining - From Concept to Implementation.* Prentice Hall, 1998.

[Cav94] Cavalli-Sforza L. L., Menozzi P. and Piazza A. *The History and Geography of Human Genes.* Princeton University Press, 1994.

[Cha91] Chan L. W. Analysis of the Internal Representations in Neural Networks for Machine Intelligence. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 578–583, 1991.

[Cha93] Chappell G.J. and Taylor J.G.. The Temporal Kohonen Map. *Neural Networks*, 6:441–445, 1993.

[Che96] Chen M., Han J. and Yu P. S. Data Mining: An Overview from A Database Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[Cot86] Cottrell M. and Fort J. A Stochastic Model of Retinotopy: A Self Organisation Process. *Biological Cybernatics*, 53:405–411, 1986.

[Deb98] Deboeck G. *Visual Explorations in Finance.* Springer Verlag, 1998.

[DeC96] DeClaris N. and Su M. A Neural Network Based Approach to Knowledge Acquisition and Expert Systems. In Simpson P. K., editor, *IEEE Technology Update.* IEEE Press, 1996.

[Dri96] Driankov D., Hellendoorn H. and Reinfrank M. *An Introduction to Fuzzy Control.* Springer, 1996.

[Ede87] Edelman G. M. *Neural Darwinism.* Basic Books, New York, 1987.

[Fay96a] Fayyad U. M., Piatetsky-Shapiro G. and Smyth P. The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM*, 39(11):27–34, 1996.

[Fay96b] Fayyad U.M.,Piatetsky-Shapiro G.,Smyth P. and Uthurusamy R. *Advances in Knowledge Discovery and Data Mining.* AAAI Press, 1996.

[Fay97] Fayyad U. M. The Editorial. *Data Mining and Knowledge Discovery*, 1:5–10, 1997.

[Fis95] Fisher D. Optimisation and Simplification of Hierarchical Clustering. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 118–123, 1995.

[Fla96] Flanagan J. A. Self-Organisation in Kohonen's SOM. *Neural Networks*, 9(7):1185–1197, 1996.

[Fla97] Flanagan J. A. Analysing a Self-Organising Algorithm. *Neural Networks*, 10(5):875–883, 1997.

[Fri91] Fritzke B. Let it Grow - Self Organising Feature Maps with Problem Dependent Cell Structure. In T. Kohonen and K. Makisara and O. Simula and j. Kangas, editor, *Artificial Neural Networks*, pages 403–408. Elsevier Science, 1991.

[Fri92] Fritzke B. Growing Cell Structures - A Self Organising Network in k Dimensions. In Aleksander I. and Taylor J., editor, *Artificial Neural Networks*. Elsevier Science Publishers, 1992.

[Fri93] Fritzke B. Kohonen Feature Maps and Growing Cell Sructures - A Performance Comparision. In Giles C. L., Hanson S. J. and Cowan J. D., editor, *Advances in Neural Information Processing Systems*. Morgan Kaufmann, San Mateo, CA, 1993.

[Fri94] Fritzke B. Growing Cell Structure: A Self Organising Network for Supervised and Un-supervised Learning. *Neural Networks*, 07:1441–1460, 1994.

[Fri95a] Fritzke B. A Growing Neural Gas Network Learns Topologies. In Tesauro G., Touretzky D. S. and Leen T. K., editor, *Advances in Neural Information Processing Systems*. MIT Press, 1995.

[Fri95b] Fritzke B. Growing Grid - A Self Organising Network with Constant Neighbourhood Range and Adaptation Strength. *Neural Processing Letters*, 2(5):9–13, 1995.

[Fri96] Fritzke B. Growing Self Organising Networks - Why ? In *Proceedings of the European Symposium on Artficial Neural Networks*, pages 61–72, 1996.

[Fu 98] Fu L. Rule Learning by Searching on Adapted Nets. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 590–595, 1998.

[Gro69] Grossberg S. On Learning and Energy-entropy Dependance in Recurrent and Non-recurrent Signed Networks. *Journal of Statistical Physics*, 1:319–350, 1969.

[Gur97] Gurney K. *An Introduction to Neural Networks*. London UCL Press, 1997.

[Hal95] Halgamuge S. K. and Glesner M. Fuzzy Neural Networks: Between Functional Equivalence and Applicability. *International Journal of Neural Systems*, 06(2):185–196, 1995.

[Hal97] Halgamuge S. K. Self Evolving Neural Networks for Rule Based Data Processing. *IEEE Transactions on Signal Processing*, 44(11):185–196, 1997.

[Han90] Hanson S.J. Advances in Neural Information Processing Networks. In Touretzky D., editor, *Meiosis Networks*, pages 533–541. Morgan kaufman CA, 1990.

[Han95] Hanson S.J. Backpropagation: some Comments and Variations. In Chauvin Y. and Rumelhart D.E., editor, *Backpropagation: Theory, Architectures and Applications*, pages 237–271. Erlbaum : Hillsdale NJ, 1995.

[Has95] Hassoun M. *Fundamentals of Artificial Neural Networks.* Massachusetts Institute of Technology, 1995.

[Hay94] Haykin S. *Neural Networks : A Aomprehensive Foundation.* Prentice Hall, 1994.

[Heb49] Hebb D. *The Organisation of Behaviour.* Wiley, 1949.

[Hol94] Holsheimer M and Siebes A. P. J. M. Data Mining: the Search for Knowledge in Databaes. Technical Report CS-R9406, CWI, 1994.

[Hor99] Horton H. L. *Mathematics at work.* New York, Industrial Press, 1999.

[Inm97] Inmon W. H. *Data Warehouse Performance.* John Wiley, 1997.

[Inm98] Inmon W. H. *Managing the Data Warehouse.* Wiley, 1998.

[Joc90] Jockush S. *A Neural Network Which Adapts it's Structure to a Given Set of Patterns*, pages 169–172. Elsvier Science, 1990.

[Jou95] Joutsiniemi S., Haski S. and Larsen A. Self-Organising Map in Recognition of Topographic Patterns of EEG Spectra. *IEEE Transactions on Biomedical Engineering*, 42(11):1062–1068, 1995.

[Kan91] Kangas J. Time-Dependant Self Organising Maps for Speech Recognition. In Kohonen T., Makisara K., Simula O. and Kangas J., editor, *Artificial Neural Networks*, pages 1591–1594. Elsevier Science, 1991.

[Kar96] Kartalopolos S.V. *Understanding Neural Networks and Fuzzy Logic.* IEEE Press, 1996.

[Kec95] Kechris A. S. *Classical Descriptive Set Theory.* Springer Verlag, 1995.

[Kei96] Keim D. A. and Kriegel H. Visualisation techniques for Mining Large Databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):923–938, 1996.

[Ken98] Kennedy R. L., Lee Y., Roy B. V., Reed C. D., and Lippmann R. P. *Solving Data Mining Problems through Pattern Recognition.* Prentice Hall, 1998.

[Kno96] Knorr E. M. and Ng R. T. Finding Aggregate Proximities and Commonalities in Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):884–897, 1996.

[Koh81] Kohonen T. Automatic Formation of Topological Maps of Patterns in a Self Organising System. In *Proceedings of the 2nd Scandinanvian Conference on Image Analysis*, pages 214–220, 1981.

[Koh82a] Kohonen T. Analysis of Simple Self-Organising Process. *Biological Cybernatics*, 44:135–140, 1982.

[Koh82b] Kohonen T. Self-Organised Formation of topologically Correct Feature Maps. *Biological Cybernatics*, 43:59–69, 1982.

[Koh89] Kohonen T. *Self-Organization and Associative Memory.* Springer Verlag, 1989.

[Koh90] Kohonen T. The Self Organising Map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[Koh91] Kohonen T. Self Organising Maps: Optimisation Approaches. In Kohonen T., Makisara K., Simula O. and Kangas J., editor, *Artificial Neural Networks*, pages 981–990. Elsevier Science, 1991.

[Koh95] Kohonen T. *Self-Organizing Maps.* Springer Verlag, 1995.

[Koh96a] Kohonen T. Things You Haven't Heard About the Self Organising Maps. In Simpson P., editor, *IEEE Technology Update*, pages 128–137. IEEE, 1996.

[koh96b] kohonen T., Oja E., Simula O., Visa A. and Kangas J. Engineering Applications of the Self-Organising Map. *Proceedings of the IEEE*, 84(10):1358–1384, 1996.

[Law99] Lawrence R. D., Almasi G. S. and Rushmeier H. E. A Scalable Parallel Algorithm for Self Organising Maps with Applications to Sparse Data Mining. *Data Mining and Knowledge Discovery*, 3(2):171–195, 1999.

[Lee91] Lee T. *Structure Level Adaptation for Artificial Neural Networks.* Kluwer Academic Publishers, 1991.

[Loo97] Looney, C. G. *Pattern Recognition Using Neural Networks : Theory and Algorithms for Engineers and Scientists.* New York : Oxford University Press, 1997.

[Mal73] Malsberg von D. Self-Organisation of Orientation Sensitive Cells in the Striate Cortex. *Kybernatics*, 14:85–100, 1973.

[Mar90] Marshall A. M. Self Organising Neural Networks for Perception of Visual Motion. *Neural Networks*, 3:45–74, 1990.

[Mar91] Martinetz T. M. and Schultan K. J. *A Neural Gas Network Learns Topologies*, pages 397–402. Elsvier Science, 1991.

[Mar94] Martinetz T. M. and Schultan K. J. Topology Representing Networks. *Neural Networks*, 7(3):507–522, 1994.

[Mar95] Marshall A. M.. Motion Perception: Self Organisation. In Arbib M. A., editor, *The Hand Book of Brain Theory and Neural Networks*, pages 589–591. The MIT Press, 1995.

[Mat93] Matheus J. M., Chan P. K. and Piatetsky-Shapiro G. Systems for Knowledge Discovery in Databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):903–913, 1993.

[McN94] McNeill F. M. *Fuzzy logic : A Practical Approach.* Boston : AP Professional, 1994.

[Meh97] Mehnert A. and Jackway p. An improved Seeded Region Growing Algorithm. *Pattern Recognition Letters*, 18:1065–1071, 1997.

[Min69] Minsky M. and Papert S. *Perceptrons.* MIT Press, 1969.

[Mir96] Mirkin B. *Mathematical Classification and Clustering.* Kluwer Academic Publishers, 1996.

[Moz89] Mozer M.C. and Smolensky P. Using Relevance to Reduce Network Size Automatically. *Connection Science*, 1:3–16, 1989.

[Ngu97] Nguyen H.T. and Walker E.A. *Fuzzy Logic.* CRC Press, Florida, 1997.

[Oka92] Okabe A., Boots B. and Sugihara K. *Spatial Tesselations, Concepts and Applications of Voronoi Diagrams.* John Wiley and Sons, 1992.

[Pur94] Purves D. *Neural Activity and the Growth of the Brain.* Cambridge University Press, 1994.

[Qui98] Quinlan P. T. Structural Change and Development in Real and Artificial Neural Networks. *Neural Networks*, 11:577–599, 1998.

[Rit88] Ritter H. and Schulten K. Kohonen's Self Organising Maps : Exploring their Computational Capabilities. In *Proceedings of the IEEE Internaiional Conference on Neural Networks*, pages 109–116, 1988.

[Rit91] Ritter H. Learning with the Self Organising Maps. In Kohonen T., Makisara K., Simula O. and Kangas J., editor, *Artificial Neural Networks*, pages 379–384. Elsevier Science, 1991.

[Rit92] Ritter H., Martinetz T. M. and Schulten K. *Neural Computation and Self-Organizing Maps.* Addison Wesley, 1992.

[Rit93] Ritter H. Parameterised Self Organising Maps. In Gielen S. and Kappen B., editor, *Artificial Neural Networks 3.* Elsevier Science, 1993.

[Roj96] Rojas R. *Neural Networks.* Springer, 1996.

[Sch93] Schomaker L. Using Stroke or Character BAsed Self Organising Maps in the Recognition of On-line, Connected Cursive Scripts. *Pattern Recognition*, 26(3):443–450, 1993.

[Sch96] Schurmann J. *Pattern Classification - A Unified View of Statistical and Neural Approaches.* John Wiley and Sons, 1996.

[Ses94] Sestito S. and Dillon T. S. *Automated Knowledge Acquisition.* Prentice Hall, 1994.

[Sie91] Sietsma J. and Dow R.J.F. Creating Artificial Neural Network that Generalise. *Neural Networks*, 4:67–79, 1991.

[Sim96] Simpson P. K. Foundatins of Neural Networks. In Simpson P. K., editor, *Neural Networks Applications*, pages 1–22. IEEE Press, 1996.

[Sri97] Srinivasa N. and Sharma R. SOIM: A Self-Organising Invertible Map with Applications in Active vision. *IEEE Transactions on Neural Networks*, 8(3):758–773, 1997.

[Suk95] Suk M., Koh J. and Bhandarkar S. M. A Multilayer Self-organising Feature Map for Range Image Segmentation. *Neural Networks*, 30(2):1215–1226, 1995.

[Tay95] Taylor G. J. Self-Organsation in the Time Domain. In Arbib M. A., editor, *The Hand Book of Brain Theory and Neural Networks*, pages 843–846. The MIT Press, 1995.

[Thi93] Thiran P. and Hasler M. Self Organisation of a One Dimensional Kohonen Network with Quantised Weights and Inputs. *Neural Networks*, 7(9):1427–1439, 1993.

[Ult92] Ultsch A. Knowledge Acquisition with Self-Organising Neural Networks. In Aleksander I. and Taylor J., editor, *Artificial Neural Networks 2*, pages 735–739. Elsevier Science, 1992.

[Vil97] Villmann T., Der R., Hermann M. and Martinetz M. Topology Preservation in Self-Organising Feature Maps: Exact Definition and Measurement. *IEEE Transactions on Neural Networks*, 08:256–266, 1997.

[Vuo94] Vuorimaa P. Fuzzy Self Organising Map. *Fuzzy Sets and Systems*, 66:223–231, 1994.

[Wan95] Wang L. and Alkon D. L. An Artificial Neural Network System for Temporal-Spatial Sequence Processing. *Pattern Recognition*, 8:1267–1276, 1995.

[Wan97] Wann C. and Thomopoulos S. C. A. A Comparative Study of Self-organising Clustering Algorithms Dignet and ART2. *Neural Networks*, 10(4):737–753, 1997.

[Wes98] Westphal C. *Data Mining Solutions*. Wiley Computer Publishing, 1998.

[Wil76] Willshaw D. J. and Malsberg von D. How patterned neural connections can be set up by self-organisation. *Proceedings of the Royal Society of London*, B194:431–445, 1976.

[Yag94] Yager R. R. *Essentials of fuzzy modeling and control*. John Wiley, 1994.

[Yan92] Yang H. and Dillon T. S. Convergence of Self Organising Neural Algorithms. *Neural Networks*, 5:485–493, 1992.

[Zad65] Zadeh L.A. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[Zha97] Zhang T., Ramakrishnan R. and Livny M. BIRCH: A New Data Clustering Algorithm and its Applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.