School of Computing

Learning and Development in Kohonen-style Self-Organising Maps

Russell Keith-Magee

This thesis is presented as part of the requirements for
the award of the Degree of Doctor of Philosophy
of Curtin University of Technology

July 2001

Contents

List of Figures

List of Tables

Abstract

This thesis presents a biologically inspired model of learning and development. This model decomposes the lifetime of a single learning system into a number of stages, analogous to the infant, juvenile, adolescent and adult stages of development in a biological system. This model is then applied to Kohonen's SOM algorithm.

In order to better understand the operation of Kohonen's SOM algorithm, a theoretical analysis of self-organisation is performed. This analysis establishes the role played by lateral connections in organisation, and the significance of the Laplacian lateral connections common to many SOM architectures.

This analysis of neighbourhood interactions is then used to develop three key variations on Kohonen's SOM algorithm. Firstly, a new scheme for parameter decay, known as *Butterworth Step Decay*, is presented. This decay scheme provides training times comparable to the best training times possible using traditional linear decay, but precludes the need for *a priori* knowledge of likely training times. In addition, this decay scheme allows Kohonen's SOM to learn in a continuous manner.

Secondly, a method is presented for establishing core knowledge in the fundamental representation of a SOM. This technique is known as *Syllabus Presentation.* This technique involves using a selected training syllabus to reinforce knowledge known to be significant. A method for developing a training syllabus, known as *Percept Masking*, is also presented.

Thirdly, a method is presented for preventing the loss of trained representations in a continuously learning SOM. This technique, known as *Arbor Pruning*, involves restricting the weight update process to prevent the loss of significant representations. This technique can be used if the data domain varies within a known set of dimensions. However, it cannot be used to control forgetfulness if dimensions are added to or removed from the data domain.

Preface

Elements of this thesis have been published in peer-reviewed literature.

Elements of Chapter 4 appear in:

Keith-Magee, R., Venkatesh, S., and Takatsuka, M. (1999) "Beyond the topological map: developing alternate mappings in self-organisation", Proceedings of the 5th International/National Biennial Conference on Digital Image Computing: Techniques and Applications (DICTA'99), pages 88–93, Australian Pattern Recognition Society.

Elements of Chapter 5 appear in:

Keith-Magee, R., Venkatesh, S., and Takatsuka, M. (1999) "An empirical study of neighbourhood decay in Kohonen's Self-Organising Map", Proceedings of the 1999 International Joint Conference on Neural Networks (IJCNN'99). Electronic Proceedings ID 2150.

Elements of Chapter 5 appear in:

Keith-Magee, R., Venkatesh, S., and Takatsuka, M. (1999) "The importance of neighbourhood size in self-organising systems", Proceedings of the 6th International Conference on Neural Information Processing (ICONIP'99), pages 267–272. Edited by T. Gedeon and P. Wong and S. Halgamuge and N. Kasabov and D. Nauck and K. Fukushima, Asia Pacific Neural Network Assembly.

Acknowledgements

> I love deadlines. I love the whooshing sound they make as they fly by.
> — *Douglas Adams*

> I may not have gone where I intended to go, but I think I have ended up where I intended to be.
> — *The Long Dark Tea-Time of the Soul* (Adams, 1988)

Although the writing of a thesis ultimately falls to the hands of a single student, it is by no means a solitary act. During the writing of this thesis, I have been fortunate to have the counsel and friendship of a great many people. This thesis would not have come to fruition were it not for these individuals.

Firstly, I would like to thank my supervisor and friend, Professor Svetha Venkatesh, for her equal parts wisdom, patience, and firm application of the metaphorical boot to my derrière. Over all others, Svetha is the person responsible for making sure this tome actually materialised. For this, I thank her.

I have an unfortunate habit of acquiring co-supervisors, and thereby encouraging them to leave the country. I would like to thank Garry Briscoe and Masahiro Takatsuka for their time and assistance while they were at Curtin University. I would also like to thank Mike Robey for being my co-supervisor and friend. I would especially like to thank him for *not* leaving the country, despite his association with my thesis.

I would like to thank my fellow travellers on the good ship PhuD: Leon Blackwell, Ezra Tassone and Brett Adams. Your ears, minds and desks are now safe from my perverted influence. Thank you for not pressing charges. May you one day find yourself at the end of your respective doctoral journeys.

I would also like to thank "The Nerds": Stuart Campbell, Tom Drummond and Andy McCabe, and their respective better halves: Linda, Jenny and Deb. Although

The Nerds have spread to the four corners of the earth over the last three years, this group of individuals is responsible for cementing my decision to start the doctoral journey, and for providing invaluable advice along the way. Despite their physical displacement, they remain close in my heart and mind.

Although I find myself to be angelic in demeanour, I have been reliably informed that I can become somewhat sub-human when stressed. I have also been reliably informed that this has been a good description of my demeanour for the last three years. I would therefore like to thank my girlfriend, April Millar for her love and support over this period, and for standing by me despite all attempts on my part to drive her away.

My family and friends are also deserving of many thanks. My parents Kay and Pat, and sister Karen, were instrumental in establishing the love of learning which has culminated in this thesis; for this, plus their emotional and financial support over the years, I shall be eternally grateful. The Rossmoyne gang have also played a large part in my life over the years, providing invaluable friendship and stress relief; for this, I thank them all. Of this group, I would especially like to thank Narrelle Morris, resident Arts Student™, for proof reading this thesis and thereby demonstrating exactly how much of the English language I don't know, and how often i forgte to finish my

While working on this thesis, I was the proud recipient of the Curtin University Silver Jubilee Scholarship. I would like to thank Curtin University for providing this scholarship.

Lastly, I would like to dedicate this thesis to the memory of Douglas Noel Adams. DNA passed away on May 11, 2001, as I was nearing completion of this thesis. Although I never met him, spoke with him, or directly sought his advice, he is singularly responsible for my love of reading, my obsession with the absurd, and a large proportion of my scripting repertoire. So long, Douglas, and thanks for all the fish.

Chapter 1

Introduction

> The Story so far:
> In the beginning, the Universe was created.
> This has made a lot of people very angry, and has been widely regarded as a bad move.
> — *The Restaurant at the End of the Universe* (Adams, 1980)

1.1 Machine Learning and Development

The last 50 years has seen an explosion in the computational power available to those attempting to solve complex problems. However, the sheer complexity of the real world defies this power. Despite the computational consequences of Moore's Law, the complexity associated with solving real-world problems necessitates that an intelligent, rather than brute-force approach be taken to solving these problems.

A large number of architectures have been proposed for the purpose of machine learning. The majority of these algorithms utilise some form of analysis of clustering in the data set, or of entropy minimisation on a predetermined heuristic. These techniques are then used to generate optimal (or near optimal) representations of a given static knowledge base. When faced with a single, static training set, it is possible for a batch learning scheme to scan the domain and range of training examples and search for an optimal representation of this data space.

However, in real-world learning problems, learning is rarely a batch process. Real-world learning systems do not usually have access to a complete and canonical set of training examples prior to the commencement of training. Rather, learning is a continuous process with input stimuli that are highly non-stationary (i.e., the probability distribution of training cases varies over time). When the training set is constantly changing, *a priori* scanning of a data set is not a viable proposition.

The use of a real-world data set changes the learning problem into an attempt to optimize a representation over time with respect to a *source* of training exemplars, rather than an attempt to learn a globally optimised representation of a single data set. The domain of the source cannot be known canonically at the start of training; it can only be derived from long term experience. In addition, an optimal representation for any given time period may not be optimal over all time. Losses in optimality must be accepted in the short term in order to achieve long term success.

This problem is exacerbated by changes in the source during the lifetime of the learning mechanism. The probability density function defining the source is not guaranteed to be constant over time — in fact, it is almost guaranteed to vary in most real-world learning problems. It would therefore be desirable for the representation of the training source to change over time to reflect these changes, adding new detail as it becomes available, and remembering old detail even if it ceases to be represented by the training source.

The static nature of conventional batch training mechanisms make them unsuitable for applications where data sets which exhibit either or both of these characteristics. An alternative approach is required; one which is able to adapt to the dynamic nature of a real-world training source.

One alternative source for solutions to these problems is the behaviour observed in biological systems. Most artificial learning systems have an two stage approach to learning (an active learning period, followed by an 'off-line' period of use). By comparison, biological systems do not cease learning after an initial, clearly defined training phase. The neural connections of an intelligent biological system are never completely frozen; they adapt, even if only slightly, to all inputs. Consequently, biological systems are able to learn by experience throughout their lifetime[1]. Biological systems must learn in a continuous fashion, as the realm of experience changes from day to day. When the domain of input stimuli changes, biological systems are able to adapt, using past experience as a guide, whereas artificial systems require retraining.

While continuously learning biological systems do not have a clearly defined period of 'training', they do experience an extended period of development. This developmental phase is the period of time required for the continuous learning process to establish a useful knowledge representation. During this phase, the classifications/predictions

[1]Although the scope for large scale change decreases with age; as the aphorism goes, 'you can't teach an old dog new tricks'

made by the system are frequently in error. In biological systems, the developmental process is frequently supervised (through parental interaction) to encourage the development of strong survival characteristics.

The process of development is extremely significant to the individual. The learning that takes place during these formative stages has an effect over the lifetime of the system. For example, kittens raised in an environment which contains only vertically oriented visual stimuli are unable in adulthood to see horizontally oriented features (Blakemore and Cooper, 1970). Care must therefore be taken to ensure that the developmental process is carefully guided.

It is interesting to note that in biological systems, the intelligence of an organism is directly related to the length of its development. Insects have a no significant development phase. As a result, their intelligence is generally limited to simple stimulus-response reactions. Cats and dogs have developmental phase which lasts 1-2 years (as kittens and puppies respectively), and exhibit a simple intelligence. Humans experience a prolonged period of development, spanning some 10-20 years, and exhibit a sophisticated ability to learn, reason, and adapt.

The use of biological metaphors is not uncommon in machine learning research. Given that the only existing complete model of intelligent behaviour is biological, it comes as no surprise that one of the most common approaches to developing intelligent behaviour is to draw inspiration from a biological source. The theory underlying neural network research is perhaps the most prominent example of such an approach. Starting with a mathematical model of a single neuron, neural network theory attempts to develop algorithms which will allow a machine to learn in the same manner as biological systems.

The process of learning and development would appear to be an essential component of the emergence of biological (i.e., animal) intelligent behaviours. By endowing a machine learning algorithm with the same characteristics, it may be possible to improve the capabilities of a learning algorithm in continuous problem domains.

1.2 Aims

This thesis presents a biologically inspired model of the learning and development process, and applies this model to Kohonen's Self-Organising Map (SOM) algorithm.

In particular, this thesis aims to answer the following questions:

- What is the role played by neighbourhood relationships in a self-organising process? Can these relationships be exploited to control the form of a self-organised data space representation, and the rate of convergence towards this representation?

- Can a Kohonen-style SOM algorithm be altered so as to provide a facility for the continuous learning of a data space? Can a SOM be constructed which will allow the acquisition of new training knowledge over time?

- Is there a mechanism for controlling the stability-plasticity dilemma within a continuously learning SOM? Can a SOM be constructed which will resist the loss of core training knowledge? Can a SOM be constructed which will prevent the loss of early training without preventing the acquisition of new knowledge?

In addition, the solutions found to these questions must be mutually compatible. Each solution must be able to be utilised in isolation, or in concert with other proposed solutions.

1.3 Approaches

The biologically inspired model of learning and development presented in this thesis decomposes the lifetime of a single learning system into a number of stages. These stages are analogous to the infant, juvenile, adolescent and adult stages of development in a biological system. This model of learning and development is applied to Kohonen's SOM algorithm.

Kohonen's SOM algorithm has been chosen as a research platform because of the biological origins of the algorithm. The application of biological metaphors to a computational algorithm provided the original inspiration for both neural networks and Kohonen SOMs. It is hoped that similar benefits may be drawn in the construction of a model of learning and development through the application of biological metaphors to Kohonen's SOM algorithm.

In order to better understand the operation of Kohonen's SOM algorithm, a theoretical analysis of self-organisation is performed. This analysis establishes the role played by lateral connections in organisation, and the significance of the Laplacian lateral connections common to many SOM architectures.

This analysis of neighbourhood interactions is then used to develop three key variations on Kohonen's SOM algorithm. Firstly, a new scheme for parameter decay, known

as *Butterworth Step Decay*, is presented. This decay scheme provides training times comparable to the best training times possible using traditional linear decay, but precludes the need for *a priori* knowledge of likely training times. In addition, this decay scheme allows Kohonen's SOM to learn in a continuous manner.

Secondly, a method is presented for establishing core knowledge in the fundamental representation of a SOM. This technique is known as *Syllabus Presentation.* This technique involves using a selected training syllabus to reinforce knowledge known to be significant. A method for developing a training syllabus, known as *Percept Masking*, is also presented.

Thirdly, a method is presented for preventing the loss of trained representations in a continuously learning SOM. This technique, known as *Arbor Pruning*, involves restricting the weight update process to prevent the loss of significant representations. This technique can be used if the data domain varies within a known set of dimensions. However, it cannot be used to control forgetfulness if dimensions are added to or removed from the data domain.

The approach taken in this thesis is largely empirical in nature, due to the impractical nature of a theoretical approach. There are many examples of theoretical proofs on self-organising processes (Amari, 1980) (Tanaka, 1990) (Amari, 1990) (Kaski and Kohonen, 1994), including proofs on Kohonen's SOM algorithm (Cottrell *et al.*, 1998) (Flanagan, 1998) (Luttrell, 1994). These proofs cover many aspects of self-organisation, including conditions for parameter selection and decay. However, these proofs either rely upon extensively simplified (and therefore limited) versions of SOM algorithms, or draw only vague conclusions on the operation of complete SOM algorithms. The complexity of interactions in a self-organising process, coupled with the iterative nature of the process, inhibits the development of strong theoretical models of self-organisation. As a result of these intrinsic difficulties, an empirical approach is adopted.

To ensure the validity of the algorithmic extensions presented in this thesis, each extension is tested in isolation and in concert, using both synthetic and real-world data sets. Each experiment is performed repeatedly in order to ensure experimental consistency. A wide range of experiments are performed to ensure that the results obtained are not aberrations caused by specific experimental conditions.

1.4 Significance

There are many examples of the practical application of Kohonen-style SOMs to static problems (Kangas and Kaski, 1998). However, the literature provides no examples of the application of Kohonen-style SOMs to continuous problems. Kohonen (2001) provides a recent canonical survey of variants of the basic SOM algorithm; this survey does not discuss any variants capable of continuous learning.

This thesis adds continuous learning problems to the range of problems to which a Kohonen-style SOM architecture can be applied. The novelty of this thesis lies in the algorithms developed to achieve this end.

Specifically, this thesis makes four key contributions:

- a metric for evaluating the quality of a SOM data representation which is independent of the weight space of the SOM;
- a parameter decay scheme for Kohonen's SOM, called *Butterworth Step Decay*, which precludes the need for *a priori* knowledge of training time, and allows a SOM to learn continuously;
- a method, called *Syllabus Presentation*, which can establish core training knowledge in the fundamental representation of a SOM, and an associated method for developing a training syllabus called *Percept Masking*; and
- a method, called *Arbor Pruning*, which can prevent the loss of trained representations in a continuously learning SOM.

1.5 Structure of Thesis

This thesis is organised as follows.

In Chapter 2, a survey of the *status quo* is presented. This survey includes the history and significance of neural and self-organising architectures, a review of Kohonen's Self-Organising Map (SOM) algorithm and its most significant variants, a review of SOM quality measures, an introduction to the continuous learning problem, and a review of machine learning algorithms which have been used to learn continuously (including Kohonen's SOM algorithm).

In Chapter 3, the theoretical and empirical framework for this thesis is established. This includes a biologically inspired model of learning and development, and a metric for evaluating the quality of SOM data representations.

In Chapter 4, a theoretical analysis of neighbourhood relationships in self-organising processes is introduced. This analysis investigates the role played by lateral connections in organisation, and the significance of the Laplacian lateral connections common to many SOM architectures.

In Chapter 5, the lesson learned in Chapter 4 are applied to Kohonen's SOM algorithm, resulting in the development of a new scheme for parameter decay. This scheme is known as *Butterworth Step Decay.*

In Chapter 6, a continuous learning extension to Kohonen's SOM algorithm is presented. This scheme is developed by exploiting a key characteristic of the Butterworth Step Decay scheme.

In Chapter 7, a method for developing a strong fundamental SOM topology is presented. This technique is known as *Syllabus Presentation.* This technique involves using a selected training syllabus to reinforce knowledge known to be significant. A method for developing a training syllabus, known as *Percept Masking*, is also presented.

In Chapter 8, a method is presented which is capable of realising a reduction in forgetfulness in a continuously learning SOM. This technique is known as *Arbor Pruning.*

In Chapter 9, the lessons learned in Chapters 5–8 are tied together in a single framework.

Finally, in Chapter 10, we conclude, and provide some directions for future work.

The data sets used throughout this thesis for training and testing are described in Appendix A.

Chapter 2

Status Quo

> "What would you say if I told you that I wasn't from Guilford after all, but in fact from a small planet in the vicinity of Betelgeuse?"
> "Why? Is that the sort of thing that you're likely to tell me?"
> — *The Hitch-Hikers Guide to the Galaxy* (Adams, 1979)

2.1 Introduction

No thesis exists in a vacuum. The theories and models presented in this thesis represent the extension of models which have existed in general terms for almost fifty years, and have been used in a computational form for almost twenty years. In this chapter, the theories and models which form the foundation of this thesis will be presented. Firstly, the history and significance of neural and self-organising architectures will be discussed. The most significant implementation of these self-organising principles is Kohonen's Self-Organising Map (SOM) algorithm; this algorithm and its most significant variants will be presented. The continuous learning problem is introduced, and Kohonen's algorithm will be critiqued with respect to its usefulness as a continuous learning mechanism. Mechanisms for evaluating the quality of a topographic mapping produced by a SOM will also be discussed. Finally, other machine learning algorithms which have been demonstrated to have an ability to learn continuously will be presented and discussed.

2.2 Biological Origins of Learning

Given that the only existing complete model of intelligent behaviour is biological, it comes as no surprise that one of the most common approaches to developing intelligent behaviour is to draw inspiration from a biological source. Neural architectures were

amongst the first general problem solving architectures to be suggested by the computer science community. Drawing from a theoretical basis drawn from biological observation, neural networks are an attempt to use a model of a biological neuron to build complex computational classification systems. From the original work of McCulloch and Pitts (1943), Hebb (1949), Rosenblatt (1958) and Widrow and Hoff (1960), through to the more recent, specialised work (such as that of Fahlman (1991)), neural networks have become a commonly used methods for generalised problem solving.

The success of neural network systems can be attributed to the many useful characteristics of these systems. Neural architectures are a computationally Turing equivalent; in some cases they have been shown to exhibit super-Turing characteristics (Siegelmann *et al.*, 1997). They have been applied to a wide range of tasks, including pattern recognition, numerical computation, prediction, and pattern classification. In addition, neural architectures are easily implemented on parallel architectures, allowing massive performance improvements when suitable hardware is available.

The original philosophy underlying neural networks — that of having a large number of densely connected simple processing elements — was originally biologically inspired. However, the majority of neural algorithms which have subsequently been proposed have moved away from the original biological model. Recent neural network models have become increasingly engineered towards specific applications rather than remaining consistent with biological observation. Consequently, most of these algorithms have many parameters which must be fine-tuned. In addition, these systems work best with a minimal number of inputs; with a larger number of inputs, it becomes harder to segment the significant features from inputs, as local gradient minima are more likely to be mistaken for global minima.

Existing feedforward neural network algorithms operate almost universally as function approximators and non-linear classifiers. This is an appropriate and useful behaviour for a wide range of engineering tasks. However, not all problems can be posed in terms of a learned numerical output resulting from a given numerical input. Problems which cannot be posed in terms of a numerical relationship (for example, 1-to-many functions) or simple class membership (for example, is this person tall?) are difficult to represent using a classical neural network.

2.3 Biological Self-Organisation

The study of self-organisation shares a common heritage with neural network research. However, there is a significant difference in the degree to which these computational models emulate biology. A traditional neural network is a computational model based on the dynamics of an individual biological neuron. Large numbers of these computational units are then organised to create an entire learning system. However, there is no evidence to suggest that these organisational structures (such as feed-forward layers) or the learning algorithms supporting these structures (such as the many variants of the back-propagation algorithm) resemble the biological structures which originally inspired them.

However, the theory underpinning self-organisation is based upon the structural and dynamic model of neuron behaviour which has observed in biological systems. Self-organising behaviour can be found in a number of cortical groups, in a wide range of species. In higher vertebrates, two dimensional self-organising maps have been observed in the retinal mapping of the striate cortex (Talbot and Marshall, 1941), and on the surface of the superior colliculus (Apter, 1945)(Cooper *et al.*, 1953); in lower vertebrates, the optic tectum has been observed to be retinotopically organised (Gaze, 1958). Self-organisation has also been found in less primitive cognitive functions such as the somatosensory cortex (Rose and Mountcastle, 1959) and the motor cortex (Woolsey, 1952), and between central structures, as has been observed in the ordered intertectal projection in amphibia (Gaze *et al.*, 1970). In addition, one dimensional tonotopic maps have been observed in the cochlea (Rose *et al.*, 1959).

These biological observations are interesting in themselves, but it is the cause of this phenomenon that is of more interest to those wishing to emulate biological intelligence in a computational framework. Why is the cortex organised in this way, and through what mechanism do neurons fall into this manner of organisation?

2.3.1 Von der Malsburg

At the time at which Hubel and Wiesel performed their original work on the physiological and cognitive structure of the brain, genetic predetermination was the best explanation that could be offered for cortical topological organisation. It was proposed by Hubel and Wiesel (1963) that the entire cortical structure was drawn from a genetic blueprint. Von der Malsburg (1973) suggested that the complete genetic predetermination of connections was implausible, for three reasons:

- it would require an immense volume of genetic information to encode every neural connection in the cortex;
- genetically predetermined connections limit the plasticity of a topological mapping during formation; and
- plasticity must be possible, long after genetic expression has been performed.

The first of these criticisms could be challenged by considering the genetic code as a fractal or generative expression method. However, the remaining two criticisms remain valid: the process of map formation is strongly influenced by the domain of stimuli presented during formation. Sensitivity to the domain of training stimuli would not be permitted by a system predetermined by a genetic code established at conception.

Von der Malsburg (1973) proposed that rather than being predetermined, the topological organisation of connection weights is an algorithmic method which is fully functionally dependent on the provided stimuli. In this model, the self-organising plane consists of excitatory E-cells, and inhibitory I-cells, in equal numbers. The connection between cells is modeled by a function $f(x)$, where x is the distance between cells. This function decreases monotonically with increasing distance — for example, a Gaussian function centered at $x = 0$. This function can be characterised by its maximum amplitude A, and the range R over which the connections are significant. The connections between $E \rightarrow E$, $E \rightarrow I$ $I \rightarrow E$, and $I \rightarrow I$ cells vary according to the following relationship:

- $f_{EE}(x)$ has $A_{EE} > 0$, range R_{EE},
- $f_{EI}(x)$ has $A_{EI} > 0$, range $R_{EI} = R_{EE}$,
- $f_{IE}(x)$ has $A_{IE} < 0$, range $R_{IE} > R_{EE}$,
- $f_{II}(x)$ has $A_{II} = 0$.

The summation of these weights over all cells yields a net Laplacian interconnection between cells. Using these lateral connections and randomly instantiated synaptic weights, each cell is allowed to relax into a stable state. During relaxation, the excitation level of the cells oscillates; however, these oscillations dampen, approaching a stable value.

Learning was then performed using a Hebbian update rule, in which synapses are increased in strength proportional to the level of excitation. The gain used in this

learning remained constant throughout the experiment. Normalisation across cells was also performed to prevent saturation.

Von der Malsburg reached four key conclusions regarding self-organising processes (Willshaw and Von der Malsburg, 1976):

1. maps develop in a step-by-step, orderly fashion;
2. lateral connections within the map are initially widespread, but during the learning process they decay in extent;
3. the orientation of the map is established early in the self-organising process, but the final pattern of connections takes longer to develop;
4. appropriate starting conditions are essential to the formation of a good map.

The conclusions reached by Von der Malsburg are theoretical generalisations, rather than experimental results (Willshaw and Von der Malsburg, 1976, Sec. 6). In particular, the second conclusion — that decaying neighbourhood size is a requirement for topological learning — was untested.

However, two important qualities are demonstrated by Von der Malsburg's experiments. Firstly, learning was achieved without the need for a decaying learning gain parameter. In most theoretical and experimental studies which have been performed subsequently, variation in gain has been considered the most significant factor controlling learning. Secondly, global topographical organisation was only observed on small maps. In large map simulations, local topographical organisation occurred but global topology was erratic.

The cause of the second result was theoretically demonstrated by Amari (1980). Amari demonstrated that a map will develop single peak of excitation provided that the domain of the lateral connection function spans the entire map. Lateral connection functions with a smaller range resulted in multiple peaks developing on a map. During Von der Malsburg's experiments, the range of the lateral connection function was small, and was not decayed. Consequently, on small maps, the connection function spanned the entire map, and global organisation was observed. On larger maps, Von der Malsburg's small connection function did not span the map, and many areas of excitation occurred, resulting in erratic global topology.

While the theoretical models of Von der Malsburg and Amari are useful in exploring the biological implications of self-organisation, they are not especially appropriate for

use as a generalised learning tool. Since the motivation for these models is one of modelling biological activity, they sacrifice computational efficiency in order to maintain consistency with observed biological structure. In engineering applications, biological consistency is of secondary interest to the efficient evaluation of a solution to a specific problem.

2.4 Kohonen's Self-Organising Feature Map

The biological process of self-organisation has been utilised by a range of computational algorithms (Bishop *et al.*, 1997) (Bishop *et al.*, 1998) (Heskes and B., 1993) (Luttrell, 1991) (Luttrell, 1992). However, it has been Kohonen's discrete approximation of this biological process (Kohonen, 1982) (Ritter and Kohonen, 1989), and variants thereof (for a summary see Kohonen, 2001, Ch 5.), which have received the most attention in the computing literature over the past twenty years. As a computational technique, Kohonen's SOM algorithm has been found to be an extremely useful method of arriving at a topological organisation for a wide range of applications (see Kohonen, 2001, Ch. 7 for an extensive list). Kohonen's algorithm is also computationally efficient, as it removes the need for iteration to establish excitation levels.

Kohonen's SOM model is structurally similar to that of Von der Malsburg: a set of neurons, each of which is connected to every other neuron through a connection of predetermined strength, and to a set of inputs via a set of initially random weights. The novel aspect of Kohonen's algorithm is the key change to the lateral interaction functions. This change significantly improves the computational efficiency of the self-organisation process.

Kohonen was able to demonstrate that the purpose of the lateral interaction kernel in neural sheet models such as Von der Malsburg's was to establish Winner-Take-All behaviour (Kohonen, 2001). In biological models of self-organisation, the interaction between each individual neuron is modeled using a lateral interaction kernel. This kernel is used in an iterative feedback system to establish the level of excitation of a set of neurons, for an arbitrary input pattern. The most commonly used kernel is a narrow Laplacian; a kernel g for which $g_{ij} < 0, i \neq j$, and $g_{ij} > 0, i = j$. When using an Laplacian kernel, the result of the iterative process is a single neuron which asymptotically approaches saturation value, and all other neurons decaying to a state of no excitation; that is, a single winner takes all. The weights supporting this winning neuron can then be reinforced.

Kohonen's innovation was to replace this iterative process of reinforcement with an algorithmic process of selection. Rather than have a winner selected using this activity control kernel, Kohonen's algorithm substitutes a meta-level decision structure which performs a similar task. In Kohonen's algorithm, a winner is artificially selected, according to some distance metric, and a second lateral interaction kernel — a plasticity control kernel (such as a Gaussian) — is artificially fitted about this winner. This plasticity control kernel is used as the basis for training. Neurons near the winner receive strong support as a result of the large value of the plasticity control kernel, while other neurons receive little or no support. In this way, the winner and its neighbours are reinforced.

This approximation allows Kohonen's SOM algorithm to be computationally simple. Instead of requiring a complicated system of equations balancing individual neuron dynamics and the dynamics of the map as a whole, it can be stated as a single iterative weight update rule. Three significant variants of this weight update rule have been presented in the literature. These variants, plus the overall Kohonen algorithm, will be discussed in Section 2.5.1.

This computational approximation does not significantly undermine the utility of Kohonen's algorithm (as is made evident by the plethora of SOM application literature), but it does limit the extent to which the underlying self-organising process can be analysed. Discontinuities and meta-level decision structures such as the winner selection mechanism make formal analysis of the system difficult. As a result, many aspects of Kohonen's algorithm, such as the use of alternate plasticity control kernels, have not been investigated.

2.4.1 Advantages of Kohonen SOMs

The most immediately apparent advantage of Kohonen's SOM algorithm is a dramatic improvement in computational efficiency. The inner loop which evaluates neuron response is the most frequently performed part of the generic SOM algorithm. By removing this loop, the computational requirements of evaluating and updating a SOM are substantially reduced.

Another key advantage of Kohonen-style SOMs comes in the type of classification provided by the algorithm. Kohonen-style SOMs, and SOMs in general, provide an unsupervised classification of training data. This places SOMs in a different domain to most other neural architectures, which are generally supervised classifiers.

The style of unsupervised classification provided by Kohonen's algorithm is also unique. Most unsupervised classifiers (K-means, decision trees, etc) generate discrete output classifications, or provide a clearly defined discrete output state. While the output state of a Kohonen map is ultimately discrete (in the form of discrete winners on the map), the interpretation of this output does not provide clear class membership — rather, it describes the topological relationships between training patterns. This is extremely useful for problems in which class membership is an ambiguous concept (for example, classifying a person as 'tall' or 'short') or for classifications which span a continuous range. Using a Kohonen SOM, complex relationships in a data space which defy a simple classification scheme can be visualised as a simple 2D spatial representation.

It should be noted that supervised variants of Kohonen SOMs do exist — for example, the Adaptive Subspace SOM (Kohonen *et al.*, 1997), and Learning Vector Quantisation (Kohonen, 1988). These are presented as extensions to Kohonen-style self-organisation which convert the output of an unsupervised self-organising map into an alternate discrete, supervised space. However, these extensions are not of interest for the purposes of this thesis.

Another useful characteristic of Kohonen's SOM algorithm is the relationship between the distribution of training patterns and the distribution of winners on the trained SOM. Once a SOM is trained, the area on the SOM surface which represents a given concept is proportional to the probability density of patterns representing that concept in the training data set. As a result, the resolving power of the SOM is increased for these concepts; that is, the SOM is able to perform much finer discrimination between concepts from a specified region (or regions) of interest. This can be an extremely useful characteristic: an inability to resolve a given concept can be rectified by altering the composition of the training set.

Kohonen's SOM algorithm also has the advantage of constrained memory requirements. Best case and worst case memory requirements are identical, and are directly related to the specified map size. The resolving power of a Kohonen SOM is also linearly related to map size; an improvement in resolving power can be achieved through a linear increase in memory requirements.

2.4.2 Limitations of Kohonen SOMs

The approximations used by Kohonen also place limitations on the topographical solutions that can be expressed by a map. The Winner-Take-All approach prevents multiple peaks from forming on the map. This is a potentially useful feature for describing complex topologies or for maps where redundancy is desirable. The insistence that there must be *exactly* one winner also prevents no peak situations. This is a potentially useful outcome if a training example is unsuited to representation on a map.

An additional limitation of the Kohonen SOM is the restriction to a 2D topological representation of training data. The output of a Kohonen SOM is a 2D map of neurons. Therefore, the topology represented by this map of neurons is limited to approximately 2 dimensions (although under some circumstances, it is possible to represent slightly more than 2 dimensions; for example, a partial representation of a third dimension). If the training data set is formed from a topology which is inherently multidimensional, the SOM will inevitably lose some topological information. If these higher dimensions contain useful topological relations, this could be a significant loss of knowledge from the data set.

Even if a 2D representation is sufficient to represent a given data set, Kohonen's SOM algorithm is unable to guarantee that the topological representation formed by a map will be in any way meaningful. Although an ontological analysis of a data set may reveal a significant underlying topology, Kohonen's algorithm provides no guarantee that this topology will be developed and reinforced on the map. In addition, Kohonen's SOM algorithm is limited to representing topologies which can be expressed in terms of a Euclidean distance between training vectors. Any weakness in the numerical representation of a concept (for example, if Euclidean distance is not a good representation of similarity) will be exhibited as a weakness in the topological representation of that concept on the map.

2.5 Variations on a Theme by Kohonen

The term 'Kohonen Self-Organising Map' encompasses a range of algorithms and algorithmic variants. Kohonen has presented three fundamental variants of SOM representation and updates. In addition, there exists a range of parameter decay schemes and architectural variants on the basic algorithm. Each of these variants have advantages and disadvantages. The significance of these advantages and disadvantages must be

judged on an application specific basis.

2.5.1 Representation and Update

The most notable variants of the Kohonen algorithm have been introduced by Kohonen himself. These variants fundamentally repose the underlying architecture of the SOM, by varying the operation of neurons and the update of weights to these neurons. Since the initial publication of his SOM algorithm, Kohonen has described three major SOM variants. These variants are the original maximal map, the minimal map and a revised, optimised maximal map.

Maximal Maps

The original formulation of Kohonen's SOM (Kohonen, 1982) bears many similarities to the biological model from which the algorithm was drawn. A sheet of I neurons m_i is arranged and fully connected to a series of J inputs x_j in the range $[0:1]$ through a set of independent weights w_{ij}. The output of each neuron is taken in a similar manner to traditional feedforward neural networks:

$$m_i = \sum_j x_j w_{ij} \tag{2.1}$$

For any given pattern, the activation response of every neuron is evaluated. Using this activation level, a winner — the neuron ω with the highest level of activation — is selected:

$$\omega | m_\omega = \max_i (m_i) \tag{2.2}$$

Having selected this winner, all the neurons on the map have their weights updated according to a Hebbian style update rule. Under Hebbian update, the quantity of reinforcement of weights is proportional to the strength of activation on the map. All training patterns have some positive effect on the weights of the map. However, if the training pattern does not engender a strong response, this effect may be minimal. The update rule used in Kohonen's maximal style SOM is defined as:

$$\vec{w}_i(t+1) = \frac{\vec{w}_i(t) + \alpha(t) h_{i,\omega}(t) \vec{x}(t)}{\|\vec{w}_i(t) + \alpha(t) h_{i,\omega}(t) \vec{x}(t)\|} \tag{2.3}$$

In this formulation, $h_{i,\omega}(t)$ is the plasticity control kernel, and $\alpha(t)$ is the learning gain

(a value in the range [0 : 1]). The plasticity control kernel controls the strength of the neighbourhood interaction in the map. It can take one of two forms. The first form is a simple neighbourhood function:

$$h_{i,\omega}(t) = \begin{cases} 1, & i \in N_\omega(t) \\ 0, & i \in \overline{N}_\omega(t) \end{cases} \tag{2.4}$$

That is, if the neuron i is in the set of neighbours of the winner N_ω at time t, update the neuron weights; otherwise, provide no update. Membership of the neighbourhood set is determined by the distance on the map from the winning neuron. The critical distance defining neighbourhood membership (and thus the size of the neighbourhood set) varies over time.

This kernel can also be stated as a simple relation by smoothing the kernel into a continuous function:

$$h_{i,\omega}(t) = exp\left(-\frac{\|\vec{r}_i - \vec{r_\omega}\|^2}{2\sigma(t)^2}\right) \tag{2.5}$$

In this formulation, $\vec{r}_i$ is a vector describing the position of neuron i on the map; $\sigma(t)$ is a function describing the radius of the neighbourhood over time. As a result of this kernel, the weights of every neuron are updated every time a new pattern is presented. However, neurons which fall a large distance from the selected winner will receive minimal update. It is this continuous form of the plasticity control kernel which is most common in the literature.

The Kohonen SOM learning algorithm is based upon the repeated application of this update rule over an entire training data set. The weights of the map are seeded with small random values (to provide some local entropy). The parameters α and σ are given initial values, and are decayed over time, according to a predefined schedule. A wide range of schedules have been suggested. These are discussed further in Section 2.5.2. During each epoch, a random order of presentation is chosen, and each pattern is presented to the update mechanism in this order. The randomisation of the presentation order at the start of each epoch ensures that the learned topology is not a function of presentation order. This process continues for a predefined number of epochs. This learning algorithm is expressed formally in Algorithm 1.

This general learning algorithm is identical for all of the Kohonen variants. The portions of the algorithm specific to Kohonen's original maximal formulation can all

Algorithm 1 The SOMLearn algorithm. This algorithm takes a set of training vectors $\mathbf{X}$, containing N patterns with J inputs, creates a map of $n \times m = I$ neurons, and trains the map for e epochs. Upon completion of training, a trained map $\mathbf{W}$ is returned.

$\mathbf{W} = \mathbf{SOMLearn}(n, m, \mathbf{X}, e)$
Establish initial value of α
Establish initial value of σ
Create $\mathbf{W}$: an array of $n \times m$ neurons, each containing J weights
Instantiate weights in $\mathbf{W}$ with small random values.
for t = 1 to e **do**
  Establish a random order of presentation R for the vectors in $\mathbf{X}$
  for p = 1 to N **do**
    Update($\mathbf{W}$,$\vec{x}_{R(p)}$,$\alpha(t)$,$\sigma(t)$)
  end for
end for
return $\mathbf{W}$

Algorithm 2 The Update portion of the SOMLearn algorithm: original maximal map version. This algorithm updates the weights $\mathbf{W}$ in the map, based upon the training vector $\vec{x}$, and the current values of the learning parameters α and σ. $h_{i,\omega}$ is as defined in Equation 2.4 or 2.5.

Update($\mathbf{W}, \vec{x}, \alpha, \sigma$)
for i = 1 to I **do**
  $\Theta_i = \vec{w}_i(t) \cdot \vec{x}$
end for
$\omega = \max_i \Theta_i$
for i = 1 to I **do**
  $\vec{w}_i(t+1) = \vec{w}_i(t) + \frac{\vec{w}_i(t)+\alpha h_{i,\omega}\vec{x}(t)}{\|\vec{w}_i(t)+\alpha h_{i,\omega}\vec{x}(t)\|}$
end for

be found in Algorithm 2.

One critical limitation of Kohonen's original formulation is the requirement that inputs must be normalised to the range [0:1]. Even within this range, there are significant restrictions placed upon problem encoding. Kohonen's maximal map relies upon a direct correspondence between magnitude and significance. A large value, be it in input, weight, or output, is interpreted as a significant value; similarly, small values are interpreted as insignificant. This poses problems when encoding certain data types. A significant negative cannot be encoded as 0; such an encoding attributes no significance to the input. A continuous range cannot be simply scaled, as low values will be interpreted as being less significant, rather than just numerically smaller.

These restrictions on problem encoding can prove a significant impediment to the representation of certain problem types. Techniques have been found to overcome some of these representational restrictions, some of which are used in this thesis. However, these techniques generally rely upon encoding tricks which increase the number of

inputs to the map, thus increasing computational complexity.

However, the largest problem associated with Kohonen's maximal map is the computational requirement of the algorithm. Although Kohonen's maximal map is able to form good topological representations of complex data sets, it is extremely slow. In the original publication, training times of O(10000) epochs were quoted for simple 8×8 grid patterns. Although the time required to perform an epoch of training is substantially smaller than the requirement in biological SOM models, the number of epochs which must be computed remains large. This fact often renders impractical the training of complex training sets. While this original algorithm demonstrates the technical feasibility of self-organisation, significant improvements in the efficiency of the algorithm are required.

Minimal Maps

Ritter and Kohonen (1989) addressed this concern by significantly restating the underlying neuron concept. Whereas the original Kohonen formulation generates winners from the maximum excitation, this revised formulation describes winners in minimal terms. The activation of a neuron is then defined as:

$$m_i = \sum_j |x_j - w_{ij}| \tag{2.6}$$

For any given pattern, the activation response of every neuron is evaluated. A winner with the lowest level of activation is then selected:

$$\omega | m_\omega = \min_i (m_i) \tag{2.7}$$

That is, the algorithm selects as winners those weights which explicitly mimic training patterns.

As a result of this formulation, the update rule becomes significantly simplified. The minimal algorithm has a desirable final state: a weight representation where $m_i = 0$. A pattern/weight combination which generates this value will be an ideal representation. The weight update rule can therefore update weights directly towards this ideal value. The maximal algorithm has no analogous state; weights are reinforced if they engender a response in the map, but there is no theoretically ideal weight. Therefore, the search process for an ideal weight representation is somewhat slower.

The update rule for the minimal algorithm can be stated in a simple fashion as a

Algorithm 3 The Update portion of the SOMLearn algorithm: minimal map version. This algorithm updates the weights $\mathbf{W}$ in the map, based upon the training vector $\vec{x}$, and the current values of the learning parameters α and σ. $h_{i,\omega}$ is as defined in Equation 2.4 or 2.5.

Update$(\mathbf{W}, \vec{x}, \alpha, \sigma)$
for i = 1 to I **do**
  $\Theta_i = \|\vec{w}_i - \vec{x}\|$
end for
$\omega = \min_i \Theta_i$
for i = 1 to I **do**
  $\vec{w}_i = \vec{w}_i + \alpha h_{i,\omega}(\vec{x} - \vec{w}_i)$
end for

direct attempt to attain the ideal goal:

$$\vec{w}_i(t+1) = \vec{w}_i(t) + \alpha(t) h_{i,\omega}(t)(\vec{x} - \vec{w}_i(t)) \tag{2.8}$$

Each update attempts to direct the weights towards the known goal $m_i = 0$; the magnitude of any change is moderated by the gain and neighbourhood parameters α and σ.

In all other respects, the minimal SOM algorithm is identical to the maximal algorithm. The basic learning algorithm presented in Algorithm 1 holds; however, the Update algorithm must be altered to accommodate the new representation of neurons and update of weights. This variation is shown in Algorithm 3.

Algorithm 3 is the most commonly used Kohonen variant. Training times using this algorithm are significantly improved over the original maximal algorithm. This improvement can be attributed to the improved weight update mechanism.

This algorithm also has the advantage of allowing a much wider variety of inputs. The maximal style SOM requires normalised inputs to ensure no component dominates the overall activation level. However, the minimal style SOM has no such requirement. All inputs attempt to attain a difference of 0, and the relative magnitude is eliminated by subtraction. As a result, boolean values and numerical magnitudes can both be directly encoded, usually as single inputs. This representational freedom allows the SOM trainer to define an extremely efficient problem representation; a fact which further improves training times.

However, the minimal SOM does suffer from some problems. The encoding scheme utilised by maximal style maps determines the significance of a weight from its magnitude. That is, a large weight indicates a significant pattern. However, using minimal

SOMs, it is impossible to determine the significance of a weight without reference to a specific pattern. The weight encoding by itself is meaningless — it requires subtraction from a training vector to establish significance. This poses a major problem for those attempting a post-training analysis of a SOM, as the weight space of the SOM cannot be analysed without reference to a specific data set. The characteristics of the map cannot be established independent of the data set used to create it.

The minimal SOM algorithm also poses restrictions from a research perspective. As more is learned about the structure and function of self-organising structures in the brain, it would be desirable to map these new theories into computational practice. However, the computational advantages gained by the algorithm are achieved by abandoning the original biological metaphor. The application of new, biologically derived modifications to the self-organising process is therefore extremely difficult. While efficient, the minimal SOM is often inappropriate for research into the fundamental nature of self-organisation.

Maximal Maps Revisited

Kohonen and Hari (1999) presents a revised version of the maximal map. As a maximal map, the activation levels are calculated using Equation 2.1, and a winner is chosen as the neuron with the highest activation as in Equation 2.2. The only difference between this algorithm and the original maximal map is the update rule:

$$\vec{w}_i(t+1) = \vec{w}_i(t) + \alpha h_{i,\omega}(t)[\vec{x}(t) - [\vec{w}_i(t) \cdot \vec{x}(t)]\vec{w}_i(t)] \tag{2.9}$$

This update rule operates on the principle that the output product of a weight and an input vector will be maximised if the weight and input vector have a high degree of correlation. Each update using this rule attempts to increase this degree of correlation. Uncontrolled growth in weight value is controlled due to the normalising influence of the dot product $\vec{w}_i(t) \cdot \vec{x}(t)$ term.

Again, the basic Kohonen training algorithm (Algorithm 1) holds, but the Update algorithm must be altered to reflect the new weight update rule. This revised Update algorithm can be found in Algorithm 4.

This approach to weight update performs a much more direct convergence towards the unknown maximum than the original maximal update rule. The original update rule attempts to reach an optimal representation by slow accumulation of minute vector components. This slow approach is necessitated by the absence of any negative rein-

Algorithm 4 The Update portion of the SOMLearn algorithm: revised maximal map version. This algorithm updates the weights $\mathbf{W}$ in the map, based upon the training vector $\vec{x}$, and the current values of the learning parameters α and σ. $h_{i,\omega}$ is as defined in Equation 2.4 or 2.5. Note that this algorithm is identical to Algorithm 2, except for the weight update rule.

Update$(\mathbf{W}, \vec{x}, \alpha, \sigma)$
for i = 1 to I **do**
  $\Theta_i = \vec{w}_i \cdot \vec{x}$
end for
$\omega = \max_i \Theta_i$
for i = 1 to I **do**
  $\vec{w}_i = \vec{w}_i + \alpha h_{i,\omega}(\vec{x} - \Theta_i \vec{w}_i)$
end for

forcement in the update algorithm; there is no mechanism for correction if the weight representation overshoots an optimal representation. The revised update rule allows weight subtraction, which subsequently permits overshoot and correction to occur in the search for an optimal weight representation.

While still not explicitly directed towards a goal (maximal SOMs, in either formulation, have no such specific goal), this update rule does converge quickly upon solutions. The number of epochs required to train this updated maximal map is roughly equivalent to that observed using minimal maps. As a maximal map, direct analogies can be drawn between biological SOM architectures and the maximal algorithm, allowing the application of biological principles to the computational map.

However, the restrictions on problem formulation imposed by maximal maps still exist. Inputs to maximal maps must be normalised to the range $[0 : 1]$ and must represent some concept of 'support'. Whereas a value of 1 indicates positive support for a concept, a value of 0 does not indicate positive support against a concept. Rather, it represents an absence of any support, either positive or negative. True/False relationships, for example, cannot be represented as a simple 1/0 encoding; 'false' must be represented as a 'positive false', not as 'the opposite of true'. Simple binary representations, or scale normalisation of numeric inputs to the $[0 : 1]$ range are therefore not possible. Concepts must be encoded so that input magnitude has a direct relationship to significance. These limitations can be overcome but only at the cost of an increase in the number of inputs. This increases the computational complexity of learning.

Despite these restrictions, this revised maximal algorithm provides a comfortable middle ground between the original maximal algorithm and the heavily optimised minimal algorithm. It is this revised maximal algorithm that is used throughout this thesis.

2.5.2 Parameter Decay Schemes

After modifications to the fundamental algorithms for representation and update, the most significant variations made to the basic Kohonen algorithm revolve around the manipulation of the key learning parameters.

The operation of Kohonen's algorithm is based upon the control of two parameters during training: learning gain (α) and neighbourhood size (σ). During the learning process, these parameters are decayed. However, the manner in which these parameters are decayed can have a profound influence on the style of organisation that develops. In addition, these parameters control the number of training epochs taken to achieve a complete map. Decay schemes which overestimate the required number of training epochs can significantly prolong training time and increase the computational requirement of the SOM algorithm.

Gain Decay

The vast majority of literature on parameter decay has concentrated upon the appropriate selection of a gain decay mechanism. The purpose of gain in the update equation is to moderate the rate of update: a high gain value allows the weights to respond rapidly to different inputs, whereas low gain limits the extent of any change. Decaying gain over time allows a SOM to rapidly adjust to training during the initial stages of learning and slowly adapt to the finer details towards the end of training.

At the time of initial publication, Kohonen suggested simple linear decay mechanisms for parameter decay. Using such a scheme, gain was decayed from the maximum value of 1, to a minimum value of 0, over some predetermined time frame. However, the selection of this time frame is a difficult task. Selecting a time frame which is too short could result in a badly organised map. However, selecting a time frame which is too long wastes processor cycles.

The general approach to selecting this decay period has been to use empiricism and past experience to guide the selection of values. However, if a data set is extremely large, or is not available *a priori* (such as would be the case in a real-world learning situation), empirical testing is not a practical option. A wide range of simple algebraic alternatives to linear decay have been proposed in the literature. For example, Kohonen (2001) proposes:

$$\alpha(t+1) = \frac{\alpha(t)}{1+\alpha(t)} \tag{2.10}$$

$$\alpha(t+1) = \frac{\alpha(t)}{1 + h_{i,\omega}(t)\alpha(t)} \tag{2.11}$$

$$\alpha(t+1) = \frac{A}{t+B} \tag{2.12}$$

where $\alpha(0)$, A and B are 'suitably chosen constants'. Ritter *et al.* (1992) proposed an exponential decay curve:

$$\alpha(t) = \alpha_0 \exp\left(-\frac{1}{\tau}\right) \tag{2.13}$$

where τ is a decay parameter and α_0 is the initial gain. Kang *et al.* (1995) proposed a hyperexponential decay curve:

$$\alpha(t) = \frac{ABt}{T} \exp\left(\frac{-Bt}{T}\right) \tag{2.14}$$

where A is an amplitude adjustment, B is a slope determining constant and T is the total training time.

These decay schemes do not obviate the need for parameter estimation — they merely change the parameter which must be estimated. The process of choosing suitable constants is largely unexplored. Of the commonly used general purpose parameter decay schemes which exist in the literature, none are accompanied by general purpose mechanisms for selecting appropriate values of key decay parameters.

Given the difficulty in evaluating decay parameters *a priori*, some attempts have been made to use automated runtime schemes for parameter decay. One example of such a runtime scheme is the Auto-SOM (Haese, 1998). The Auto-SOM algorithm attempts to use a Kalman filter to recursively update SOM parameters during learning. The Auto-SOM system consists of two process models: one of the learning process and one of the organising process. The Kalman filter corrects these models based upon errors between the predicted map output and the actual map output. By comparing the Kalman filter state equation of the learning process with the weight update rule for each neuron, it is possible to evaluate a gain value for each training epoch. Using this model, each neuron has an independent learning gain.

However, complex schemes such as Auto-SOM can be extremely computationally complex. The Auto-SOM Kalman algorithm has complexity of $O(IJ)$, where I is the number of neurons and J is the number of inputs (Haese, 1998). Since this Kalman process must be evaluated after each training epoch, this can be a significant computational requirement if the training regime is prolonged. This is exacerbated by the

fact that the Auto-SOM algorithm errs on the side of caution by converging slowly; training times in the order of $O(10^4)$ epochs are quoted in Auto-SOM literature (Haese and Goodhill, 2001).

Neighbourhood Decay

Although many attempts have been made to find an optimal decay mechanism for gain in Kohonen style maps, there have been comparatively few investigations into neighbourhood decay. This is exemplified by the cursory analysis of the topic provided by Kohonen in his seminal book (Kohonen, 2001). In the short analysis presented in this source, Kohonen suggests that as long as the neighbourhood is initially large (a value the same order of magnitude as the largest dimension of the neuron array) there should be no problems obtaining a stable topological mapping of a data set. However, he provides no guideline as to an appropriate rate of decay.

The only investigations into neighbourhood decay seem to take the form of basic restrictions on neighbourhood decay, rather than specific decay schemes. Erwin *et al.* (1992b) found that if the neighbourhood function was convex, the only stable states would be ordered ones. Therefore, as long as a neighbourhood function such as a Gaussian with a 'large' standard deviation is used, "ordering can be assured almost surely" (Erwin *et al.*, 1992b). The only exception presented to this statement is the special case of a SOM with an input signal of similar dimensionality to the SOM array. However, such an arrangement would represent an extremely unusual SOM; in practice, most SOMs have many more neurons than inputs.

In practice, most SOM practitioners follow the lead of Kohonen (1982) and use a simple linear or geometric decay of neighbourhood size, mostly because practice has shown such decay schemes to be "good enough" (Ritter *et al.*, 1992). However, this practice is endowed with the same problems as linear gain decay, as an appropriate decay period must be found. One simple approach to selecting this decay period is to ignore it by selecting a constant value for neighbourhood size. This approach is particularly common in theoretical analyses, as it dramatically simplifies the analysis of the organisation process.

The runtime approach used to evaluate gain can also be used to update neighbourhood size. For example, the Auto-SOM technique discussed in the previous section models neighbourhood size in addition to learning gain. The Auto-SOM algorithm uses a measure of local topological preservation to evaluate the preservation of input

topologies. As this preservation increases, neighbourhood size is decreased. As with the gain parameter, the neighbourhood for each neuron is independent. However, as discussed previously, the Auto-SOM technique is computationally complex, and does produce extended training times.

Discussion

The vast majority of SOM literature treats gain as the most significant factor affecting SOM organisation, leaving neighbourhood decay as a relatively ancillary concern. Mulier and Cherkassky (1994), for example, describe the choice of gain as 'critical for good performance'. However, no proof has been offered that this is indeed the case. Theoretical analyses of self-organisation largely ignore the neighbourhood question, dealing with idealised single neighbour neurons or static neighbourhood relationships (Flanagan, 1998) (Martin-Smith *et al.*, 1993) (Cottrell *et al.*, 1998). The decay of neighbourhood size in self-organisation is one area of research which calls for much further investigation.

2.5.3 Neighbourhood Variants

The traditional neighbourhood function (Equation 2.5) is tightly bound to a geometric interpretation of the neighbourhood. The SOM is assumed to be a rectangular or hexagonal plane of evenly spaced neurons. The physical distance between neurons on this plane is then used to evaluate the strength of any neighbourhood interaction. A neuron which is physically close to a winner will receive strong support during an update cycle, while other neurons which fall further away on the plane will receive minimal support.

However, there is no requirement for this geometric organisation to be preserved. The original formulation of neighbourhood (Equation 2.4) defines the 'neighbourhood' of a neuron as the members of a set; geometry is merely a simple way of determining membership of the neighbourhood set. Any other scheme can be used, and can result in some interesting topological structures.

Amongst the simplest variations to the neighbourhood set is to add neighbourhood relationships to the existing planar geometrical relationships. By allowing neurons near the edge of a map to be considered neighbours, the planar surface of the SOM effectively becomes the surface of a cylinder or sphere (depending on which edges are wrapped). In this way, data which has a looped structure need not break the loop in order to find

a 2D topological representation.

The removal of neighbourhood relationships is also possible. By introducing boundaries in the SOM over which geometric relationships are not preserved, the simple 2D SOM plane can be transformed into a network of loosely attached local neighbourhoods. If these neighbourhoods are completely detached, the SOM algorithm becomes a classifier, with each isolated neighbourhood corresponding to a classification category.

The neighbourhood set can also be extended into multiple dimensions. A number of authors have suggested the use of a SOM whose neurons form the vertices of an N-dimensional hypercube (Bauer and Villmann, 1997) (Truong, 1991) (Wan and Fraser, 1993). All geometric neighbourhood relationships familiar to the 2D case are then abstracted to the N-dimensional case, allowing for more complex geometric relationships to be represented in the output data set. This hypercubical topology can also have neighbourhood relationships added and removed, allowing for topological loops and boundaries in multiple dimensions.

The technique of adding and removing neighbourhood relationships is not a recent discovery. Indeed, the possibility of developing eclectic neighbourhood relationships was part of Kohonen's initial discussion of his algorithm (Kohonen, 1982), and was demonstrated later with triangular and 'cactus' shaped topologies (Kohonen, 1990). This technique has also been used successfully in a practice, such as in the clustering of data from satellite imagery by Koikkalainen and Oja (1990).

Using these neighbourhood variants, it is possible to improve the quality of topologies formed by complex data sets on a SOM. It stands to reason that a data set which has an underlying 'widget' topology will fit on a map with 'widget-like' neighbourhood relationships better than it will on a rectangular map. Providing a map whose topological structure suits your data is logical step.

Developing an appropriate set of neighbourhood relationships to represent a complex map topology poses a problem. In order to create a neighbourhood structure which suits the data, the user must have a good knowledge of the underlying topology of the data set. However, the purpose of Kohonen's SOM algorithm is the visualisation and discovery of the topologies in a data set. Therefore, topological information of the kind required to create an appropriate neighbourhood structure will not be known until after organisation has occurred. If sufficient topological knowledge *is* known *a priori*, there is little value (beyond novelty) in using a self-organising process to expose the underlying topology.

Another approach to neighbourhood decay is to consider the neighbourhood decay of each neuron to be independent. In this way, the resolution of the map is not a constant (as is the case with simple arithmetic or geometric decay), and local resolution decays at a rate determined by the difficulty of finding a good local topological representation. The AdSOM algorithm (Kiviluoto, 1996) is one SOM model which implements this idea. Using this algorithm, neighbourhood size is lowered exponentially, but is increased if a local topological discrepancy is found.

However, determining the existence of a local topological discrepancy relies upon an accurate measure of local map quality. Such a measure can be difficult to establish (as will be discussed in Section 2.6). If topological errors are overestimated, dynamic techniques such as AdSOM will never decay neighbourhood size or will finish training while some regions of the map still have a large neighbourhood size. This is exacerbated by the fact that topological errors are more likely to occur in the mapping of complex data sets — the same data sets which benefit most from a SOM visualisation tool. Maps which contain zones with large neighbourhood sizes are of limited use for visualisation purposes, as the neighbourhood size directly relates to the precision of the output representation.

2.5.4 Other Variants

The variants presented here are by no means exhaustive: Kohonen (2001) cites 46 of the more interesting extensions to the original SOM algorithm; Kangas and Kaski (1998) cites a total of 3043 works which have used, refined or provided variants upon Kohonen's basic SOM algorithm.

There are, however, some common themes in SOM research. The most common of these themes are summarised by Kohonen (2001):

Different matching criteria which redefine the 'winner' function;

Accelerated searching for winners using traditional optimisation methods such as linear programming or heuristic search to speed the search for the winning neuron;

Hierarchical searching, which simplify the SOM problem by breaking a large SOM into a tree structure of smaller SOMs;

Accelerated learning of SOM weights;

Handling sequential signals such as temporal sequences;

Supervised SOM, which provide methods for clustering outputs, and labeling those clusters;

Systems of SOMs, where large networks of SOMs are interconnected, with the output of one SOM becoming the input to another.

A full exploration of these other variants is beyond the scope of any single study. However, the wide range of modification themes demonstrates that Kohonen's SOM algorithm is a highly adaptable and extensible algorithm which can and has been modified to represent many new ideas in machine learning.

2.6 Assessing SOM Quality

Topological descriptions are not created equal — it is easy to conceptualise 'good' and 'bad' mappings of a given data set. For example, the topology of a simple 2D data set should be clearly preserved on a 2D SOM. Failure to do so should be considered a failure of the algorithm. Given that there is a conceptual metric of 'good' and 'bad' topological representations, it would be desirable to have a quantitative description of the 'goodness' of a given topological description. If such a metric were to exist, it would be possible to compare the performance characteristics (such as quality of topology, organisation time to reach a given quality, etc) of two variants of the SOM algorithm.

Unfortunately, the nature of the SOM complicates the definition of such a measure. The output state of a SOM is a qualitative description, consisting of a topology describing the key relationships present in a training data set, combined with some measure of the relative importance of those concepts. While this is the ideal output for a visualisation tool, it poses problems for the researcher attempting to quantify the performance of a SOM algorithm.

Unlike most supervised and unsupervised learning algorithms, the SOM does not produce discrete output classifications. Consequently, it is not possible to use the traditional approach of counting false positives and false negative in the classification of a test set. Developing an accurate and simple quantification of the continuous nature of a SOM representation is a non-trivial task. This task is made more difficult by the fact that there is no natural entropy description for the weight space of a SOM (Erwin *et al.*, 1992a). If such an entropy description were to exist, it would be trivial to quantify this entropy, and use it for optimisation purposes (Heskes, 1999).

The difficulty associated with developing a topological quality metric has not prevented a large number of authors attempting to develop such a metric. A number of methods have been presented in the literature for analysing the performance of topological organisation.

2.6.1 Quantisation Error

One of the simplest methods for evaluating SOM performance was suggested by Kohonen as an extension of error measurement used in the Vector Quantisation (VQ) techniques he pioneered (Kohonen, 2001). The weights of a minimal SOM can be considered to be a set of reference vectors. If a training pattern corresponds exactly to the weights of a given neuron, a perfect activation (i.e., distance 0) will occur. However, such perfect mappings will be rare. The extent to which perfect mappings do not occur is one measure of map performance. Formally, given a training set $\mathbf{X}$ consisting of N training patterns $\vec{x}_j$, which generate winning neurons with indices $\omega(j)$, Kohonen's quantisation error can be stated as:

$$E = \sum_{j=1}^{N} \|\vec{x}_j - \vec{w}_{\omega(j)}\|^2 \tag{2.15}$$

While this technique does give a measure of the extent to which good winners are found for training patterns, it ignores the concept of neighbourhood preservation. Under a good topology, two similar vectors should fall close together on the output map. Conversely, two significantly different vectors should fall some distance apart. Using the quantisation error metric, there is no bonus for preserving topological relationships in the data set, nor is there a penalty for losing existent topological relationships or introducing false relationships. This is a significant limitation, as the topology preserving characteristic of the SOM is its most desirable characteristic.

2.6.2 Neighbourhood Quantisation Error

This limitation can be overcome by adding the neighbourhood of a winner to the quantisation calculation. This variation, also suggested by Kohonen (2001) can be expressed as:

$$E = \sum_{i=1}^{I} \sum_{j=1}^{N} \|(\vec{x}_j - \vec{w}_i)h_{i,\omega(j)}\|^2 \tag{2.16}$$

Using this variation, the quantisation error is computed as the sum squared distance between the training vector and all the vectors in the neighbourhood of the winner. In this way, local topological continuity is added to the quantification, as a low quantisation error will only occur if the neighbourhood of the winner supports the training pattern in the same way as the winning neuron.

One significant limitation of this technique is that it only works on minimal SOMs. The weights of a maximal SOM do not attempt to match input patterns, so subtraction of maximal style weights from an input will not give any useful measure. If a maximal style map is to be used (as is the case in the remainder of this thesis), quantisation error of this form is not a useful measure.

However, the greatest limitation of this technique is the significance placed upon the choice of neighbourhood size. Choosing a neighbourhood which is too small will not encompass enough topological information to adequately quantify overall topological performance. However, choosing a neighbourhood which is too large will result in misleading measure of topological performance. Large scale variations in the map will be considered an indication of a poor topological mapping, rather than a required characteristic of a map which is to represent a large data domain.

The existence of a tunable parameter which has such a critical role in determining map quality is problematic, as there is no method for determining if a poor quality score is the result of a bad map, or a poor parameter selection. Viable quality metrics should avoid such parameters.

2.6.3 Ideal Map Distance

Another simple quality metric was proposed by Lo and Bavarian (1991). This scheme attempts to measure, for each pattern in the test set, the distance on the map between the between the winning neuron for that pattern, and the 'ideal' position of the winning neuron. The error in the map is then taken as the sum square distance between weights and the 'ideal' grid position.

This technique presupposes that the user knows the ideal position for each winner. In demonstrating their technique, Lo and Bavarian (1991) used a square grid — the ideal winners for this topology are easy to establish. However, this will not be the case for the vast majority of interesting data sets. Furthermore, if the ideal position is known, or can be easily computed, there should be no need for a SOM exploration of input topology as the user has already determined the topology of the data set.

2.6.4 Topographic Product

Bauer and Pawelzik (1992) have suggested the use of the topographic product as a measure of map quality. Topographic product is a measure of the preservation of neighbourhood relations between spaces of possibly different dimensionality. Its use as a technique is not limited to self-organising maps; it is also used in the context of nonlinear dynamics and time series analysis (Liebert *et al.*, 1991), although it is referred to as "wavering product" in these areas.

The topographic product $\mathcal{P}$ is evaluated to be:

$$\mathcal{P} = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N-1} \log \left(\Pi_{l=1}^{k} Q_1(j,l) Q_2(j,l) \right)^{\frac{1}{2k}} \tag{2.17}$$

$$Q_1(j,k) = \frac{\|\vec{w}_j - \vec{w}_{n_k^X(j)}\|^2}{\|\vec{w}_j - \vec{w}_{n_k^M(j)}\|^2} \tag{2.18}$$

$$Q_2(j,k) = \frac{\|\vec{x}_j - \vec{x}_{n_k^X(j)}\|^2}{\|\vec{x}_j - \vec{x}_{n_k^M(j)}\|^2} \tag{2.19}$$

In these expressions, $n_k^M(j)$ denotes the index of the kth nearest neighbour of neuron j in the map space, and $n_k^X(j)$ denotes the index of the kth nearest neighbour of training sample j in the input space. The ratios $Q_1(j,k)$ and $Q_2(j,k)$ represent a comparison between nearest neighbours in the map space and the input space, respectively. These ratios will equal 1 if and only if the kth order nearest neighbours in the input and output space coincide. Any deviation from 1 indicates a violation of the nearest neighbour ordering on the map.

The topographic product suffers from two significant problems. Firstly, the measure only works on minimal maps. The reasons for this are identical to those discussed previously. Secondly, as noted by Villmann *et al.* (1994), the topographic product is unable to distinguish between the folding of the map along nonlinearities in the data space and folding within the data space itself. While the former is a desirable characteristic in a map, the latter is a cause of discontinuities in the output map. A good quality metric should be able to distinguish between these two cases, ignoring the former, but counting the latter as a negative quality.

2.6.5 Topographic Function

A somewhat more sensitive technique was proposed by Villmann *et al.* (1997). This, known as the topographic function, compares the neighbourhood relationships between

receptive fields on the map. The receptive field of a neuron M is the range of patterns in the input space which will cause M to be selected as a winner; that is, cells in the Voronoi tessellation of the map units within the data manifold. On a well ordered map, only units that are neighbours on the map may have adjacent receptive fields.

The topographic function $\Phi_X^M(d)$ is defined as:

$$\Phi_X^M(d) = \sum_i f_i(d) \tag{2.20}$$

$$f_i(d) = \#\{j | \|i-j\|_{\max} > d; R_i \cap R_j \neq f_i\emptyset\} \tag{2.21}$$

The function $f_i(d)$ determines the number of units j which have receptive fields R_j adjacent to R_i, while at the same time have a distance on the map larger than d. In this formulation, $\#X$ is the cardinality of the set X, and the maximum norm $\|x\|_{\max} = \max_i |x_i|$. The value $\Phi_X^M(d)$ is then computed for d values in the range $[1..I]$.

This technique is a powerful measure of topological consistency in a SOM. It is not immediately restricted to use on minimal SOMs, as Voronoi tessellation and receptive fields can be computed on the output space of a SOM without consideration of the underlying weight space. The function can also be normalised to allow comparison between maps and input spaces of differing sizes.

However, this technique is sensitive to noise or misrepresentative inputs. Such inputs may cause poor selection of receptive fields, causing a poor judgement of adjacency. Noise is a consistent problem with real world data sets. While the topographic function may be useful for noise free data sets, the suitability of this technique for real world applications must be questioned.

2.6.6 Topographic Error

Another measure of topographic integrity is Topographic Error, proposed by Kiviluoto (1996). In contrast to the complexity of computing receptive field tessellation, Topographic Error is a simple calculation. It computes the proportion of samples for which the nearest and second nearest units reside in non-adjacent positions on the map.

The topographic error ε_t is defined as:

$$\varepsilon_t = \frac{1}{N}\sum_{i=1}^{N} u(\vec{x}_i) \tag{2.22}$$

$$u(\vec{x}) = \begin{cases} 1, & \text{iff winner and runner up neurons for x are non-adjacent} \\ 0, & \text{otherwise} \end{cases} \tag{2.23}$$

While this measure is not as susceptible to noise as previous methods, it does not consider the extent of discontinuities. Given two similar points in the input space, there is no difference between mapping them one neuron apart, or to opposite corners of the map. Kiviluoto justifies this behaviour by claiming that the Topographic Error allows for the distinction between a small number of major discontinuities and a large number of minor discontinuities.

However, there are a large number of situations for which this behaviour is more of a hindrance than a help. If a data set is sparse, winners will never be adjacent (as they will fall 2 or more neurons apart). In such a case, the topographic error will report a bad score even if the topology is preserved perfectly. Fundamental limitations of this kind render the topographic error largely unusable as a quality metric.

2.6.7 Path Integral Goodness

The measure proposed by Kaski and Lagus (1996) is based upon the average Euclidean distance between the data vector $\vec{x}$, the weight vector of the winning neuron $\vec{w}_\omega$, and the minimum sum of differences between pairs of neighbouring weight vectors that lie on a path from the winning neuron to the second best neuron match on the map. The map goodness C is expressed as:

$$C = E\left[\|\vec{x} - \vec{w}_{\omega(\vec{x})}\| + \min_i \sum_{p=0}^{P-1} \|\vec{w}_{I_i(p)} - \vec{w}_{I_i(p+1)}\|\right] \tag{2.24}$$

where $\omega(\vec{x})$ is the index of the winning neuron for training pattern $\vec{x}$, $\omega'(\vec{x})$ is the index of the runner up neuron, and $I_i(j)$ is the index of the jth neuron on a path of P neurons from the winning neuron to the runner up neuron; in this way, $I_i(0) = \omega(\vec{x})$, $I_i(1)$ is the index of the first neuron on the path, and so on to $I_i(P) = \omega'(\vec{x})$, the index of the Pth neuron on the path, which is the runner up neuron. $E[\cdot]$ is the expectation value over all winners.

This measure evaluates both continuity in the mapping, and quantization error (the Euclidean distance between the training vector and the weight vector) and can be applied to any data set. However, it does require a minimal map, as the computed path is essentially a path integral through the data space through the reference vectors stored as weights. This path cannot be easily computed if the map is maximal.

A similar line integral method which does not required a minimal map was proposed by Kraaijveld *et al.* (1992). In this method, the path is determined by using training

vectors as the reference vectors on the path integral. Each neuron on the path through the map is assumed to have been selected as a winner; the training vector which caused this selection is used as a reference vector on the integral path in the data space. However, this technique assumes that every neuron will be selected as the winner for at least one pattern in the input space. Such correspondence cannot be guaranteed, as discontinuities in mappings will frequently result in map neurons which are never selected as winners. Although this allows the measure to be used on maximal SOMs, the requirement for 1-to-1 correspondence is unrealistic, especially with large data sets which are prone to topological errors.

Another limitation of path integral techniques is the computational requirement. In order to compute the goodness for a given training vector, every path from the winner to runner up must be considered. Furthermore, this path must be computed for every pair of training vectors in the training set. In the worst case, this represents a pair of multiplied exponential complexities. Statistical sampling of training patterns and an informed search for the minimal path will reduce this complexity. However, if this metric is to be computed continuously during training, this will impose a significant slowdown on the training process.

2.6.8 Discussion

The concentration in the literature on minimal maps has lead to the vast majority of quality metrics relying upon the use of the weights as reference vectors. While this can lead to some useful measures of topological quality for minimal maps, they cannot be used on the maximal maps which are used throughout this thesis.

The more promising approaches are the methods which ignore the weight space and rely upon the output space as the true indicator of topological quality, such as those proposed by Kiviluoto (1996), Kraaijveld *et al.* (1992) and Villmann *et al.* (1997). While these specific approaches suffer from significant problems, the general approach of considering the output topology and its relationship to the input data space bears the greatest promise of yielding a general measure for map quality.

2.7 Approaches to Continuous Learning

2.7.1 What is Continuous Learning?

The real world is a difficult thing to model. Real-world problems are nonlinear, as the mathematical relationships underlying real-world problems are not a simple linear combination of factors. Furthermore, real-world problems are time invariant; that is, the statistical properties of real-world systems change over time. Although theoreticians often assume the existence of linear systems with time invariant characteristics, the vast majority of interesting (i.e., hard) problems in machine learning remain nonlinear and time variant.

Real-world process control, for example, is both nonlinear and time variant. Not only does the process change continuously in a nonlinear manner but the characteristics of the process control problem are constantly changing due to aging, exchange of components, and changes in environmental conditions. Many successful process control models have been developed which assume a time invariant process. However, the lifespan of models which make this assumption is inherently limited to a time frame over which the variation in the process is constrained.

The difficulty associated with developing algorithms which can learn in time variant environments has not prevented a large number of attempts to develop such algorithms. However, there is some confusion in the literature over the terms of reference to be used in describing approaches to problems of this sort. All of the following terms have been used to describe solutions to this general class of problem:

- on-line learning
- incremental learning
- sequential learning
- adaptation
- life-long learning

Unfortunately, these terms are not used consistently. Although they are sometimes used to refer to solutions for nonlinear, time variant problems, they are also commonly used to refer to solutions for linear, time invariant problems. For example, 'On-line learning' is often used to refer to pattern-by-pattern update as opposed to batch update. 'Adaptation' is sometimes used to refer to weight (or parameter) change in general,

which is part of any learning method. 'Life-long learning' often refers to the problem of transferring domain knowledge between similar tasks. When used in these contexts, these terms describe important components of a solution to time invariant problems, but they do not encompass the problem as a whole.

The ambiguity of the existing terms lead to Protzel *et al.* (1998) dividing all learning approaches into two categories:

1. systems which allow learning during a certain time interval (the training period), after which the learning system is put into operation. At some point there might be a new training period on an updated (usually large) data set; and

2. systems which allow learning takes place all the time in an uninterrupted fashion; there is no difference between periods of training and operation. Learning and operation can both commence after the first training pattern is presented.

Based upon this division, Protzel *et al.* (1998) defined the term *continuous learning* to refer to the second class of learning problem. Systems which learn in this manner are particularly useful for time variant problems, as they constantly learn from their environment. If the environment changes, these changes should be reflected in the learning system. This thesis concerns itself with an approach to continuous learning which can be used to solve time variant problems.

It is interesting to note that continuous learning is a characteristic inherent in many biological learning systems, including the human brain. Psychological studies have shown that humans often employ more than just initial training data for the purposes of generalisation. In addition, it is sometimes possible to generalise from a single training example (Ahn and Brewer, 1993) (Moses *et al.*, 1993). A prominent example of this phenomenon is the recognition of faces (Beymer and Poggio, 1995) (Lando and Edelman, 1995). When learning to recognise a person, a human does not rely solely upon an initial static training set of faces; if the individual is new, the new face is incorporated into existing knowledge. Furthermore, the learning process involves a self-clarification of the learning task itself. With experience, humans learn that certain facial characteristics, such as eye shape, are more important than other characteristics, such as facial expression (Thrun, 1996).

2.7.2 Naïve Approaches to Continuous Learning

At a simplistic level, almost any algorithm which learns on a pattern-by-pattern basis can be thought of as a continuous learning algorithm. A traditional pattern-by-pattern training algorithm can be considered to be an update algorithm which acts upon a single training pattern drawn from a static distribution. This update process is then repeated until the system is considered to have converged (according to some system specific measure).

However, most algorithms do not *require* that the probability distribution be static. Neither do they prevent the use of the system for predictive purposes before the system has converged — although the quality of predictions made prior to convergence cannot be guaranteed. The naïve approach to continuous learning exploits these facts; training examples are drawn from a changing source and the system is allowed to make predictions at any time. A similar approach can be taken with batch learning systems, by altering the composition of the batch after each training iteration.

If such an approach to continuous learning is taken, there is an inevitable initial period of development during which prediction quality will be poor. However, as the system matures (over a time frame comparable to the time required for traditional training) the quality of predictions will improve. However, this naïve approach to continuous learning suffers from a significant set of closely related problems:

Training set size is theoretically infinite. Over a sufficiently long time frame the amount of training information will exceed any memory requirements if encoded explicitly; implicit encodings may be prone to overspecification as a result of constant stream of minor additions to knowledge representation. Good generalisation mechanisms are therefore required.

Old knowledge can become false or irrelevant. Over time, knowledge which was assumed to be true may reveal itself to be misleading or false. Systems must be able to remove such false information. In addition, specific knowledge can become irrelevant to a given problem domain over time; irrelevant information wastes storage space and has the potential to impede generalisation.

Old knowledge can remain relevant despite lack of frequency. The absence of supporting examples in a training set does not necessarily correspond directly to irrelevance. Some domains will contain knowledge which cannot be challenged and must be retained at all costs.

These problems encompass what is known as the stability-plasticity dilemma (Heins and Taurisz, 1995): any continuously learning system must remain stable enough to retain old training information but remain sufficiently plastic to allow the incorporation of new information or alter the representation of existing knowledge to reflect changes in the problem domain. In addition, the system should remain plastic to single, significant events but stable in response to single, irrelevant events. In many cases, these goals are contradictory; the distinction between irrelevant and poorly supported information is vague and will be problem specific in many cases.

The naïve approach is therefore impractical for most — if not all — machine learning architectures. More sophisticated approaches, which provide solutions for the stability-plasticity dilemma, are required.

2.7.3 Memory Based Approaches

The simplest approaches to continuous learning are memory based. These approaches memorise all training examples presented to the system and interpolate between these training examples at query time.

One of the most widely used memory based algorithm is K-Nearest Neighbour (KNN) (Stanfill and Waltz, 1986). The 'memory' X of a KNN system is a set of training examples $\langle \vec{x}_i, \vec{y}_i \rangle \in X$ which map an input space $\mathbf{x}$ to an output space $\mathbf{y}$. A query on the KNN system takes the form of a query vector $\vec{x}$. Given this query vector, the KNN algorithm searches X for the those K examples whose $\vec{x}_i$ is nearest to $\vec{x}$ (according to some distance metric, usually Euclidean distance). It then returns the mean of these K nearest neighbours as the output vector. The only significant design decision is the choice of K; this value should be chosen to reflect the number of patterns in memory, and the expected local density of those patterns.

Another common memory based algorithm is Shepard's Method (Shepard, 1968). In calculating output values for the memory system, Shepard's Method considers *all* training examples, weighting each example according to the inverse distance to the query vector $\vec{x}$. The output vector $\vec{s}$ of Shepard's method can be expressed as:

$$\vec{s}(\vec{x}) = \left(\sum_{\langle \vec{x}_i, \vec{y}_i \rangle \in X} \frac{\vec{y}_i}{\|\vec{x} - \vec{x}_i\| + \varepsilon} \right) \cdot \left(\sum_{\langle \vec{x}_i, \vec{y}_i \rangle \in X} \frac{1}{\|\vec{x} - \vec{x}_i\| + \varepsilon} \right)^{-1} \tag{2.25}$$

In this formulation, ε is a small constant used to prevent division by zero. The result of Shepard's model is very similar to that of KNN. However, there is no need to select

an appropriate value for K (the value selected for ϵ is not as critical).

Memory based approaches are essentially a naïve approach, as they make no attempt to control plasticity or stability. All new training examples are remembered, and no old examples are forgotten. Compounding this problem is the problem of memory requirements. If a system is allowed to learn indefinitely, the physical memory required to uniquely store the training examples will eventually become excessive.

Although memory based can be used in a continuous learning framework, the limitations stemming from their naïveté generally restricts their usefulness in real-world applications.

2.7.4 Neural Network Approaches

The foundations of neural network theory were established in the 1950's as attempts to model the activity of the brain in computational structures (Hush and Horne, 1992). The mathematical theory underlying neural networks is covered extensively in the literature (Hush and Horne, 1992) (Hecht-Nielsen, 1990). A wide range of mechanisms exist for training neural networks. However, they all demonstrate fundamentally similar properties.

One of the more useful neural network properties is the ability to generalise. The generalisation property of neural networks is well established. A large number of attempts have been made to improve the generalisation performance of neural networks (Squire, 1996). A neural network generalises through activation of nodes that support a particular output. The decision of class membership is not based upon a single computational evaluation. Instead, a large number of nodes interact and combine to arrive at a classification. A single piece of apparently contradictory evidence about an input does not immediately invalidate a classification, as it is weighted against supporting evidence. If supporting evidence outweighs negative evidence, the network generalises and discards the contradictory evidence. Good generalisation performance allows a very small training set to be used. As a result of generalisation, the network is able to 'fill in the gaps', and interpolate results for unseen data (Lippman, 1987).

This generalisation behaviour is particularly desirable for continuous learning. Any form of explicit encoding of training data will eventually fail due to memory requirements. Using neuron generalisation, neural networks remove the need to explicitly encode patterns, instead using a weight network of predetermined fixed size to represent the problem domain.

The problem of choosing an appropriate network size is the most critical issue in neural network design. Choosing a network which is too large may result in overfitting of the training data and poor generalization to unseen data. A network which is too small may not be able to capture the relations underlying the training data, resulting in poor prediction performance.

This problem is compounded in a continuous learning environment. In a static training environment, a statistical analysis of the training set can assist in the creation of an network of appropriate size. However, in a continuous learning environment, this statistical information is not available — the statistical properties of the problem domain cannot be guaranteed into the future. Choosing a static network topology for such a dynamic training environment is a difficult task.

Neural networks are also subject to the stability-plasticity dilemma, although in the opposite way to memory based systems. Weights in a neural network are constantly updated to reflect changes in the problem domain. Providing that the weight update algorithm allows a mechanism for escaping local minima (a common characteristic), neural networks experience no difficulty in tracking changes in training data. However, they do suffer from forgetfulness. Patterns which are not reinforced are quickly forgotten, as there is no incentive to reinforce weights which are not used.

The problems of determining network topology and knowledge retention can both be solved by allowing learning algorithms to alter the network topology during the training process. Using constructionist mechanisms, it is possible to start with a minimal network, and add neurons and weights as required. Alternatively, one can commence training with an extensive network topology, and prune weights or neurons which do not contribute to generalisation performance. A combination of these techniques is also possible (adding neurons when required, removing others when they become redundant).

One destructionist approach (that is, starting with a full network, and selectively removing weights from the network) is called Optimal Brain Damage (Le Cun *et al.*, 1990). Optimal Brain Damage involves the removal of weights which have the least effect on training error, based upon a diagonal approximation of the Hessian of the network (in some examples, other techniques, such as principle components pruning, is used).

However, Optimal Brain Damage has two significant down falls. Firstly, it is extremely computationally expensive (since the network must be re-trained after each

pruning). Secondly, the pruning process is driven by the training data, and different training data can generate different pruned weights (Squire, 1996).

The Cascade Correlation Neural Network (CCNN) architecture (Fahlmann and Lebiere, 1990), is a good example of a constructionist approach. This architecture possesses some useful properties: it learns quickly, generalises well and is able to learn complex behaviours through a sequence of simple lessons (Fahlman, 1991). The CCNN learning process starts with a minimal network, which grows only when the existing network is shown to be inadequate. This approach has been shown to be effective in a number of architectures. Elman (1993) noted that this form of approach mimics human learning, since children start with an initially restricted memory capacity. In this case, the long learning period plays a positive role in the acquisition and development of behaviour.

Initially, a CCNN consists of input and output units (including an input bias unit, held at a value of 1.0), and connections between these units. This single layer is then trained until error ceases decreasing considerably. If this value of error is sufficiently small, the process stops. Otherwise, a new hidden unit is added to the network.

When a new hidden unit is required, a pool of candidate units are created, each receiving inputs from all existing units (except output units) in the network. The outputs of these candidate units are not connected to the existing network. The input weights of each candidate are trained to maximise correlation between the output of the candidate, and the error of the existing network. When correlation ceases to improve, the best candidate unit is selected and is added to the network. The input weights to this new unit are then frozen, and the weights attached to the output units are retrained (including the weights from the new unit). This process of adding units and retraining continues until error in the entire network falls below a desired level (or a certain number of nodes have been added).

CCNN are well suited to continuous learning problems as their constructive approach adds neurons if the problem domain changes. In addition, the freezing of weights allows the preservation of old concepts in the network. As a result, CCNN (and its recurrent cousin, RCCNN (Fahlman, 1991)) has been used successfully in a range of long term signal processing and process control problems.

Constructive and destructive approaches to topology development both suffer from a related problem. Constructive networks are prone to overdevelopment as the continuous addition of neurons to a network can lead to overspecification and a loss of generalisation

performance. Conversely, destructive networks require an overspecified network at the commencement of training, before being pruned to a network capable of generalisation. Optimal results for continuous problems require a combination of both constructive and destructive approaches.

Another problem common to these architectures is a tendency to be highly sensitivity to parameter selection. CCNN is especially sensitive to parameters; the choice of gain and patience is critical for good performance (Keith-Magee, 1997). As a result, the task of selecting appropriate parameters must be considered carefully.

Optimal Brain Damage and CCNN are not the only mechanisms which have been proposed for constructing and deconstructing network topologies; others include the Upstart algorithm (Frean, 1990), Weight Decay (Krogh and Hertz, 1992) and Weight Elimination (Weigend *et al.*, 1991). Using these constructionist and destructionist algorithms, neural networks have demonstrated themselves well be suited to the representation of continuous learning problems.

2.7.5 Adaptive Resonance Theory (ART)

Although generally classified as a neural network, Adaptive Resonance Theory (ART) is sufficiently eclectic that it warrants separate discussion. The original theory of ART networks was originally proposed by Grossberg (1986), and implemented by Carpenter and Grossberg (1987c).

An ART network performs an unsupervised batch clustering of input data. Given a set of input patterns, an ART network attempts to separate the data into clusters of similar patterns. An ART network consists of two layers of processing elements (nodes), which form an iterative feedback loop. The first layer, F_1, acts as short term memory; the second layer, F_2 is an adaptive layer which acts as long term memory. Each node in the F_2 layer represents a cluster in the set of input patterns and contains the node prototype representing the centre of the cluster.

The number of nodes in F_2 is allowed to change as required, in order to best represent the input patterns. The creation of nodes in F_2 is controlled by a parameter ρ, known as the *vigilance* of the network. This parameter determines the granularity of the clusters represented by F_2 by acting as a threshold on the similarity between clusters represented by nodes.

The *resonance* in ART occurs in the iterative feedback loop between F_1 and F_2 when outputs between the two layers are within some threshold of similarity (that is,

the output vector $F_1 \rightarrow F_2$ is similar to the output vector $F_2 \rightarrow F_1$). The selection of a winning node in an ART network utilises this resonance. Presentation of an input pattern activates the nodes in the F_1 layer, which propagates the input vector through a set of fully connected weights to the F_2 layer. This activates the nodes in the F_2 layer, and the node with the highest inputs initiates a feedback loop with the F_1 layer. This alters the values of nodes in the F_1 layer, which in turn alters the F_2 layer, and so on. This process terminates when a node in the F_2 layer places the network in a resonant state; this node is then labeled as the winning node. If no node can be found to place the network in a resonant state, the network adds a new node to the network, and declares this new node to be the winner.

There are a large number of variations on the basic ART concept. Amongst these are:

ART2 (Carpenter and Grossberg, 1987a,b), which allows for the clustering of analogue input vectors;

ART3 (Carpenter and Grossberg, 1990) which adds a bias to the search process.

FuzzyART (Carpenter and Grossberg, 1991), which incorporates fuzzy operations into the activation functions of the nodes in the network; and

ARTMAP (Carpenter *et al.*, 1991), which uses a pair of ART networks to create a network capable of supervised learning.

Each of these networks are based upon the same basic principles defined by the original ART network. As a result, ART based architectures, regardless of the specific implementation, tend to suffer from some common problems. As a result of the Winner-Take-All competitive nature of ART networks, they are prone to local minima in the search space (Baraldi and Blonda, 1998). They are also prone to overfitting, especially in the presence of noisy data . They are also potentially sensitive to the order of presentation of input vectors (Shih *et al.*, 1992).

Just as with neural networks, ART networks can be easily applied to continuous learning problems. The naïve approach of continuously presenting new training data will result in a network which continuously adds new information to the network. However, just as with standard neural networks, ART networks are prone to plasticity problems. Over time, the network will tend to drift away from concept representations which are not reinforced through training.

This plasticity issue has been addressed in ART variants such as Fuzzy ARTMAP (Carpenter *et al.*, 1992), FOSART (Fully Organising Simplified ART) (Baraldi and Parmiggiani, 1997) and IFOSART (Incremental FOSART) (Loh *et al.*, 2001). IFOSART, for example, maintains the learning rate of the network as a whole above a given threshold, while allowing the learning rate of individual neurons decay to a stable minimum. The decay in individual neuron learning rate prevents the loss of old patterns. However, by maintaining an overall network learning rate, new nodes can be added to the system as required.

ART architectures have proven themselves to be good unsupervised (or in the case of ARTMAP, supervised) classifiers for general learning problems (as is evidenced by the extensive range of literature and variants on the basic ART architecture). They are also well suited to continuous problems, and have been successfully used for this purpose in a number of situations (Loh *et al.*, 2001).

2.7.6 Conceptual Clustering

Another prominent machine learning technique is the use of conceptual clustering algorithms. The most common examples of these techniques are decision tree algorithms, such as ID3 (Quinlan, 1982)(Quinlan, 1986) and C4.5 (Quinlan, 1993). When presented with a static data set, algorithms such as these are able to rapidly develop an efficient representation of the training data. Each training pattern is grouped into a conceptual cluster based upon some measure of conceptual similarity.

However, decision tree algorithms such as these cannot be used for continuous learning problems. Static conceptual clustering algorithms cannot modify their conceptual descriptions. If training examples begin to contradict the conceptual clusters developed during initial training, the problem representation must be completely rebuilt to accommodate the new facts. Additionally, if recently presented data suggests a better primary partitioning of the problem space, static clustering algorithms are unable to generate a new cluster hierarchy without complete retraining.

Compounding this problem is the fact these many of these static algorithms contain only generative functions — they do not contain pruning functions. As new data is presented, these algorithms maintain the ability to add new classification rules to their concept descriptors. However, these classification rules, once added, are never removed. This potential for continuous rule creation introduces a risk of overfitting, which would result in a loss of generality. Consequently, naïve approaches to continuous conceptual

clustering are not feasible.

The most simple continuous conceptual clustering algorithms bear some resemblance to the memory based systems discussed in Section 2.7.3. In these simple systems, all training examples are stored and a partitioning algorithm is used to perform conceptual clustering. One example of such an algorithm is COBWEB (Fisher, 1990; Reich, 1991), which performs a hill-climbing search through a space of hierarchical classification schemes using operators that enable bidirectional travel through this space.

The COBWEB algorithm builds a conceptual hierarchy, with general nodes at the top of the hierarchy and more specific nodes at the bottom. Each leaf node on this hierarchy contains every one of the specific instances used to create that leaf. COBWEB never deletes instances, and therefore never forgets any training pattern to which it has been exposed. Each node in the hierarchy is a probabilistic concept representing a cluster in the data set. COBWEB is able to merge nodes, split nodes, and create new nodes. By using these operators on the node space, it is possible for the COBWEB algorithm to reconfigure the representation of the problem over the lifetime of the system.

One limitation of COBWEB is that can only represent symbolic data. A variant of COBWEB, known as CLASSIT (Gennari *et al.*, 1989), is able to represent symbolic and real valued attributes. However, even with this variant, COBWEB is fundamentally unsuitable for large scale continuous learning problems. Just as with memory-based approaches, the requirement that the algorithm remember every training example imposes an impractical memory requirements on a long term learning system.

UNIMEM (Lebowitz, 1989, 1986) overcomes this memory requirement by implementing a simple forgetting mechanism. UNIMEM represents the problem space in a similar manner to COBWEB: as a hierarchy of nodes representing concepts. Each time a new training example is presented, and a feature is matched, the value of that feature is strengthened; each time a feature is not matched, the value of that feature is weakened. If any feature value ever falls below a user specified threshold, it is pruned from the concept hierarchy, and is forgotten. As a result of this pruning, UNIMEM, unlike COBWEB, is not a full memory system. UNIMEM will forget training patterns.

However, UNIMEM is not without problems. Firstly, it requires the user to set system parameters to make clustering decisions. Inappropriate parameter values could completely restrict pruning (resulting in no forgetting) or could result in excessive pruning (resulting in an amnesic system). Secondly, pruning is performed based upon

usage. Old knowledge, which has not diminished in truth value, will be pruned from the concept hierarchy if not reinforced. This is not desirable behaviour if significant, but infrequently reinforced training information is to be retained.

A limitation of both COBWEB and UNIMEM is that they make the assumption that the concepts in question have some concept of central tendency and that it is only the boundaries between these central concepts which vary over time (Widmer and Kubat, 1992). The STAGGER algorithm (Schlimmer and Granger, 1986b)(Schlimmer and Granger, 1986a) counteracts this notion. In STAGGER, each conceptual representation is accompanied by two weighting measures, which describe the logical sufficiency and logical necessity of the item for the general description of the concept. These two measures are then used to guide the search for new descriptions; weak examples are removed from cluster descriptions, and strong examples are used to form new clusters. As a result, STAGGER is able to cope with substantial changes in concept meaning over time.

The FLORA algorithm (Kubat, 1989) (Kubat, 1991) (Kubat, 1993) uses a windowing approach to overcome memory requirements. This algorithm maintains a set of positive, negative and candidate descriptors. During training, the presentation of a new positive training example results in either a new descriptor being added to the set of positive examples or in a the transferal of a candidate descriptor to or from the positive descriptor sets. A similar process is performed after the presentation of a negative example. The training process also contains a history window. As training examples fall out of this window, they are removed from the descriptor sets. In this way, the memory requirements of the FLORA algorithm are constrained, the algorithm stores as many instances as will fit in the window.

However, this approach, as with UNIMEM, requires the forgetting of old training data. Although the truth value of old training examples may not diminish over time, FLORA will reject old training examples from its knowledge representation. This represents an unacceptable loss for a continuous learning system.

One algorithm which removes the need for training pattern memory is Incremental Learning with Forgetting (ILF) (Lazarescu *et al.*, 1999). The ILF algorithm is a hierarchical clustering tool, derived from COBWEB and UNIMEM. However, ILF does not store complete training instances. Instead, ILF stores specific values from training instances when those values represent a novel aspect of a given feature. This prevents fragmentation in concept representation as no excess information is stored in feature

nodes, making ILF well suited to the learning of problem domains with large numbers of attributes and training instances.

In addition, ILF implements a localised forgetting mechanism. The forgetting mechanism of UNIMEM affects the entire representation in that the strengthening of one feature requires the weakening of all other concepts. Consequently, all forgetting actions have global impact. The ILF algorithm implements a series of local aging procedures to simulate short, medium and long term memory. These aging procedures allow local plasticity to account for drift in the problem domain but ensure the stability of fundamental hierarchy concepts. As a result of these characteristics, ILF represents a robust, efficient method of performing incremental conceptual clustering.

2.8 Kohonen's SOM in Continuous Learning

In the previous section, a number of approaches to continuous learning were presented. These approaches range from naïve training of traditional learning algorithms to sophisticated approaches of network growth, pruning and coalescence to overcome the stability-plasticity dilemma.

However, these systems are all fundamentally similar. Although both supervised and unsupervised variants exist, all the examples presented ultimately yield a discrete classification mechanism. While this is a useful representation for a wide range of problems, it ignores those problems for which a continuous representation would be more appropriate. In addition, discrete classification mechanisms are not well suited to problems which require the visualisation of a complex data space.

Kohonen's SOM algorithm has been shown to be well suited to the representation of static problems which exhibit these characteristics. The 2D output representation of a Kohonen SOM avoids discrete classification, representing output as a topological relationship. This relationship is a highly simplified representation of a high dimensionality data space, which allows visualisation. Given that Kohonen's SOM algorithm has demonstrated itself to be a flexible and extensible algorithm, which can be adapted to a wide range of applications, it seems reasonable to attempt to apply Kohonen's algorithm (or a variant of it) to the continuous learning problem.

The naïve approach to a continuous learning SOM will inevitably fail. Kohonen parameter decay schemes universally decay neighbourhood size and gain to 0 (or a very small asymptotic value) over some predetermined training period. At the completion of this training period, the map becomes a static weight representation and, consequently,

no further adaptation is possible. A simple method to overcome this limitation would be to decay neighbourhood size and/or learning gain to nonzero values. However, this approach has not been tested in the literature.

Kohonen (2001) describes no attempts to achieve continuous learning using Kohonen-style self-organisation. However, some of the techniques presented for use with static data sets have the potential to be adapted for continuous problems.

2.8.1 Growing Maps

A recurrent theme in continuous learning algorithms is the idea that representations must be able to develop over time. Successful continuous learning algorithms for neural networks, ART networks, and conceptual clustering involve mechanisms for adding descriptions or nodes to their knowledge representations as the problem domain evolves.

A similar line of research has been explored in SOM research. The idea of growing a SOM extends from the plastic concept of neighbourhood described in Section 2.5.3. Although traditional SOMs have a static array of neurons in a predetermined topology, there is no requirement that this array remain static. New neurons can be added to the candidate pool at any time; if the weights of a new neuron result in the selection of that neuron as a winner, it will be reinforced, and will eventually become an integral part of the SOM topology.

This leaves two problems for the growth algorithm:

1. When is a new neuron added to the network?

2. Where is it added in the topology, and what neighbourhood relationships does it possess?

The constraints placed upon the neighbourhood relationships of new nodes determine the topological structure that evolves.

If there are no restrictions imposed on neighbourhood relationships in the SOM, the resultant SOM structure is a Neural Gas (Martinetz and Schulten, 1991)(Martinetz, 1993): that is, a graph of interconnected neurons without any predetermined structure. The graph resulting from this learning process is a subgraph of the Delaunay triangulation corresponding to the reference vectors of the map (Fritzke, 1996).

However, if restrictions are imposed, a formal underlying structure will evolve. The Growing Grid (GG) method (Fritzke, 1992) (Fritzke, 1995) enforces a hyperrectangular structure on the node graph. Each addition to the map takes the form of a complete

hyperrow or hypercolumn in the map. A similar approach was taken by Blackmore and Miikkulainen (1995), to develop the Incremental Grid Growing Neural Network (IGG). This network restricts the growth of the network to a rectangular plane.

More complex restrictions are also possible. Fritzke (1994) defines a k-dimensional simplex known as a hypertetrahedron. In this structure, each of the (k+1) nodes of the simplex are connected by $(k+1)k/2$ neighbourhood relationships. These simplicies are joined together to form a complex hypertetrahedral map relationship known as a Growing Cell Structure (GCS). This allows the representation of complex topologies within a regular cell structure.

The methods for adding and removing neighbourhood relationships are almost as varied as the topological structures they generate. For example, the Neural Gas (Martinetz and Schulten, 1991) modifies the topology locally by inserting and deleting neighbourhood relationships depending on their 'age'. A neighbourhood relationship between a pair of neurons is established (or reinforced) whenever both neurons are judged to be similar to a training pattern. However, the relationship is removed if both neurons are not selected again within a given time frame. IGG (Blackmore and Miikkulainen, 1995) attempts to minimize errors in representation, adding and removing edges as required to minimize this error. GCS (Fritzke, 1994) attempts to generate a uniform probability distribution between neurons; neurons are added and removed to maintain this uniform probability distribution.

Regardless of the method used to add neurons and neighbourhood relationships, constructive approaches suffer from a significant problem. One of the principal uses of Kohonen's algorithm is as a visualisation tool; using a SOM, high dimensionality data sets are reduced to a 2D representation, allowing the user to examine complex topological representations in a form which is easily interpreted. If the SOM surface ceases to be a simple rectangular map, this visualisation becomes increasingly difficult. The dynamic introduction and removal of neighbourhood relationships further increases the complexity of the visualisation task. Given that the goal of visualisation is to simplify the representation of a problem, techniques which explicitly add complexity to this representation are of limited use.

Constructive SOM methods are also prone to generating specific, rather than generalised maps. The process of generalisation requires the acceptance of error and localised distortions. If constructive mechanisms attempt to minimize these errors and distortions, the resulting SOM will specifically encode training patterns, rather than

generalise to broader concepts. This process can be controlled using parameters to define acceptable levels of distortion. However, this then requires a method for selecting appropriate parameters.

Computational complexity is also a significant issue with constructive SOM approaches. When a static SOM structure is used, the computational complexity is constrained and the number of computations per training epoch is known *a-priori*. However, if a constructive technique is used, the number of neurons could potentially grow indefinitely, introducing a significant computational load per training epoch. In addition, the process of creating and deleting new SOM neurons is a computationally intensive process. Again, these factors could be controlled by a parameter describing maximum network size or constraining the neuron creation process, but this would also required a method for selecting appropriate parameters.

Constructive SOM models of this type have not been applied to problems with continuous domain. Although many papers refer to the incremental nature of constructive learning, they are referring to the incremental addition of nodes to the network, not the extension of the problem domain itself. The use of constructive SOM models for continuous problems is one potential area for further research.

2.8.2 Neighbourhood Pruning

The analogy with continuous neural networks can also be drawn with network pruning. Consider a map with neurons which initially have a generous neighbourhood set. This map can be trained, establishing an initial topology. After this initial organisation, the neighbourhood set can be pruned. If a given neuron has a neighbourhood relationship which conflicts with the topology of the data set, this neighbourhood relationship can be removed from the neighbourhood set of the neuron. Several schemes for removing neighbourhood relationships in this fashion have been proposed in the literature (Kangas *et al.*, 1989) (Blackmore and Miikkulainen, 1993) (Szepesvári and Lőincz, 1993). In this way, the definition of neighbourhood changes over time, but not in a pattern of simple geometric decay. The resulting SOM will have neighbourhood relationships which mirror the topological relationships found in the data set.

The dynamic neighbourhood technique allows for the development of complex map topologies without the need for *a priori* knowledge regarding data set topology. However, as with growing map structure, the use of pruning techniques impedes the ability of the map to be used as a visualisation tool. It is also worth considering that if the

neighbourhood pruning process is extensive, the resulting map will form a number of isolated islands. In this way, the SOM algorithm effectively reduces to an unsupervised clustering tool. While unsupervised clustering is a useful task, there are many other algorithms which are much more efficient at performing this task than a highly customised SOM algorithm.

As with constructive SOM models, destructive and pruning models have not been applied to continuous problem domains. This suggests another avenue for further SOM research.

2.8.3 SOMs with a Conscience

One feature of Kohonen's SOM learning algorithm is that the output map is directly related to the probability distribution of the training set. As a result, there is no guarantee that each neuron on the map will be selected as a winner with equal probability. Certain neurons near the center of key concepts will be often selected as winners. Conversely, it is entirely plausible (and common) for a specific neuron on map to never be selected as a winner.

This can be a useful feature, as it gives increased resolution for those concepts in the training data which are heavily represented. However, this occurs at the cost of resolution in other areas which, although less frequent in the training set, may be just as significant to the learning task.

To overcome this limitation, DeSieno (1998) introduced the 'SOM with a Conscience'. In a SOM with a Conscience, each neuron is trained as normal. However, every time a neuron is selected as a winner, this win is recorded. The win count is then used to develop a bias value B: thus, values with a high win count develop a large bias. This bias value is then added to the function used to evaluate the winning neuron in the map. In this way, any single neuron is prevented from dominating the winner list. If a neuron wins repeatedly, the bias value for that neuron will increase until it is impossible for that neuron to be selected over others. The term 'conscience' derives from the fact that any given neuron will begin to 'feel guilty' if it wins too often, and will prevent itself from being selected in the future.

The application of the conscience scheme to SOM learning generates SOM representation in which each neuron wins with approximately equal probability. Neurons which win too often are penalised and are not selected, allowing underrepresented neurons to be selected as winners. In this way, the conscience parameter achieves a similar goal to

Neural Gas methods (Martinetz and Schulten, 1991). However, equiprobability in the output space is not achieved at the cost of a regular map topology as the SOM with a conscience can be represented on a simple rectangular grid.

By itself, it is not obvious how the introduction of a conscience parameter can be used to assist continuous learning in a SOM. However, the SOM with a conscience does demonstrate that by controlling the weight update process, it is possible to prevent the expansion of one concept on the map, while encouraging the expansion of other concepts. This feature could be exploited to develop long term memory on a SOM. Although conscience itself may not be of use for continuous learning, there is significant scope for investigation in the manipulation of map selection and update procedures to achieve specific learning goals.

2.9 Conclusion

In this chapter, the current state of SOM and continuous learning research has been presented. Three key SOM algorithms were presented, along with a range of variations on this base algorithm. A number of metrics were presented for establishing the quality of SOM representations. Finally, the continuous learning problem was defined, along with a range of naïve and informed solutions for these problems.

The literature explored in this chapter has shown that there are many variations on Kohonen's SOM algorithm which have been applied to static problems. Furthermore, there have been many attempts to perform continuous learning using other machine learning architectures. However, Kohonen's SOM algorithm has not been applied to continuous learning problems. In the following chapters, a model of learning and development will be developed, and applied to Kohonen's SOM algorithm. This model will add continuous learning problems to the range of problems to which a Kohonen-style SOM architecture can be applied.

Chapter 3

A Theory of Lifelong Learning

> The scientists at the Institute thus discovered the driving force behind all change, development and innovation in life, which was this: herring sandwiches. They published a paper to this effect, which was widely criticised as being extremely stupid.
> — *Mostly Harmless* (Adams, 1992)

3.1 Introduction

Although a large number of machine learning algorithms have been presented in the literature, the vast majority of these algorithms treat the task of machine learning as a once-off problem of error maximisation or entropy minimisation with respect to a single static data set. While these machine learning algorithms can generate useful solutions in specific problem domains, they have met limited success as general problem solvers. The success of these algorithms tends to rely upon application specific heuristics or random elements in the convergence method.

One of the reasons that these algorithms fail as general problem solvers is that they fail to take into account that nature of real-world data. Real-world problems consist of data which can change over the lifetime of the system. Biological learning systems have developed methods of learning during the entire lifetime of the organism, rather than restricting learning to a short training period prior to use.

The purpose of this chapter is twofold. Firstly, in Section 3.2, it is argued that a biological approach to learning and development could aid in the development of improved representations of a data space. In Section 3.3, a biological model of learning is proposed. This model describes the life-cycle of an organism, and describes the benefits of a lifelong approach to learning and development. Further analysis of this model requires a machine learning architecture which has a biological analogue;

maximal Kohonen SOMs are proposed as an appropriate architecture.

Secondly, a method for quantifying SOM performance is presented. This metric, called the $\mathcal{Q}$ metric, does not attempt to examine the weight space of the SOM or exploit aspects of the training algorithm. Rather, it quantifies the observable output of the SOM by comparing the distances between vectors in the training set with the distance between the SOM neurons selected as winners by those vectors. This metric is demonstrated using the Simple Grid training set.

3.2 A Biological Metaphor

There are three key differences between traditional machine learning algorithms and the learning process observed in biological systems:

Learning is a continuous process: Learning is not a once off process of minimising error. A key aspect of learning in biological systems is the way in which new knowledge is continuously assimilated into the existing knowledge base.

Simple tasks are learned first: Biological systems do not learn from a canonical training set consisting of a complete set of complex training examples. Rather, initial training occurs using simplified or heavily filtered data. For example, a child learns to do simple mathematics before being exposed to calculus. The act of learning simple addition establishes base knowledge upon which more complex knowledge can be built. Traditional machine learning techniques would teach calculus by attempting to minimise representational error while training with a spanning set of calculus problems.

Hard lessons are learned well: Using traditional learning techniques, strong reinforcement of an idea requires that the frequency of the relevant training example in the data set be proportional to its importance. In biological systems, highly significant training examples do not require repeated presentation. Life threatening or life changing experiences may result in a small amount of training data, but the lessons learned from this data are not forgotten easily. For example, a child will only touch a hot saucepan once; the lesson linking heat and pain is not lost due to a lack of reinforcement.

These three differences are largely the product of the way in which data is presented to traditional learning systems. The training algorithms for most machine learning

systems divide the lifetime of an algorithm into two parts: a learning phase and a use phase. The algorithm is exposed to training data during the learning phase. At the completion of learning, the system is frozen and is used to perform classification.

However, this duality is not reflected in the real world. Biological learning systems are constantly exposed to new training data and remain plastic to this new training data throughout the lifetime of the organism (although the extent of this plasticity does decrease over time). Biological learning systems make no distinction between the process of learning and the process of use.

Traditional learning algorithms require the separation of the learning process to provide a stable goal state upon which the algorithm may converge. However, the dynamic nature of real-world learning problems is not compatible with this method of separation. Real-world data requires a learning algorithm which can adapt to changes in the data space over the lifetime of the system. In order to encapsulate the desirable features of a biological learning system, a new model for learning and development is required — a model which incorporates the entire life-cycle of an organism.

3.3 Life-Cycle of an Organism

Figure 3.1 describes the learning and development life-cycle of a typical organism. The organism begins as a genotype, which is expressed through a set of expansion rules as a juvenile phenotype. During the juvenile phase of life, this phenotype interacts with a minimal environment, developing characteristics which are affected by, but are not a direct function of the genotypic description of the organism. Once the juvenile reaches a certain level of maturity, the organism is released into the real world where it interacts with the environment and adapts further to its surroundings.

After this juvenile phase, the organism becomes able to reproduce. Under a Darwinian model, only the fittest organisms reproduce and through the copying and mutation of the genotype of fit parents, a new generation of organisms is born.

In addition to the genetic heritage obtained directly from parents, a young organism gains much useful training data through interaction with, and supervision by, the adults of a population. By presenting useful lessons and training examples to a young individual during the juvenile phase, it is possible to encourage the organism to develop sophisticated behaviours which the experienced parent has found useful.

The biological learning processes observed today are the result of millions of iterations of this cycle, as simple learning structures developed into more complex structures,

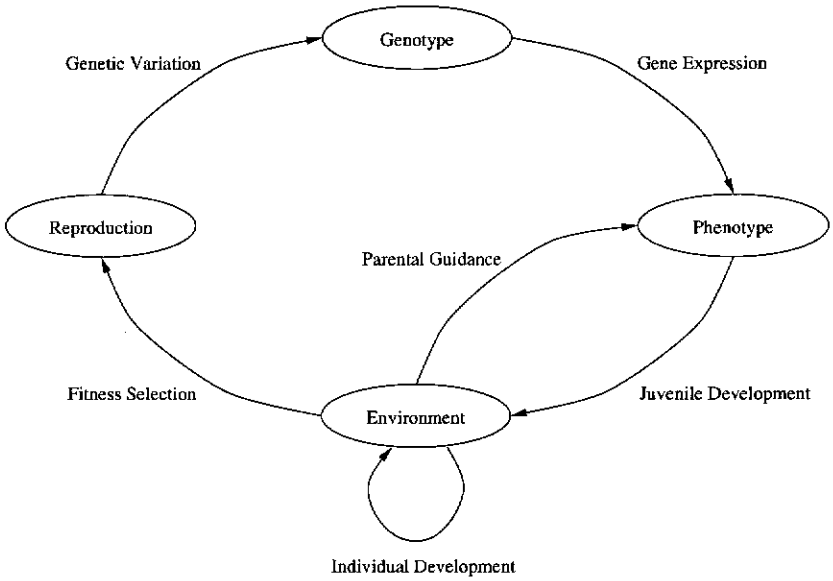


Figure 3.1: Learning and development life-cycle of a typical organism, adapted from Vaario and Ohsuga (1992)

yielding the sophisticated reasoning system that are observed in biological systems.

This thesis will concentrate on just one part of this life-cycle; the stages of development of a single individual, including the training information gained from interaction with an experienced 'parent'. The problem of species wide development over a generational time frame is left as a separate problem for future work.

3.3.1 Stages of Learning

An individual organism goes through four keys stages during its lifetime:

Infant: During infancy, the cognitive process of an organism are extremely impressionable, as the presence or absence of key training examples can drastically affect the later development of the individual. While at this stage of development, the organism is incapable of fine resolution of ideas.

Juvenile: After establishing the fundamental cognitive processes, the individual progresses into a juvenile period. During this stage, training examples still consist of broad concepts. However, there is significant improvement in the resolution of these concepts.

Adolescent: During adolescence, directed learning on specific topics can take place. The training examples presented as broad concepts during the infant and juvenile

phases of development are increased in complexity to produce sophisticated representations of domain knowledge. The resolving power of the system improves to reflect these changes.

Adult: Finally, the organism experiences a period of adulthood. This period of adulthood is prolonged, and the space of input values can change during this time. Additional knowledge acquisition is limited to the fine tuning of well established cognitive patterns. During this period of the life-cycle, fundamental reorganisation of knowledge is neither useful nor desired; at this point, a fundamental reorganisation would result in the loss of all training acquired during the previous stages of development.

The parents of the individual play an important role through the first three of these stages. The process of incremental training relies upon the presentation of a syllabus of useful training examples which increase in complexity as the individual develops. In nature, the syllabus would be the result of many generations of refinement. However, in an artificial system, it may be possible to use domain knowledge to establish the core principles.

In this thesis, it is proposed that by utilising this process of lifelong development, it is possible to develop a better representation of a specific data domain, in a shorter or equivalent period of time than would be required by traditional learning architectures. Furthermore, it is proposed that by controlling plasticity in concept representation, it is possible for a system to adapt to changes in the composition of the data domain.

3.3.2 Modeling Individual Development

This lifelong learning model is based upon a metaphor which is tied to the behaviour observed in biological systems. The use of a learning algorithm which is also biologically derived will simplify the application of this biological metaphor. It is for this reason that Kohonen-style Self-Organising Maps will be used to demonstrate the principles and benefits of a lifelong process of learning and development.

Kohonen-style SOMs derive from an attempt to mathematically model low-level brain functionality (although there are some notable differences with biology, discussed in Section 2.4). The Kohonen algorithm derives from the initial attempts to explain the organisation observed in the cortex in a simple algorithmic manner, rather than by a process of dynamic feedback. Although Kohonen's SOM algorithm is not a literal model of biological behaviour, it does maintain an underlying biological metaphor.

Specifically, the maximal SOM algorithm presented in Section 2.5.1 will be used in this thesis. Despite the fact that Kohonen's minimal SOM algorithm is more flexible, and generally learns faster than maximal SOM algorithms, it destroys a key biological metaphor — that of weight significance — making the application of further biological metaphors a difficult task. Since the goal of this thesis is to explore additional biological metaphors, it is reasonable to begin with the most biologically plausible algorithm which is computationally feasible.

This choice of maximal SOMs does introduce a requirement for careful consideration in the choice of data set. As was discussed in Section 2.5.1 many data sets are not appropriate for use with a maximal map. However, these data sets can be posed in a manner which is compatible with maximal SOMs, at a slight increase in computational complexity. The data sets presented in Appendix A have been constructed in such a manner.

3.4 Measuring Success

The approach taken in this thesis is largely empirical in nature, due to the impractical nature of a theoretical approach. There are many examples of theoretical proofs on self-organising processes (Amari, 1980) (Tanaka, 1990) (Amari, 1990) (Kaski and Kohonen, 1994), including proofs on Kohonen's SOM algorithm (Cottrell *et al.*, 1998) (Flanagan, 1998) (Luttrell, 1994). These proofs cover many aspects of self-organisation, including conditions for parameter selection and decay. However, these proofs either rely upon extensively simplified (and therefore limited) versions of SOM algorithms, or draw only vague conclusions on the operation of complete SOM algorithms. The complexity of interactions in a self-organising process, coupled with the iterative nature of the process, inhibits the development of strong theoretical models of self-organisation.

Given that a theoretical analysis of self-organisation is infeasible, an empirical approach must be taken. However, to perform an empirical study, a mechanism for objectively evaluating success must exist.

Section 2.6 introduced a number of metrics exist for the quantification of topology preservation in self-organising maps. However, these metrics tend to be either computationally intensive, inappropriate for maximal SOMs, or specific to a single type of data set or combinations.

In order to solve these problems, a new metric is proposed. The proposed metric ignores the Kohonen algorithm and the underlying weight space, instead observing the

end product (the visual topology) and evaluating the quality of this end product.

3.4.1 Topological Preservation

The end goal of a SOM is to represent the relationships which exist in a $\Re^n$ data space on a $\Re^2$ map space. This process is successful if the topological relationships in the data space are preserved in the map space.

Given that the relationships to be preserved are geometric in nature, it seems logical to measure them in a geometric fashion. In formal terms, given two vectors $\vec{x}_i, \vec{x}_j$, their projections into map space $\vec{x}_i^{\perp}, \vec{x}_j^{\perp}$, and the distance function $d(\vec{A}, \vec{B}) = |\vec{A} - \vec{B}|$, the relationship:

$$d(\vec{x}_i, \vec{x}_j) \propto d(\vec{x}_i^{\perp}, \vec{x}_j^{\perp}) \tag{3.1}$$

$$i.e. \quad d_{ij} \propto d_{ij}^{\perp} \quad \forall i, j \tag{3.2}$$

will hold if the SOM projection is a good topological representation of the data set. If we consider $\vec{x}_i$ and $\vec{x}_j$ to be drawn from a population $\mathbf{X}$ consisting of N vectors, there exists a population of N^2 possible distances $\mathbf{d}$ constructed from all possible pairs of vectors i, j drawn from $\mathbf{X}$. This population $\mathbf{d}$ will have values in the range $[0 : max(\mathbf{d})]$, and could follow any probability distribution over this range. A similar set $\mathbf{d}^{\perp}$ exists for distance between the projections of the data vectors in map space.

To simplify analysis, $\mathbf{d}$ can be normalised with respect to their maximum possible values to yield a set of values in the range $[0 : 1]$. This is done by searching the set $\mathbf{d}$ for a maximum value, and dividing all other distances by this constant. In addition, a process of histogram equalisation is applied, yielding a data set with uniform distribution over the normalised range.

The distances $\mathbf{d}^{\perp}$ also require normalisation. However, the maximum possible distance for these distances is known (the length of the map diagonal), so the scaling process is much easier. No histogram equalisation is performed on these distances, as they are the dependent quantity in this relationship.

The resultant of this normalisation and histogram equalisation process is two new set of distances, $\mathcal{D}$ and $\mathcal{D}^{\perp}$, both on the range $[0 : 1]$. This distances sets yield a much simpler relationship, without a constant of proportionality:

$$\mathcal{D}(\vec{x}_i, \vec{x}_j) = \mathcal{D}(\vec{x}_i^{\perp}, \vec{x}_j^{\perp}) \tag{3.3}$$

$$i.e. \quad \mathcal{D}_{ij} = \mathcal{D}_{ij}^{\perp} \quad \forall i, j \tag{3.4}$$

The extent to which this relationship does not hold is indicative of the level of failure of the self-organising process.

Minor deviations from this relationship are to be expected — after all, SOMs are not intended to be a precise quantitative mapping. However, major deviations from this relationship should be examined. There are two types of deviation which can occur:

$$\mathcal{D}_{ij} \ll \mathcal{D}_{ij}^{\perp} \tag{3.5}$$

$$\mathcal{D}_{ij} \gg \mathcal{D}_{ij}^{\perp} \tag{3.6}$$

Equation 3.5 relates to the phenomenon of *tearing* on the map; a pair of points may fall close together in the data space, but are placed apart from each other on the map. This is an indication that a continuity that exists in the data set is not observed in the mapping of the points. Equation 3.6 relates to the phenomenon of *spread*; a pair of points may be distant in the data set, but fall close together on the map. The presence of either of these qualities in a trained SOM suggest that the map may not be a good representation of the complete data set.

It is worth noting that some deviation, while not desirable, is often inevitable. The mapping of an n-dimensional space[1] onto a 2-dimensional space must result in a loss of some topological information; either the complete loss of $n - 2$ dimensions, or the partial loss of information from all n dimensions.

This inevitable loss in dimensionality also indicates that tearing is a more significant problem than spread in a SOM topology. Tearing represents the introduction of a new topological relationship on the map. Given the scarcity of dimensions in a SOM output representation, this characteristic should be strongly avoided. Spread represents an inevitable consequence of a loss of dimensionality; if dimensions in the data set are lost, points will inevitably be distributed across the map, despite having a small Euclidean distance between them. While not desirable, this characteristic should not be considered as critical a failure as tearing.

[1] In this context, it is only the dimensionality of the principal components that is significant. A data set representing a plane in a 3D space has an underlying dimensionality of 2, and so no information will be lost in the mapping of the 3D data to a 2D surface.

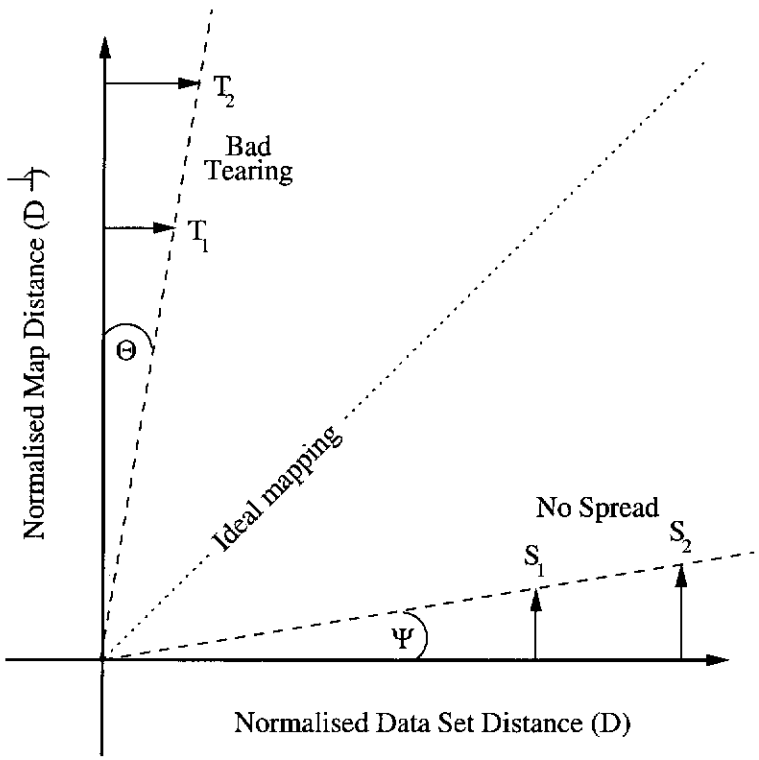


Figure 3.2: An ideal topology preservation plot. Points in the shaded region are indicators of a poor topological mapping.

3.4.2 A New Metric

This loss of dimensionality can be visualised by plotting the distances $\mathcal{D}$ against the distances $\mathcal{D}^{\perp}$. Figure 3.2 gives the idealised result of such a plot. In the ideal case, every point in this plot would fall along the line $\mathcal{D}^{\perp} = \mathcal{D}$. Deviations from this line indicate faults in the topology, with points falling in the shaded regions indicating the worst possible faults.

More generally, any points falling on the line $\mathcal{D}^{\perp} = \kappa\mathcal{D}$ (for some constant κ) will share identical tearing and spread characteristics. If $\kappa = 1$, this corresponds to the ideal case, with no tearing or spread. Other values of κ indicate data which demonstrate excessive tearing or no spread. If $\kappa \gg 1$ (for example, the points T_1 and T_2), the data exhibits bad tearing; a value of $\kappa \ll 1$ (for example, the points S_1 and S_2) indicates poor spread.

This value κ can therefore be used as a quantifiable measure of the extent of tearing or spread of a specific data pair:

$$\kappa = \frac{\mathcal{D}_{ij}}{\mathcal{D}_{ij}^{\perp}} = \tan\Theta_{ij} \tag{3.7}$$

$$\frac{1}{\kappa} = \frac{\mathcal{D}_{ij}^{\perp}}{\mathcal{D}_{ij}} = \tan \Psi_{ij} \tag{3.8}$$

The tangent values resulting from these expressions will fall in the range $[0 : \infty]$, with 0 corresponding to the worst case (bad tearing or no spread respectively). To simplify the analysis of a large set of data, a Gaussian can be used to scaled this infinite range to a range of $[0 : 1]$:

$$\mathcal{T}_{i,j} = \exp\left(\frac{-(\tan \Theta_{ij})^2}{2(0.3)^2}\right) \tag{3.9}$$

$$\mathcal{S}_{i,j} = \exp\left(\frac{-(\tan \Psi_{ij})^2}{2(0.1)^2}\right) \tag{3.10}$$

Equation 3.9 can then be used to quantify the extent of tearing in the map representation and Equation 3.10 can be used to quantify the extent of spread in a map representation. As Θ and Ψ increase, the values of $\mathcal{T}$ and $\mathcal{S}$ will decrease. The rate at which this decrease occurs is determined by the constant denominator term; the values of 0.3 and 0.1 have been selected to ensure near 0 values of $\mathcal{T}$ and $\mathcal{S}$ when Θ and Ψ are $\approx 45°$. These constants also include an acknowledgement that tearing of a map is a worse characteristic in a topology than poor spread.

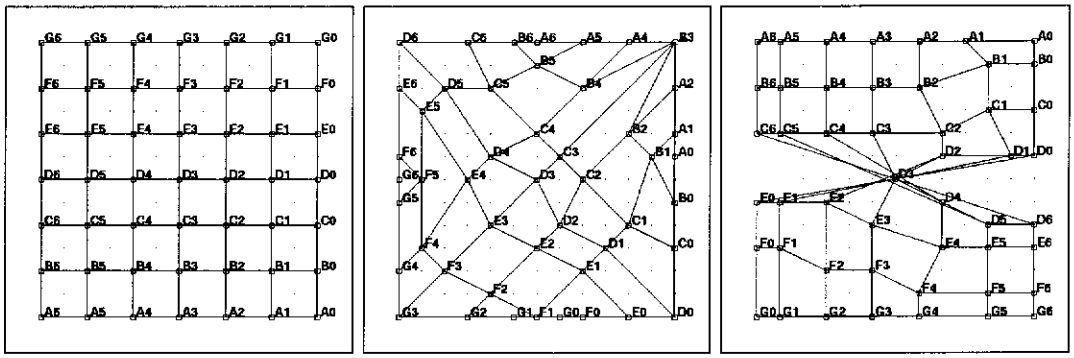
Both of these measures return ≈ 0 in the ideal case, and larger values in the presence of a bad topology. Consequently, an overall quality metric $\mathcal{Q}$ for the dataset $\mathbf{X}$ can be defined as:

$$\mathcal{Q} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{T}_{i,j} \mathcal{S}_{i,j}}{N^2} \tag{3.11}$$

This $\mathcal{Q}$ metric will yield values near zero in the ideal case, and increases if spread or tearing becomes a problem in the mapping of the data set. This metric provides a single value which can be used to describe the quality of a topology provided by a SOM, in a manner independent of the SOM algorithm.

Consider the trained maps in Figure 3.3. The algorithm used to train the map is irrelevant — only the output state and its relationship to the data set is significant. The data set used to train these maps is a 7×7 grid (See Section A.3 for details). The SOM is a map of 15×15 neurons. Technically, this is a 49 dimensional input set; however, the underlying topology is 2 dimensional. Thus, it should be possible to completely preserve the underlying topology in a trained SOM.

At the end of training, a test set of 100 points was generated (as described in Section A.3); the distances between all possible pairs of points taken from this set were

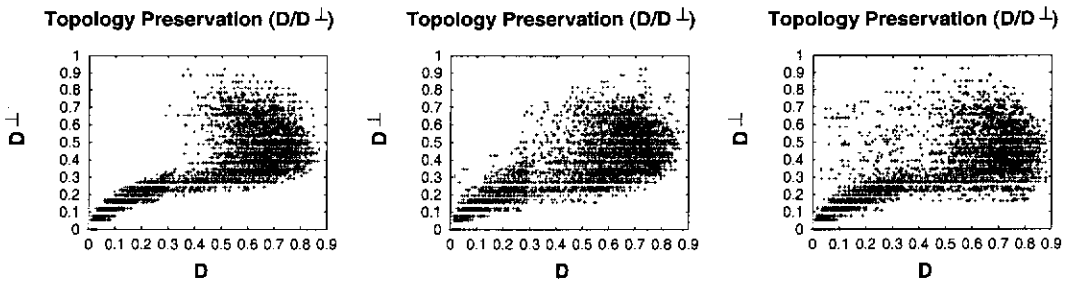


(a) Well trained map (b) Adequate map (c) Badly trained map

Figure 3.3: Three trained maps. (a) A well trained map — topology is well preserved; (b) An adequately trained map — topology is well preserved, but orientation is not optimal; and (c) A badly trained map, incorporating a twist — a loss of topological information.

generated, normalised, histogram equalised, and plotted. Figure 3.4 is a plot of the $\mathcal{D}/\mathcal{D}^{\perp}$ relationship. This plot demonstrates a generally linear trend. However, this trend is a somewhat vague with larger data distances. As expected, it is the deviations from this trend that indicate poor topologies. The badly organised map shows extensive spreading away from the ideal line; the adequate map also demonstrates this quality, but to substantially reduced extent.

The $\mathcal{Q}$ metric values for each of these plots gives an indication of the quality of the mapping provided. The well trained map yields a $\mathcal{Q}$ metric value of 1.0×10^{-4} ($\mathcal{T} = 6.0 \times 10^{-3}$, $\mathcal{S} = 1.7 \times 10^{-2}$); the adequately trained map yields a $\mathcal{Q}$ metric value of 2.0×10^{-4} ($\mathcal{T} = 7.9 \times 10^{-3}$, $\mathcal{S} = 2.5 \times 10^{-2}$); the badly trained map $\mathcal{Q}$ metric value of 4.6×10^{-4} ($\mathcal{T} = 1.2 \times 10^{-2}$, $\mathcal{S} = 3.9 \times 10^{-2}$). This is commensurate with the expected relationship between these values: the good map is slightly better than the adequate map (a result of having a more even distribution of training vectors), but both are significantly better than the badly trained map. The use of this quality measure is not limited to the comparison of single values at the end point of training. The quality of a mapping can be evaluated at any time in the training process, giving a representation of the rate at which organisation is occurring. Figure 3.5 is a plot of the $\mathcal{Q}$ metric values of the three cases, evaluated at every epoch during the training process. During initial training, the map quality is low, due largely to poor spreading characteristics. However, as training continues, the $\mathcal{Q}$ metric slowly improves, until the map reaches the asymptotic quality levels described previously.



(a) Well trained map (b) Adequate map (c) Badly trained map

Figure 3.4: Normalised Topology Preservation Graphs for three trained maps. (a) A well trained map — topology is well preserved; (b) An adequately trained map — topology is well preserved, but orientation is not optimal; and (c) A badly trained map, incorporating a twist — a loss of topological information.

It is worth note that similar plots can be generated for $\mathcal{S}$ and $\mathcal{T}$. These plots can be used to interpret the specific cause of a failed organisation, or to evaluate the effect of a training technique on different aspects of topological preservation.

Advantages

The $\mathcal{Q}$ metric has a number of distinct advantages over existing metrics for evaluating the quality of topological organisation. Firstly, and perhaps most importantly, this metric allows for the analysis of mappings produced regardless of the algorithm used to produce them. Since only the raw observable output is used to evaluate quality, the metric is not tied to specific weight structures or data representations. Consequently, it is possible to compare mappings produced with maximal SOMs with those produced by minimal SOMs. It is also possible to compare the rate of learning produced by various update rules.

In addition, the metric can be tuned to specific dimensions, or specific subsets of a data set. By considering only the components of the training vector from a subset of dimensions, it is possible to evaluate the preservation of specific topologies in the overall topology. Alternatively, by drawing a test data set from a specific subset of the full data space, it is possible to test the topological preservation of a specific component of a data set. This is not particularly relevant for simple topologies like the grid, but in more complex data sets with high dimensionality and many significant subsets, this can be a useful feature (one that will be exploited later in this thesis).

The metric also has value in its parts. Since the metric is the composite of two

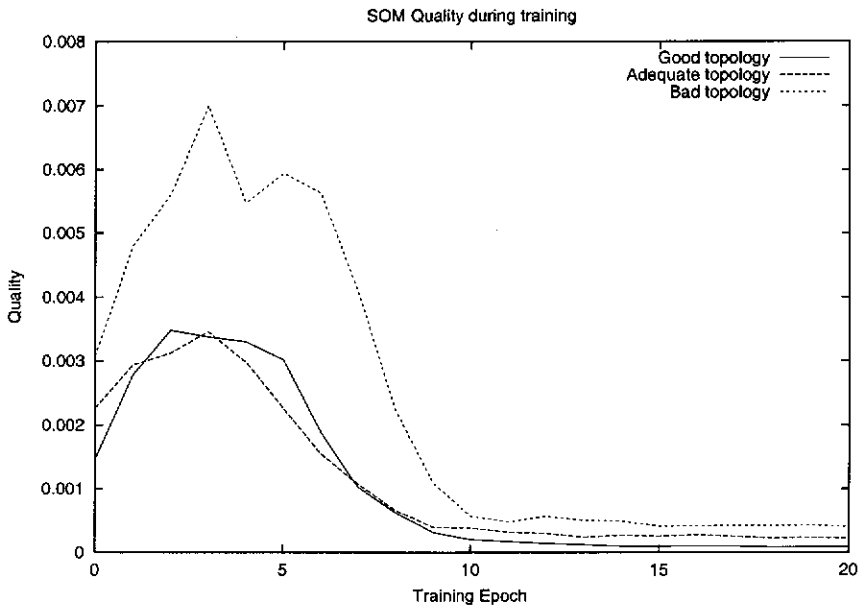


Figure 3.5: SOM Quality during training for the 'good', 'adequate' and 'bad' maps.

separate descriptors, it is possible to get descriptors of two specific underlying characteristics of the mapping, as well as an overall quality description. This can be useful while establishing the cause of the failure of a training process, or in optimising specific aspects of a new training algorithm.

Disadvantages

One limitation of the $\mathcal{Q}$ metric technique is the fact that it is an empirical measure. Rather than being a single canonical value drawn from the weight space of the SOM, the calculation of the $\mathcal{Q}$ metric value requires the selection of a representative testing set to empirically evaluate the topological relationships within the SOM with respect to a specific data set. As such, care must be taken to choose an appropriate testing data set. If the testing data set spans a subset of the training data or spans potential data space not covered by the training data, the resulting $\mathcal{Q}$ metric value will not be representative of the underlying topology learned by the SOM. Using the training data set as the testing set is one potential method which can overcome this problem.

An alternative to the use of the training set for computing the $\mathcal{Q}$ metric is the use of a randomly generated testing set, such as is used with the simple grid data set. However, this introduces a new concern. When using such a testing data set, it must be remembered that the data set is a statistical sampling of the underlying

topology. As with all techniques which utilise statistical sampling, sample size is a potential concern. When using a randomly generated test set, care must be taken to choose an test set of an appropriate size. A failure to do so should be easily identifiable through random fluctuations in the value of the metric once the topology has visibly converged. A test set of insufficient size will not provide enough points to average out the random fluctuation associated with test set selection. The use of a larger test set should eliminate this fluctuation.

Although the problems of statistical sampling can be obviated by selecting a testing set consisting of a large number of testing patterns, the use of a large testing set does reveal another limitation of the $\mathcal{Q}$ metric technique: the complexity of the algorithm. The complexity of the metric calculation is $O(n^2)$ with number of test vectors. This is unavoidable, as every training vector must be compared with every other training vector. While this did not prove a practical limitation during testing performed in this thesis, it could become an issue if large testing sets are required. It should also be noted that this technique is no more computationally inefficient than many other proposed measures of topological quality.

Another limitation of the $\mathcal{Q}$ metric technique is the sensitivity of the technique at the start of training when used on sparse testing sets. During initial stages of training, the random influences introduced to provide entropy for the learning process can lead to extremely poor placement of some training vectors. This results in extreme values for $\mathcal{T}$ and $\mathcal{S}$. These extreme values are then multiplied to provide a $\mathcal{Q}$ metric component. However, extremely large values of $\mathcal{T}$ can be accompanied by very small values of $\mathcal{S}$ (and vice versa). After multiplication, this can result in a misleadingly small value for the $\mathcal{Q}$ metric.

If the testing set is large, this does not pose a problem, as the effect of a small number of poor $\mathcal{Q}$ metric components is averaged over the large number of reasonable components. However, if the training set is small, the averaging process does not remove the effect of a bad sample. As a result of this limitation, small (that is, 'good') $\mathcal{Q}$ metric values registered at the start of training must be viewed with caution. A map representation should only be considered good if the $\mathcal{Q}$ metric reaches an asymptotic value.

A final limitation of the $\mathcal{Q}$ metric technique is the use of the constant terms in the evaluation of $\mathcal{T}$ and $\mathcal{S}$. The values of 0.3 and 0.1 were determined as a result of empirical testing. No formal justification of these values has been developed. The

$\mathcal{Q}$ metric technique would be greatly strengthened by the development of a formal mechanism for selecting these constant terms.

3.5 Conclusion

In this chapter, it was proposed that the use of an architecture which models the lifelong development observed in nature would enable the development of improved representations of a data domain. Furthermore, it was proposed that by controlling plasticity in concept representation, it is possible for a system to adapt to changes in the composition of the data domain. The biological metaphor which underpins maximal SOMs suggests that SOM algorithms are an appropriate tool for investigating lifelong development mechanisms.

In addition, a metric was proposed for evaluating the topological output of the self-organising process. This metric, called the $\mathcal{Q}$ metric, does not attempt to examine the weight space of the SOM or exploit aspects of the training algorithm. Rather, it quantifies the observable output of the SOM by comparing the distances between vectors in the training set with the distance between the SOM neurons selected as winners by those vectors.

In following chapters, a number of algorithmic extensions to Kohonen's maximal SOM algorithm will be proposed. The purpose of these extensions will be to implement the lifetime learning and development model proposed in this chapter. In Chapters 4 and 5, an algorithm is developed which separates the juvenile phase of development from the remaining stages. An algorithm which allows for the continuous learning of new training examples after the juvenile phase is presented in Chapter 6. In Chapter 7, an algorithm is presented which allows for the gradual introduction of progressively more complex training examples, such as is observed during adolescence. Finally, in Chapter 8, an algorithm is presented which is able to restrict the reorganisation of a knowledge representation during the adulthood. The metric proposed in this chapter will be used to evaluate the success or failure of these algorithmic extensions.

Chapter 4

Neighbourhood and Organisation

> He [Arthur] had been told that when looking for a good oracle it was best to find the oracle that other oracles went to, but he was shut. There was a sign on the entrance saying, 'I just don't know any more. Try next door, but that's just a suggestion, not formal oracular advice.'
>
> — *Mostly Harmless* (Adams, 1992)

4.1 Introduction

The SOM algorithm is based upon the control of two key parameters: gain and neighbourhood size. Through careful decay of these parameters it is possible to shape an initially random set of SOM weights into an organised topological representation of any data set with an underlying topology.

To date, analysis of SOM performance has concentrated on optimising learning gain. The role played by the size, form and decay of the neighbourhood function within Kohonen's algorithm has been largely ignored or merely alluded to in the vast majority of literature. However, no evidence has been presented to support the assertion that the role played by neighbourhood size is indeed trivial. The size, form and decay of neighbourhood interactions may play a much more significant role in the development of Kohonen style topological maps than has previously been reported.

In this chapter, the role played by neighbourhood interactions in the organisation process is established. In Section 4.2, a mathematical model is presented which consolidates the biological model of Von der Malsburg, and the computational model of Kohonen. This model is used to demonstrate the significance of lateral connection strategies on the process of self-organisation. In particular, the significance of Laplacian lateral connections is established. Lateral connections of this type form the foundation of many self-organising algorithms, including Kohonen's.

In Section 4.3, an investigation is performed into the role played by the size of the Laplacian neighbourhood on the organisation process. The computational model used in this section is similar to that used by Von der Malsburg. Using this model, some fundamental characteristics of the relationship between neighbourhood size and SOM output states are demonstrated.

4.2 Theoretical Non-WTA

Traditional Kohonen style networks are described as being Winner-Take-All (WTA); during each training iteration, a single winner is selected from the map, about which weight reinforcement takes place. However, this algorithm — including its assumptions about the importance (or lack thereof) of neighbourhood size — is an algorithmic abstraction of the behaviour observed in non-WTA systems (Kohonen, 2001). In non-WTA systems, such as those used to initially demonstrate self-organising behaviour (Von der Malsburg, 1973)(Willshaw and Von der Malsburg, 1976), winners are selected through a complicated system of feedback, balancing individual neuron dynamics and the dynamics of the map as a whole. Kohonen's algorithm is an algorithmic abstraction of an underlying dynamic process.

In performing this abstraction, it becomes possible to state the self-organising algorithm as a single iterative weight update rule. However, this eliminates many of the subtleties of the self-organisation process. For example, a dynamic update method allows for no-winner and multiple-winner outcomes. Such outcomes cannot develop if a single winner is selected externally. These subtleties are largely due to the complex neighbourhood interactions which occur during the organisation process. Therefore, to clearly demonstrate the role played by neighbourhood in the organisation process it is necessary to return to a dynamically generated, non-WTA map.

In this section, we present a mathematical model of a non-WTA self-organising neural system. This mathematical model is then used to demonstrate that a system which has Laplacian lateral connections will, through a process of repeated iteration, provide a Gaussian output centered over the winning node. This gives a similar result to the manual selection of a maximum. However, in this model, winner selection is an inherent characteristic of the system, precluding the need for a meta-level decision to be made.

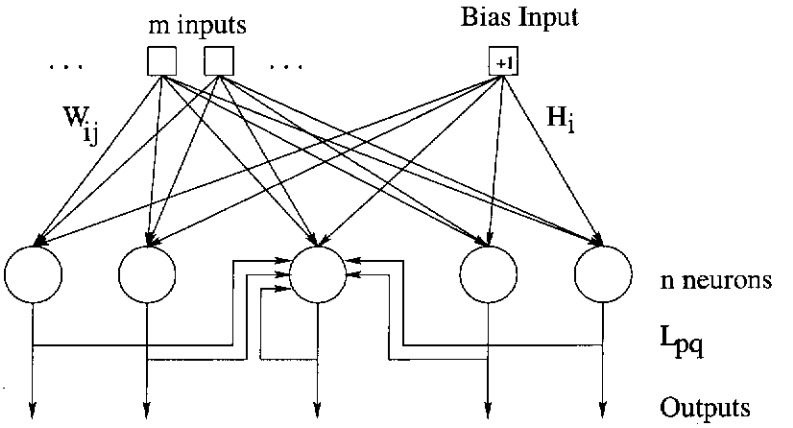


Figure 4.1: The model neuron used in this experiment. Information flows in the direction of the arrows. Note: the lateral connections for all but the centre neuron have been omitted from this diagram for clarity.

4.2.1 Continuous Time Model

Consider the theoretical model neuron pictured in Figure 4.1. Each of the n neurons has an internal voltage u_i which is a function of the input and lateral connections and an output voltage $v_i = \sigma(u_i)$, where σ is a sigmoid function. This neuron receives a vector $\vec{X}$ of m independent inputs x_j; each neuron i is connected to these inputs via an independent weight w_{ij}, forming an $n \times m$ matrix of input connections $\mathbf{W}$. In addition, each neuron p belongs to a plane of other neurons, and is connected to every other neuron q via a connection of strength L_{pq}. These connections form an $n \times n$ matrix of lateral connections $\mathbf{L}$. Included in these lateral connections is a connection to the neuron itself.

Using these definitions, the voltage of neurons in a SOM over time t can be modeled as:

$$\frac{\partial u_i}{\partial t} = -u_i + \sum_{k=1}^{m} w_{ik} x_k + \sum_{k=1}^{n} L_{ki} \sigma(u_k) + h_i \tag{4.1}$$

In this formulation, h_i is a neuron activation bias. The system has the initial condition $u_i(0) = 0 \, \forall i$.

To simplify the analysis of this system, the sigmoid function is approximated by a linear function:

$$\sigma(x) = x \tag{4.2}$$

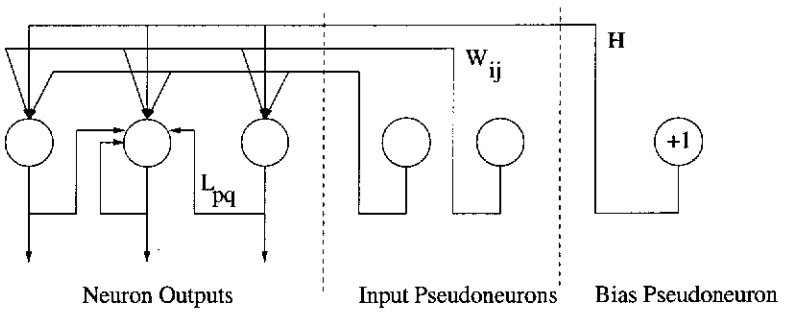


Figure 4.2: The rearrangement of neurons used to simplify the formulation of the SOM problem. Input and neuron bias are considered to be pseudoneurons within the SOM layer. Note: many lateral connections within the true neurons have been omitted from this diagram for clarity.

This is a reasonable approximation, as the sigmoid function is almost linear for all except extreme input values. At these values the sigmoid function is truncated to 0 or 1, to curb exponential growth. If we ignore exponential growth of the system, and consider only the near linear region of the sigmoid, this simplification allows Equation 4.1 to be reposed as:

$$\frac{\partial u_i}{\partial t} = -u_i + \sum_{k=1}^{m} w_{ik} x_k + \sum_{k=1}^{n} L_{ki} u_k + h_i \tag{4.3}$$

If we define $\vec{U}$ as the vector of voltages u_i, this formulation simplifies to:

$$\frac{\partial \vec{U}}{\partial t} = -\vec{U} + \mathbf{W}\vec{X} + \mathbf{L}\vec{U} + \vec{H} \tag{4.4}$$

To solve this equation for $\vec{U}$, we use an Euler approximation over a time step of size ∂t:

$$\begin{aligned} \vec{U}(t+\partial t) &= \vec{U}(t) + \partial t \left.\frac{\partial \vec{U}}{\partial t}\right|_t & (4.5)\\ &= \vec{U}(t) + \partial t \left[-\vec{U}(t) + \mathbf{W}\vec{X} + \mathbf{L}\vec{U}(t) + \vec{H}\right] & (4.6)\\ &= \left[(1-\partial t)\epsilon^n + \partial t L\right]\vec{U}(t) + \partial t \mathbf{W}\vec{X} + \partial t \vec{H} & (4.7) \end{aligned}$$

Where ϵ^k is the identity in $\Re^k$.

To further simplify this formulation, let:

$$\vec{\Psi} = \begin{bmatrix} \vec{U} & \vec{X} & 1 \end{bmatrix}^T \tag{4.8}$$

This simplification represents the bias and input neurons as pseudoneurons within the SOM layer. These pseudoneurons receive no lateral input; each pseudoneuron has a connection to each of the normal neurons. Figure 4.2 shows this arrangement graphically. This simplifies Equation 4.7 to:

$$\begin{bmatrix} \vec{U}(t+\partial t) \\ \vec{X} \\ 1 \end{bmatrix} = \begin{bmatrix} (1-\partial t)\epsilon^n + \partial t\mathbf{L} & \partial t\mathbf{W} & \partial t\vec{H} \\ 0 & \epsilon^m & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{U(t)} \\ \vec{X} \\ 1 \end{bmatrix} \tag{4.9}$$

$$\begin{bmatrix} \vec{U}(t+\partial t) \\ \vec{X} \\ 1 \end{bmatrix} = \begin{bmatrix} (1-\partial t)\epsilon^n + \partial t\mathbf{L} & \partial t\mathbf{W} & \partial t\vec{H} \\ 0 & \epsilon^m & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{\Psi} \tag{4.10}$$

$$\vec{\Psi}(t+\partial t) = \mathbf{M}\vec{\Psi(t)} \tag{4.11}$$

$$\vec{\Psi}(t+2\partial t) = \mathbf{M}\vec{\Psi}(t+1) = \mathbf{M}^2\vec{\Psi}(t) \tag{4.12}$$

$$\vec{\Psi}(t+k\partial t) = \mathbf{M}^k\vec{\Psi}(t) \tag{4.13}$$

$$\vec{\Psi}(k\partial t) = \mathbf{M}^k\vec{\Psi}(0) \tag{4.14}$$

The stable state of the system is the infinite limit of this process; i.e.,

$$\lim_{k\to\infty,\partial t\to 0} \mathbf{M}^k\vec{\Psi}(0) \tag{4.15}$$

4.2.2 Discrete Time Model

While the continuous case is interesting from a biological point of view, the discrete version of the algorithm is of more computational interest. The discrete case is almost identical to the continuous case of Equation 4.1, expressed as discrete state variables:

$$\vec{U}_{t+1} = L\sigma(\vec{U}_t) + \mathbf{W}\vec{X} + \vec{H} \tag{4.16}$$

or, with the benefit of the $\sigma(x) = x$ approximation:

$$\vec{U}_{t+1} = L\vec{U}_t + \mathbf{W}\vec{X} + \vec{H} \tag{4.17}$$

The analysis of the discrete system can be performed as a special case of the continuous system, where $\partial t = 1$. Substituting into Equation 4.11, this yields:

$$\begin{bmatrix} \vec{U}_{t+1} \\ \vec{X} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{L} & \mathbf{W} & \vec{H} \\ 0 & \epsilon^m & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{U}_t \\ \vec{X} \\ 1 \end{bmatrix} \tag{4.18}$$

$$\vec{\Psi}_{t+1} = \begin{bmatrix} \mathbf{L} & \mathbf{W} & \vec{H} \\ 0 & \epsilon^m & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{\Psi}_t \tag{4.19}$$

$$\vec{\Psi}_{t+1} = \mathbf{M}\vec{\Psi}_t \tag{4.20}$$

$$\vec{\Psi}_{t+2} = \mathbf{M}\vec{\Psi}_{t+1} = \mathbf{M}^2\vec{\Psi}_t \tag{4.21}$$

$$\vec{\Psi}_{t+k} = \mathbf{M}^k\vec{\Psi}_t \tag{4.22}$$

$$\vec{\Psi}_k = \mathbf{M}^k\vec{\Psi}_0 \tag{4.23}$$

This system can be analysed using eigenvectors; if $s_{1\ldots n}$ and $\lambda_{1\ldots n}$ are the eigenvectors and eigenvalues, respectively, of $\mathbf{M}$, and $\mathbf{S} = \begin{bmatrix} \vec{s}_1 & \vec{s}_2 & \cdots & \vec{s}_n \end{bmatrix}$, and

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \\ \vdots & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}, \tag{4.24}$$

we can use the Eigenvector relation $\mathbf{M}^k = \mathbf{S}\mathbf{\Lambda}^k\mathbf{S}^{-1}$ to yield a computationally simple output state of the system:

$$\vec{\Psi}_k = \mathbf{S}\mathbf{\Lambda}^k\mathbf{S}^{-1}\vec{\Psi}_0 \tag{4.25}$$

The final output state of the system is given by $\lim_{k\to\infty} \vec{\Psi}_k$.

This process ignores the fact that sigmoid scaling occurs during relaxation, not just as a final step. Under the full model (Equation 4.16), a sigmoid is used to scale lateral inputs, so as to prevent exponential explosion. σ is a monotonically increasing function, so the relative ordering of inputs remains unchanged as a result of this scaling process. Some shaping of relative magnitudes occurs as a result of the nonlinear nature of the sigmoid function but the largest output will remain the largest throughout scaling. The infinite limit of the process described by Equation 4.16 can therefore be derived

by comparing relative magnitudes of outputs of the function $\vec{\Psi}(t)$.

This formulation allows us to examine the convergence properties of a given lateral connection kernel. In the discrete system, $\vec{\Psi}$ will grow exponentially if $\exists\lambda > 1$; these values, when raised to a large power will increase. These values combine with S to determine the output state of the system. In systems where $\lambda_i \leq 1\ \forall i$, stagnant or dissipative solutions exist, as no components exist to interact with S. This provides a simple test for the ability of a lateral interaction kernel to converge on a useful solution; M of Equation 4.23 must possess at least one eigenvalue > 1.

It should be noted that this is a somewhat trivial analysis. A number of sophisticated theoretical analyses exist in the literature (Wilson and Cowan, 1973) (Amari, 1977). However, this analysis suffices to demonstrate the important role that the lateral interaction function can play in the stable states assumed by self-organising maps.

Testing Laplacian Connection Kernels

The theoretical formulation presented in the previous section operates on a neural construct of arbitrary size and demonstrates whether a given lateral connection kernel will produce stable output states. However, the specific output state resulting from a given kernel cannot be determined without considering a specific neural construct and input pattern. In this section, a 15 input, 1D map of 100 neurons connected using a Laplacian connection kernel is used in conjunction with a simple input spike to confirm empirically the results obtained in the previous section, and to demonstrate the wide range of stable output states which can be generated by simply changing the lateral interaction kernel.

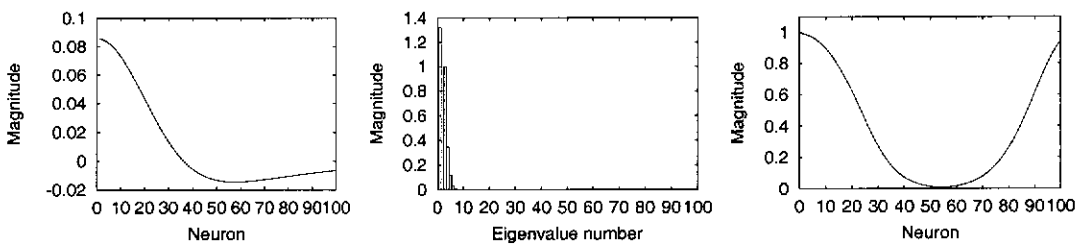
In this experiment, the Laplacian connection weights follow the relation:

$$L_{ij} = (1+a)\mathcal{G}\left(\|i-j\|, r\right) - a\mathcal{G}\left(\|i-j\|, br\right) \tag{4.26}$$

where L_{ij} is the connection strength between neurons i and j, r is the size of the neighbourhood function, a is the magnitude of the negative Laplacian component, b is the spread of the negative component of the Laplacian, and $\mathcal{G}$ is the Gaussian function:

$$\mathcal{G}(x,\sigma) = exp(-\frac{x^2}{2\sigma^2}) \tag{4.27}$$

In addition, these Laplacian weights are normalised so that $\sum_i L_{ij} = 1$. This is done to prevent edge effects.



(a) (b) (c)

Figure 4.3: Stable state of a lateral interaction kernel, $a = 0.3$, $b = 3$, $\sigma = 20$: (a) Lateral interaction function for Neuron 1; (b) Eigenvalues; and (c) Stable output state resulting from a single pulse input.

Now consider a specific instance of this lateral connection function, in which $a = 0.3$, $b = 3$, and $\sigma = 20$. W is instantiated with random values on the range $[0 : 0.01]$; $\vec{H}$ is instantiated to $\vec{0}$. Figure 4.3 shows the results of analysis using the proposed technique. Figure 4.3(a) shows a sample set of lateral connection strength values — a classic Laplacian curve. Figure 4.3(b) shows the eigenvalues of this system; the single $\lambda > 1$ shows that this system will converge to a useful result.

The results on output can be simulated by providing a sample training pattern; say, a single pulse ($\vec{X} = [0\ 0.5\ 1.0\ 0.5\ 0\ \cdots 0]^T$). The result of using this training pattern can be seen in Figure 4.3(c). This plot of neuron outputs clearly demonstrates the ability of a Laplacian kernel to produce Gaussian-like output, such as is required to perform self-organisation; an alternative to the artificially fitting of a Gaussian about a winner, in the winner-take-all mechanism of Kohonen. In Figure 4.3(c) the training pattern has a peak on input 3; almost identical results are obtained using training patterns with peaks in other positions.

Similar results are shown in Figures 4.4–4.6. In Figure 4.4, a system with large σ ($\sigma = 40$) is tested. This clearly demonstrates that if a wide Laplacian is used, it acts as a feature smoother, rather than peak enhancer: the wide Laplacian reinforces all neurons, rather than selecting a single neuron and its neighbourhood as a winner. The eigenvalues for this system give an indication of this behaviour. The largest eigenvalue is a value of $\lambda = 1$; consequently, the system is stagnant. Repeated iteration of this system will not cause a stable pattern to emerge in the outputs.

Figure 4.5 considers the case of a small σ. If a small σ is chosen ($\sigma = 4$), such that the range of the interaction function is vastly less than the size of the map, multiple

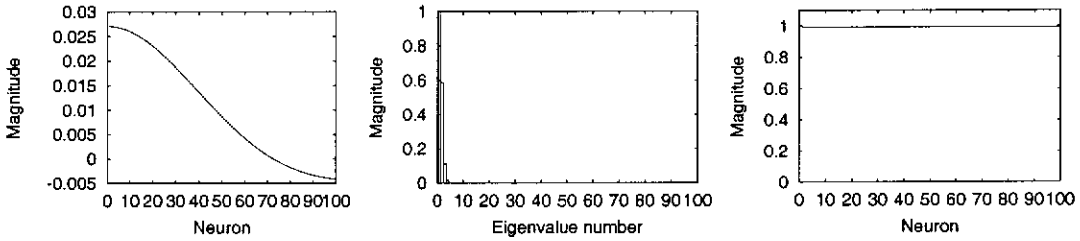


Figure 4.4: Stable state of a lateral interaction kernel, $a = 0.3$, $b = 3$, $\sigma = 40$: (a) Lateral interaction function for Neuron 1; (b) Eigenvalues; and (c) Stable output state resulting from a single pulse input.

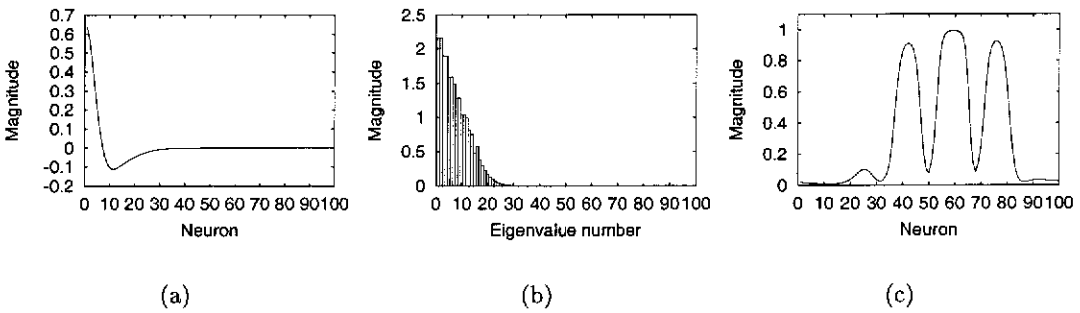


Figure 4.5: Stable state of a lateral interaction kernel, $a = 0.3$, $b = 3$, $\sigma = 4$: (a) Lateral interaction function for Neuron 1; (b) Eigenvalues; and (c) Stable, but unusual (non-Gaussian) output state resulting from a single pulse input.

peaks can and will result. This multiple peak state is a stable state of the system. However, it is not a useful stable state for global topological organisation as it does not provide a single neighbourhood of positive support. It may, however, yield other interesting topological effects.

As an exercise, the $\sigma = 1$ case was tested. The results of this trial can be seen in Figure 4.6. This lateral connection kernel (Figure 4.6(a)) is essentially the same Laplacian used by Kohonen in his Winner-Take-All system. This system has a single, very large eigenvalue (Figure 4.6(b)), yielding the expected result of a single winner on output (Figure 4.6(c)).

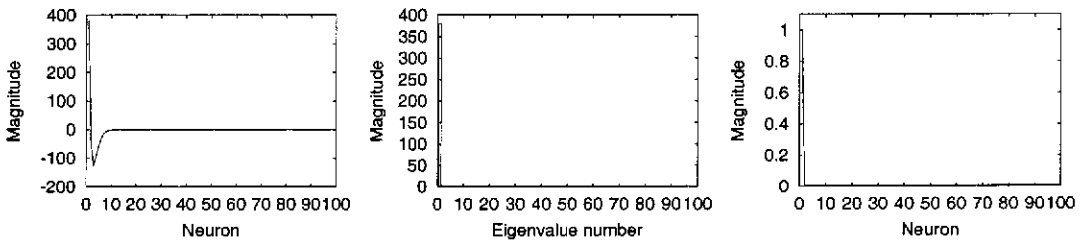


Figure 4.6: Stable state of a lateral interaction kernel, $a = 0.3$, $b = 3$, $\sigma = 1$: (a) Lateral interaction function for Neuron 1; (b) Eigenvalues; and (c) Stable output state resulting from a single pulse input.

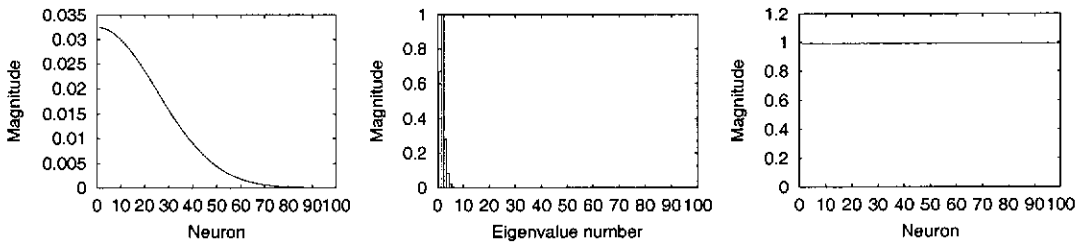
Testing Other Lateral Connection Kernels

The mathematical model presented in this chapter is an empirical tool. Given the current state of an SOM, and a set of lateral connections, it will give an indication as to whether a given combination of weights and lateral connections will provide conditions conducive to learning topologies (as defined by Amari (1980) and other theoretical studies). In these tests, a random set of input weights was used. The results obtained in these experiments therefore relate to the formation of output states on untrained maps. This technique can be used to investigate novel lateral connection strategies by giving insight into the output patterns resulting from these strategies.

Three such novel lateral connection strategies can be seen in Figures 4.7–4.9. Figure 4.7 demonstrates the effect of using a Gaussian, rather than a Laplacian lateral interaction kernel. The smoothing effect of the Gaussian kernel is obvious, as the output state of the system is a state of uniform excitation. This is a similar effect to the use of a wide Laplacian kernel, seen in Figure 4.4.

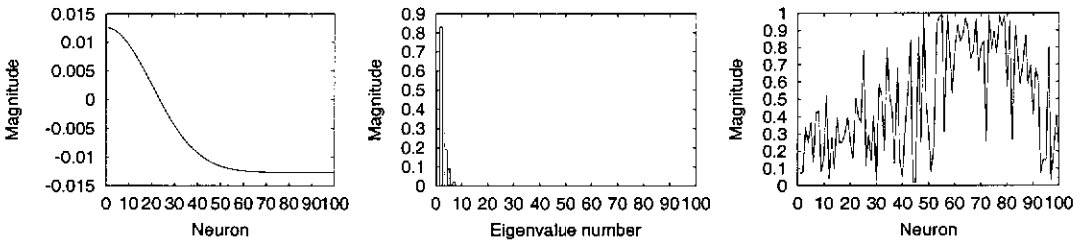
Figure 4.8 demonstrates a variant on the simple Gaussian lateral connection — a Gaussian with a negative shift superimposed. This is equivalent to a Laplacian kernel for which negative support does not decay with distance. This system does not have any eigenvalues greater than 1; as a result, the output state of this system is highly unstable.

Figure 4.9 demonstrates an centre surround kernel. This kernel is the inverted form of a Laplacian kernel; whereas a Laplacian kernel has a positive central region surrounded a negative region, the centre surround kernel has a negative centre region, surrounded by a positive region. This kernel can be seen to generate a stable output



(a) (b) (c)

Figure 4.7: Stable state of a Gaussian lateral connection kernel: (a) Lateral interaction function for Neuron 1; (b) Eigenvalues; and (c) Saturated output state resulting from a single pulse input.



(a) (b) (c)

Figure 4.8: Stable state of a shifted Gaussian lateral connection kernel: (a) Lateral interaction function for Neuron 1; (b) Eigenvalues; and (c) Unstable output state resulting from a single pulse input.

state. As with the narrow Laplacian, this output state does not provide result in a single neighbourhood of positive support; therefore, this kernel will not cause a global topological organisation to develop. However, this kernel may yield other interesting topological effects.

There is one significant limitation to this technique. While it is possible to use this technique to demonstrate the form of the stable output state (if any) that will arise from a specific lateral interaction kernel, it does not indicate the form of the topological mapping that will result from use of that kernel. A further analysis of the stable output state is required to establish the topological mappings resulting from these eccentric stable output states.

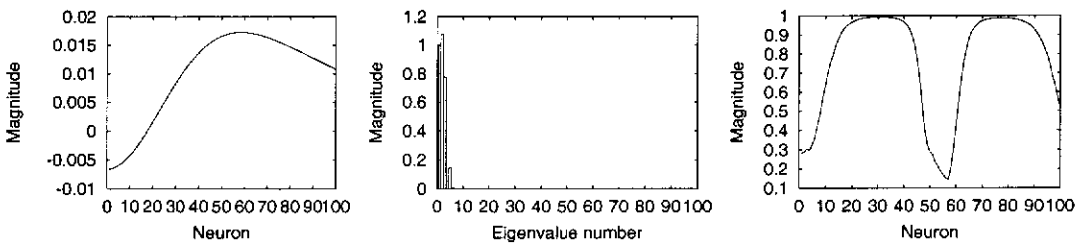


Figure 4.9: Stable state of a centre surround lateral connection kernel: (a) Lateral interaction function for Neuron 1; (b) Eigenvalues; and (c) Stable, but unusual (non-Gaussian) output state resulting from a single pulse input.

4.3 The Role of Neighbourhood Size

In the previous section, a method was presented for demonstrating the manner in which an iterative mechanism with a lateral connection kernel can be used to provide a Gaussian fitted about a winning neuron, without the need to select the winner through some external process. In this section, a similar iterative mechanism will be used to demonstrate some key features of neighbourhood size change during the learning and development process of topographical self-organising maps.

4.3.1 Implementation

The algorithm used to implement the SOM model is very similar to that used by Von der Malsburg. The learning process consists of two stages: an internal iterative loop, which allows lateral connections to relax the output of each neuron; and an external loop which controls the Hebbian update based upon output values. Given a set of input vectors $\vec{X}$, this algorithm learns a topographical mapping for the set of weights w_{ij}. Lateral connections are defined in a kernel $\mathbf{L}$, defined as in Equation 4.26, with $a = 0.3$ and $b = 3$. The size of weight update for neuron i is proportional to the output of neuron i in response to the training pattern; this update is a normalised Hebbian update, drawn from (Kohonen, 1982). Each cycle of the outer Repeat-Until loop (Algorithm 5) is an epoch of the training regime.

The algorithm for the inner loop (the function **EvaluateMapOutput**) is not as straightforward. This is a result of the same algebraic explosion problem observed in the previous section. To counter this algebraic explosion, a process of normalisation

Algorithm 5 Mainline algorithm to implement SOM ordering.

repeat
  Evaluate changes in learning parameters α and σ
  while $\exists$ a pattern which has not been presented this epoch **do**
    Randomly select a training pattern $\vec{x}$ which has not been presented this epoch
    EvaluateMapOutput $\vec{v}$ for training pattern
    Update weights according to rule:

$$\vec{w}_i(t+1) = \frac{\vec{w}_i(t) + \alpha(t)\vec{x}v_i}{\|\vec{w}_i(t) + \alpha(t)\vec{x}v_i\|}$$

  end while
until finished training

Algorithm 6 Computationally normalised version of the **Evaluate_Map_Output** algorithm.

Evaluate_Map_Output (L,$\vec{u}$,β,W,$\vec{X}$,N)
$\vec{u} = \vec{0}$
for $C = 1 \cdots N$ **do**
  $\vec{u} = \mathbf{W}\vec{X} + \beta\mathbf{L}\vec{u}$
  $\vec{u} = \frac{\vec{u}}{\|\vec{u}\|}$
end for
$\vec{v} = \sigma(\vec{u})$
return $\vec{v}$

is required. This normalisation can be performed using a number of methods. Two versions of the inner iterative loop are presented here.

In the first version (Algorithm 6), algebraic normalisation is performed at the end of each iteration. At the conclusion of the iteration process, the normalised output is passed through a sigmoid function to provide maximum and minimum saturated outputs. In the second version (7), there is no explicit normalisation process - the process of normalisation is performed on each iterative loop by a sigmoid function tuned to the dynamic range of the iterative process.

Algorithm 6 is much easier to control. Given an input data vector $\vec{X}$ and a weight matrix $\mathbf{W}$, the algorithm calculates the output $\vec{v}$. The strength of feedback is controlled by the lateral efficacy parameter β; the strength of lateral connections is modeled by $\mathbf{L}$, a function of $|i-j|$. At each iteration, the internal voltage is dynamically scaled to prevent algebraic explosion. The final output value is the result of passing a sigmoid over the internal voltage $\vec{u}$; parameters for this sigmoid are custom selected to suit the dynamic range of $\vec{u}$.

This process of algebraic normalisation guarantees a consistent dynamic range on output, permitting the use of a fitted sigmoid for the final transformation of output.

Algorithm 7 Biologically plausible version of the **Evaluate_Map_Output** algorithm.

Evaluate_Map_Output $(L,\vec{u},\beta,W,\vec{X},N)$
$\vec{u} = \vec{0}$
$\vec{v} = \vec{0}$
for $C = 1 \cdots N$ **do**
  $\vec{u} = \mathbf{W}\vec{X} + \beta\mathbf{L}\vec{v}$
  $\vec{v} = \sigma(\vec{u})$
end for
return $\vec{v}$

However, this model is somewhat artificial, as normalisation is a meta-level function, requiring knowledge of the raw output of all other neurons in the system.

The second model (Algorithm 7) is biologically more plausible, but it requires significantly more effort to tune. Given an input data vector $\vec{X}$ and a weight map matrix $\mathbf{M}$, the algorithm calculates the output $\vec{v}$. The strength of feedback is controlled by the efficacy parameter β, and the strength of lateral connections is modeled by $\mathbf{L}$, a function of $|i-j|$. At each iteration, the output value is the result of passing a sigmoid over the internal voltage $\vec{u}$; parameters for this sigmoid are selected to suit the dynamic range of $\vec{u}$. Note that in this model it is the output voltage $\vec{v}$, not the internal voltage $\vec{u}$ that is used for feedback. In biological systems, this tuning would be performed by intra-neuron monitoring of input, output and weight conditions. However, in this computational framework, tuning of this kind is not possible.

Aside from this minor difference in implementation, the differences between the two algorithms are negligible. The output states obtained from the two systems are analogous. Some slight differences can be observed but this can be attributed to slight differences in the tuning of the sigmoid functions.

It should be noted that this algorithm does not require a winner to be chosen or a Gaussian-like function to be superimposed for the purposes of Hebbian learning. The algorithm, as demonstrated in Section 4.2, naturally converges to a Gaussian-like function about the map area of best fit for each input pattern, given a suitable Laplacian lateral connection kernel.

In the following experiments, the first, algebraically normalised version is utilised. This is done for no reason other than convenience. The biological version is presented for completeness, in order to demonstrate that the self-organising process *could* be performed using a completely biologically plausible model, without the need for any meta-level functions.

4.3.2 Experimental Results

In these experiments, a 15 input, 100 neuron SOM was used. These 100 neurons were arranged into a ring, so neurons on one end were attached to the other. This was achieved by wrap around in the lateral interaction kernel. The Laplacian **L** of Equation 4.26 was used for this kernel, with $a = 0.3$, $b = 3$, and σ varying through the experiment. This kernel was then normalised. 10 internal iterations were used to evaluate the neuron output values. In the learning algorithm, the gain parameter was set at a constant $\alpha = 0.3$, to minimize the degrees of freedom in the experiment. Lateral efficacy β was set at 1. This SOM was trained using the simple line data set described in Section A.2.

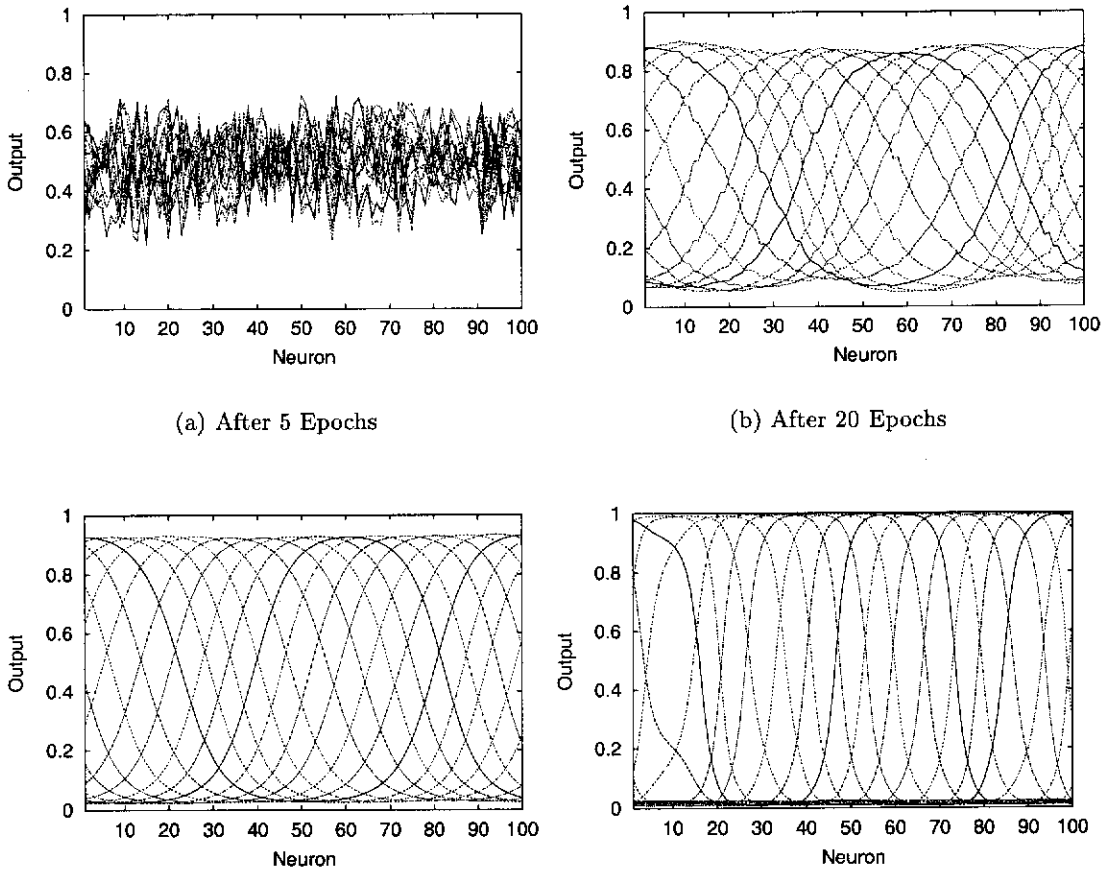
Two similar experiments were performed, to establish the effect of a reduction in neighbourhood size on the map. In both experiments, the SOM was first trained for 50 epochs with a constant neighbourhood size of $\sigma = 20$. This value for σ was shown to develop a stable output state in Figure 4.3. After this initial training, the neighbourhood size was reduced to lower value.

The output states resulting from this initial training can be seen in Figures 4.10(a)–4.10(c). At the conclusion of this initial training, the map can be seen to converge into a stable state, with good topological organisation. Each curve in Figure 4.10(c) represents the output response of a single training pattern; the maxima of these curves are seen to be evenly distributed across the 1D SOM.

In the first experiment, the neighbourhood size was reduced to $\sigma = 4$. As a result of this reduction, the neuron output responses can be observed to sharpen significantly (compare Figures 4.10(c) and 4.10(d)). This highlights the benefit of a reduction in neighbourhood size to precision on a map.

In the second experiment, the neighbourhood size was reduced to $\sigma = 2$. However, the result of this reduction was not as positive as in the first experiment. Instead of sharpening the output states, each output state is split into multiple peaks. The effect of this splitting can be seen in Figure 4.11. The effect is mild at first, but the splitting behaviour becomes more severe as training with the smaller σ continues.

To clarify the role played by the period of training with a large neighbourhood size, an attempt was made to train the same network without the initial training period with a $\sigma = 20$ kernel. Instead, all training was performed with a kernel of size $\sigma = 4$. However, this training regime was unable to form a global topographical organisation — instead, a large number of zones of local organisation formed. This indicates that



(a) After 5 Epochs

(b) After 20 Epochs

(c) After 40 Epochs

(d) After 100 Epochs

Figure 4.10: Output response over time for a non-WTA network. Each line is the output response for a single training pattern. $\sigma = 20$ for epochs 1–50; $\sigma = 4$ for epochs 51 onwards.

a period of training with a large neighbourhood size is essential to the formation of global topology; small neighbourhood size kernels can be used to improve precision on an existing map, but not to establish a map from scratch.

4.4 Controlling Organisation with the Neighbourhood

This experiment highlights a number of key features of SOM training. Firstly, it highlights the importance of neighbourhood size to the formation of topographic maps. In order to produce a global topographical organisation, an initially wide spanning - empirically, it would seem reasonable to hypothesize that the 6σ diameter (3σ, or 99.5% of area, both sides of the mean) of the lateral interaction kernel must slightly exceed the size of the map. This concurs with the experimental results of Von der Malsburg (1973) and Willshaw and Von der Malsburg (1976), and one of the conclusions of Amari

Kohonen's, where the artificial selection of a maximum prevents the formation of local topologies in this manner. However, if we intend to use the entire output response, the increase in precision brought about by training with a reduced neighbourhood size is an essential feature. Care must therefore be taken to not decay neighbourhood size by too large an amount. This concurs with the results of Zell (1994). The optimal rate of decay for the kernel neighbourhood size remains to be determined.

It is also interesting to note that these self-organising systems converge *without* the need for a sophisticated scheme of learning gain decay. This seems to contradict the vast majority of literature (Kohonen, 2001) (Cottrell *et al.*, 1998) (Mulier and Cherkassky, 1994), which emphasizes the importance of learning gain, leaving neighbourhood size as a peripheral concern. This is not to imply that maps *cannot* form without neighbourhood size decay. Takeuchi and Amari (1979) demonstrates that topological maps will form if the size of connections is fixed. However, this work demonstrates empirically that the rate of decay of the lateral interaction kernel, and the form of this lateral interaction kernel plays a significant role in the process of topology formation.

4.5 Conclusion

In this chapter it was shown that neighbourhood interactions play a significant role in the organisation process. Using a mathematical model, it was shown that the shape of the lateral interaction kernel has a significant effect on the style of organisation that takes place on a SOM. A biologically inspired computational model was then used to show that in the case of Laplacian lateral connections, the size of the lateral interaction kernel can have a significant effect on the speed of convergence, and the precision of representations. Neighbourhood size also determines whether the map will converge at all.

In the next chapter, these findings about neighbourhood interactions will be applied to a traditional Kohonen SOM in the form of a new set of parameter decay schemes for Kohonen's algorithm.

Chapter 5

Step Neighbourhood Decay

> The History of every major Galactic Civilization tends to pass through three distinct and recognizable phases, those of Survival, Inquiry and Sophistication, otherwise known as the How, Why and Where phases.
>
> For instance, the first phase is characterized by the question 'How can we eat?', the second by the question 'Why do we eat?', and the third by the question 'Where shall we have lunch?'.
>
> — *The Hitch-Hikers Guide to the Galaxy* (Adams, 1979)

5.1 Introduction

Kohonen's algorithm involves two key parameters for operation: learning gain (α), and neighbourhood size (σ). A wide range of decay schemes have been proposed for these parameters. However, the majority of these decay mechanisms concentrate on optimising the decay of learning gain within Kohonen's algorithm. Optimisation of neighbourhood size has been largely ignored in the literature, being largely limited to defining conditions for ordering, not optimising convergence time.

However, as the previous chapter has shown, this is a false assertion: the manner of neighbourhood interaction can have a dramatic effect on the rate and quality of topological organisation that occurs within a SOM.

In this chapter, neighbourhood relationships in Kohonen-style SOMs are investigated, resulting in the development of a new scheme for parameter decay. Section 5.2 presents a study of the role of neighbourhood size in Kohonen-style SOMs. This study shows the effect of using large and small neighbourhood interactions on the organisation process of Kohonen-style SOMs.

As a result of the observations made in this study, a new scheme for parameter decay is suggested. This scheme, known as *Butterworth Step Decay*, involves the use of a constant value for gain, and a smoothed step between a large and a small neighbourhood

size. This scheme provides training times comparable to the best times possible using traditional parameter decay schemes, but does not require *a priori* knowledge of the likely training time for the data set.

The Butterworth Step decay scheme is tested on a range of map sizes in Section 5.3. As a result of these tests, the operational parameters of the scheme are established. The performance of the scheme is compared to traditional decay schemes in Section 5.4. This comparison is performed using the Simple Grid, Questionnaire, and Scotch Whisky data sets.

5.2 Testing the Effect of Neighbourhood

One requirement which has been established for organisation to occur in a SOM is that the neighbourhood size must strictly decrease over time (Flanagan, 1998). This requirement of a strict pattern of decay suggests that learning is not invariant to neighbourhood size. Rather, it suggests that different neighbourhood sizes perform different roles during the learning process. In order to optimise the decay of neighbourhood during learning, it is important to have an understanding of these roles.

The experiments presented in this section attempt to test the role played by different neighbourhood sizes in the learning process. In each experiment, 3 data sets were used: 5×5; 10×10; and 20×20 grids. Each grid was tested on the smallest map that could fit the entire data set and with a 73×73 map.

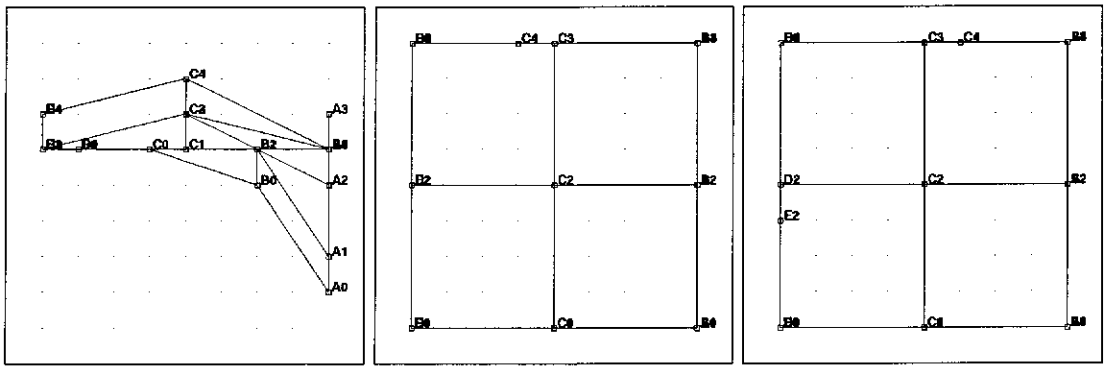
5.2.1 Experimental Results

Data for each experiment in this section was drawn from the Simple Grid data set (Section A.3). These data was trained on maximal Kohonen SOMs. In all cases, the learning gain parameter was set to 0.2. In this way, the role played by neighbourhood size decay can be established independent of learning gain decay.

In these experiments, a visual inspection of the final topologies was made. Using these observations, it is possible to make broad statements regarding the role played by neighbourhood size in the learning process.

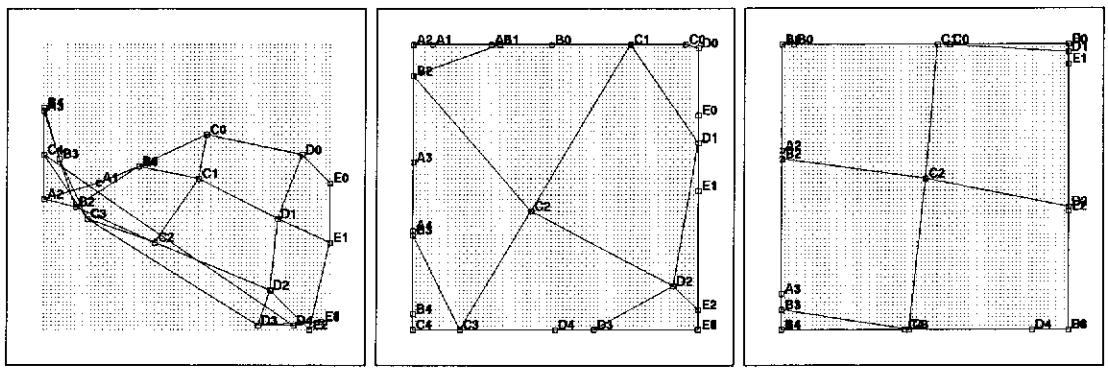
Large Neighbourhood Size

In the first experiment, the effect of a large neighbourhood size on map convergence was tested. In each test, a neighbourhood size equivalent to half the size of the map



(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.1: SOM Convergence using a 5×5 data set, 9×9 map, $\sigma = 5$.

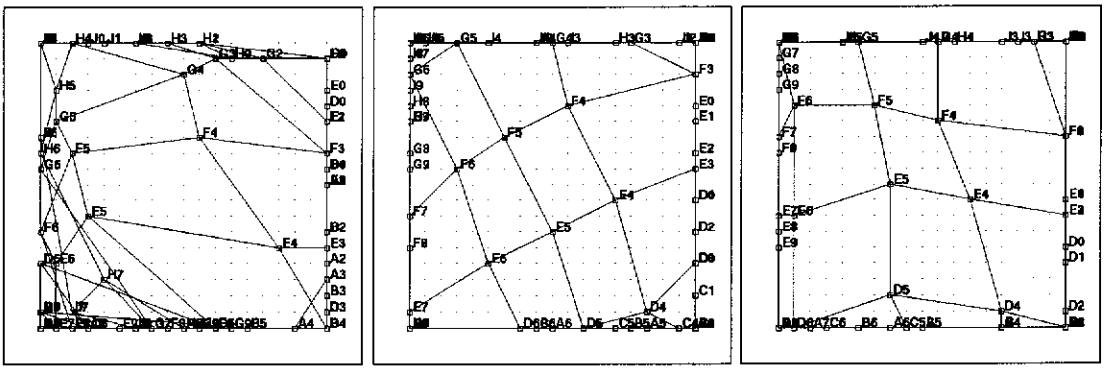


(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.2: SOM Convergence using a 5×5 data set, 73×73 map, $\sigma = 37$.

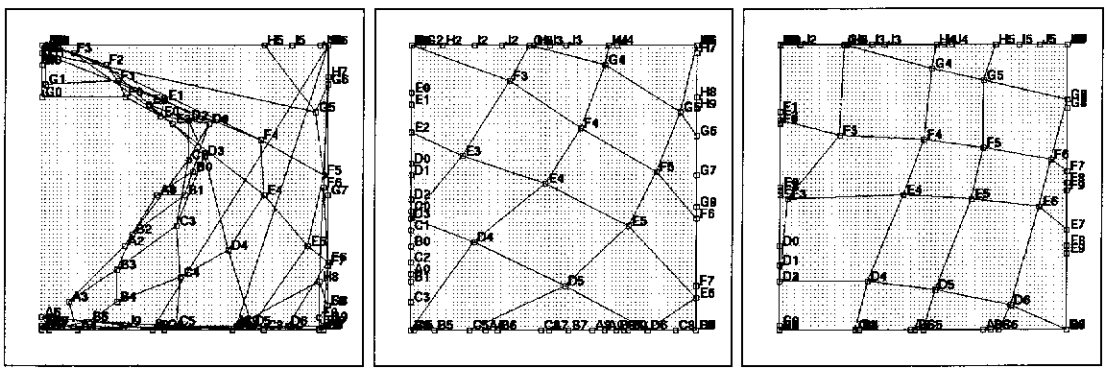
was used. This allows a Gaussian curve centered on one side of the map to have a small, but non-trivial influence on neurons on the opposite side of the map.

The results of this experiment are shown in Figures 5.1–5.6. These results demonstrate analogous behaviour in all cases. Using the large neighbourhood size, data points are rapidly spread to the extent of the map, resulting in an initially good topological mapping. However, the rough topology reached in the first few epochs of training does not improve with further training. In addition, the resolution of this topology is very poor; training vectors are not evenly spread over the resulting map. This is especially evident on the larger data sets, in which the majority of winners are to be found on the extremities of the map.



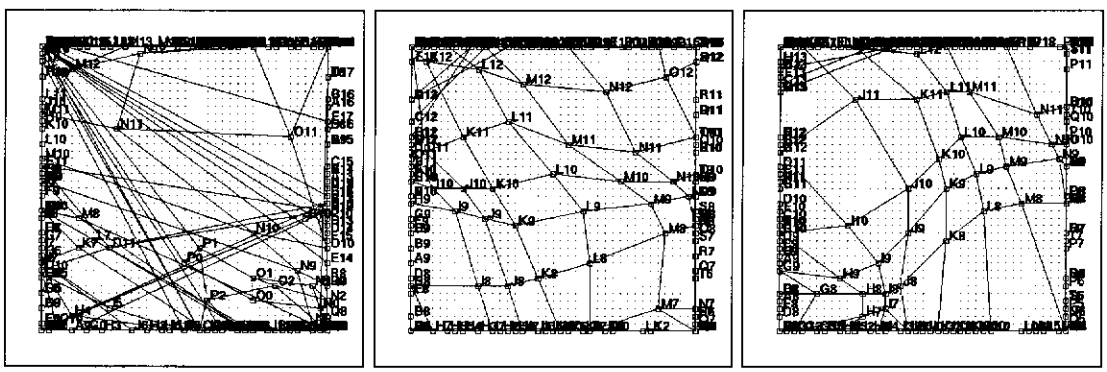
(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.3: SOM Convergence using a 10×10 data set, 19×19 map, $\sigma = 10$.



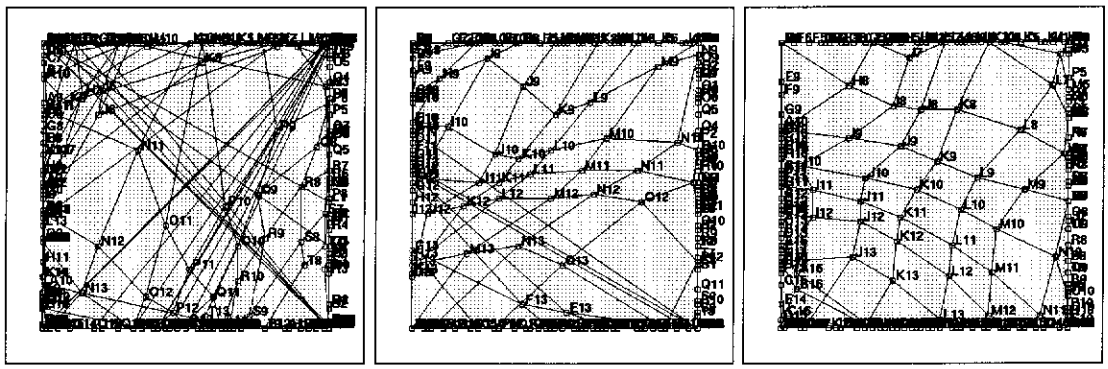
(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.4: SOM Convergence using a 10×10 data set, 73×73 map, $\sigma = 37$.



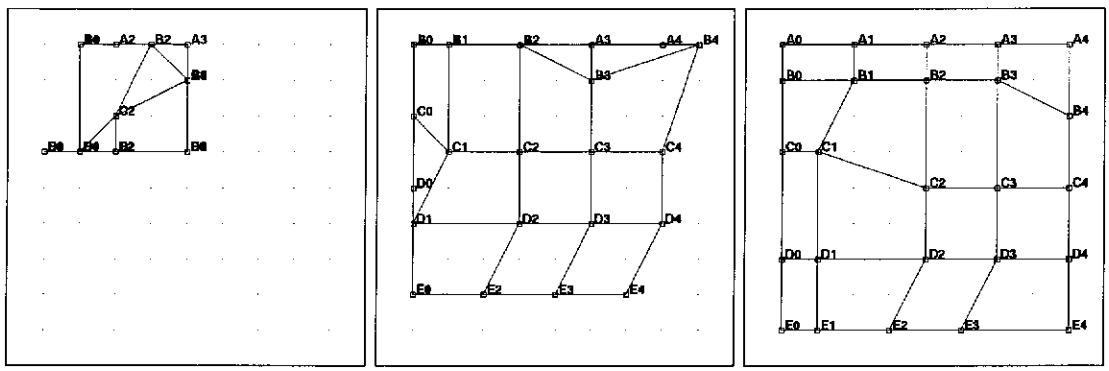
(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.5: SOM Convergence using a 20×20 data set, 39×39 map, $\sigma = 20$.



(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.6: SOM Convergence using a 20×20 data set, 73×73 map, $\sigma = 37$.



(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

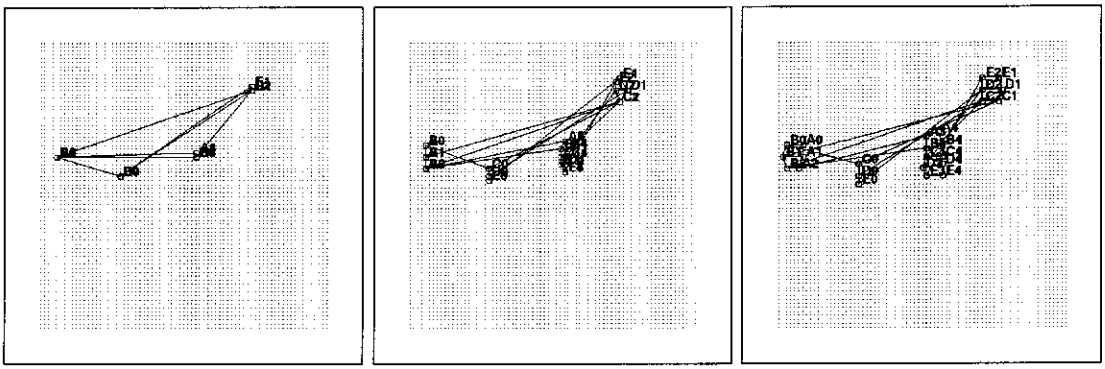
Figure 5.7: SOM Convergence using a 5×5 data set, 9×9 map, $\sigma = 2$.

Small Neighbourhood Size

In this experiment, the effect of a small neighbourhood of interaction on map convergence was tested. In each test, a neighbourhood size of 2 was selected.

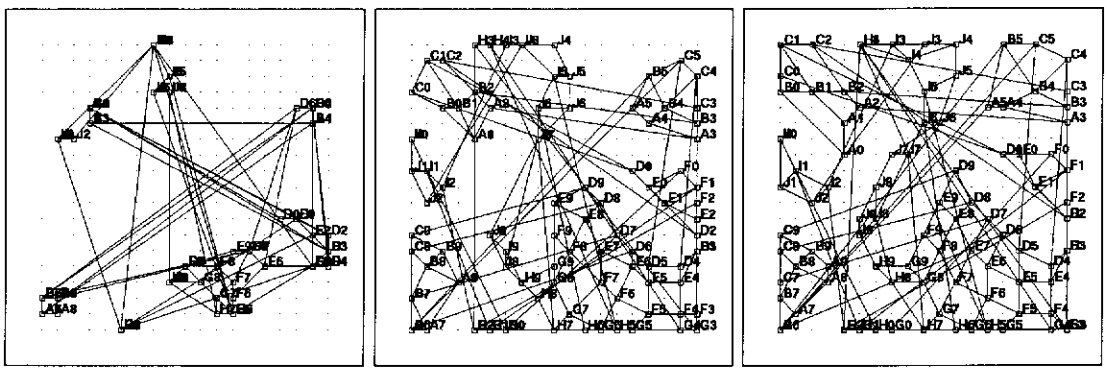
The results of this experiment are shown in Figures 5.7–5.12. From these results it can be seen that, while global topologies do not develop quickly using the small neighbourhood size, local topologies are evident almost immediately. Winning neurons can be easily divided into small groups, each of which exhibits a locally consistent topology. However, the topologies of adjacent groups do not necessarily correlate; the global topology is therefore twisted.

The only exception to this pattern is the smallest data set on the smallest map



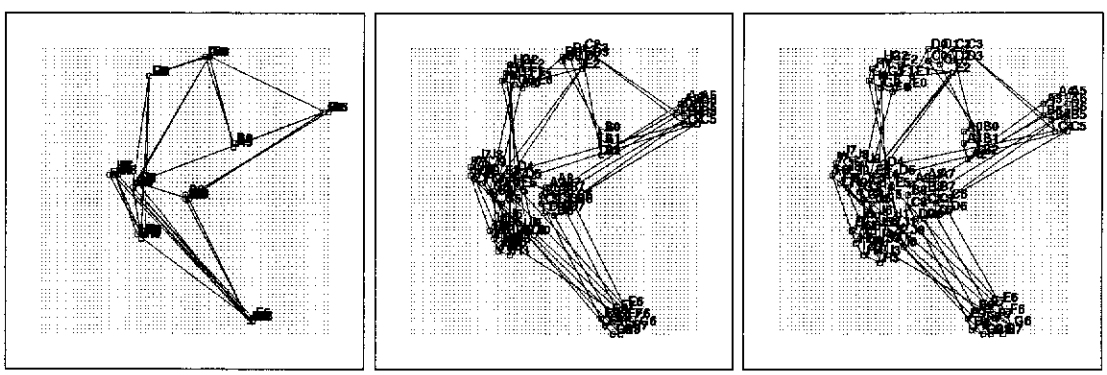
(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.8: SOM Convergence using a 5×5 data set, 73×73 map, $\sigma = 2$.



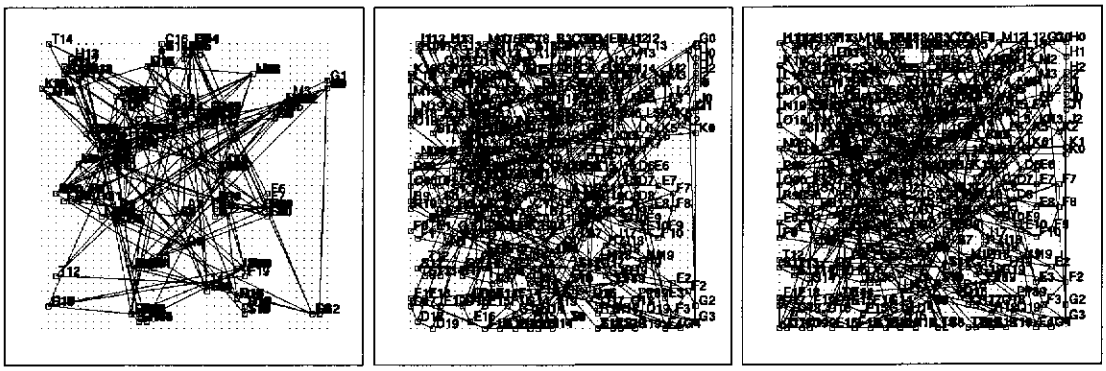
(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.9: SOM Convergence using a 10×10 data set, 19×19 map, $\sigma = 2$.



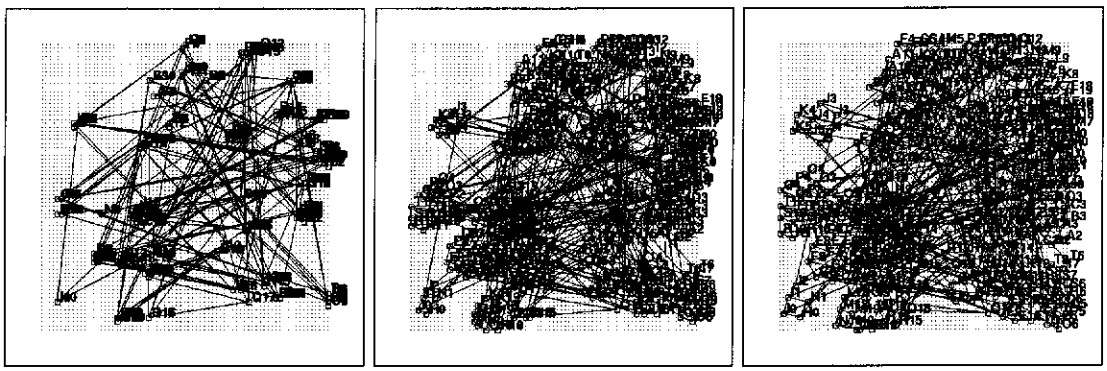
(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.10: SOM Convergence using a 10×10 data set, 73×73 map, $\sigma = 2$.



(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.11: SOM Convergence using a 20×20 data set, 39×39 map, $\sigma = 2$.



(a) After 1 Epoch (b) After 10 Epochs (c) After 20 Epochs

Figure 5.12: SOM Convergence using a 20×20 data set, 73×73 map, $\sigma = 2$.

(Figure 5.7). In this case, a good global topology of the entire data set is observed to form. This is the result of the combination of two factors. Firstly, although localised clustering does occur in this data set, the data set is sufficiently small that only one cluster is able to form — a cluster encompassing all the training data. Secondly, the size of the map is sufficiently small that the 'small' neighbourhood size of 2 is sufficient to impose global effects on the map. The corresponding 'large' neighbourhood size experiment used a neighbourhood size of 5, a value only marginally larger (with respect to the size of the map) than the 'small' value used here.

It is interesting to note that at the completion of each training sequence, winning neurons are, for the most part, 2 neurons apart. This is the same distance used in the neighbourhood update function. This suggests a relationship between the neigh-

bourhood size used for training and the extent of spread in winning neurons. This relationship could be used to ensure adequate spreading of winning neurons in SOMs.

5.2.2 Discussion

The use of large and small neighbourhood sizes during training demonstrates two distinct behaviours, each with unique benefits:

Large Neighbourhood Sizes rapidly generate a consistent global topology but do not generate a precise local topological mapping; and

Small Neighbourhood Sizes allow for the fine tuning of local topologies but are not good at arranging global topologies.

Attempts to quickly generate precise global topologies require a combination of these two characteristics; the rapid spreading of points across the map afforded by a large neighbourhood size, and the fine control afforded by a small neighbourhood size. A learning scheme which utilises a combination of these behaviours should provide near optimal learning behaviour.

This result is supported by the observations made by Von der Malsburg (1973) in his experiments on biologically faithful self-organising map models. Two important qualities are demonstrated by Von der Malsburg's experiments. Firstly, learning was achieved without the need for decaying the learning gain parameter. In most theoretical and experimental studies since, variation in gain has been considered the most significant factor controlling learning. Secondly, global topographical organisation was only observed on small maps; in large map simulations, local topographical organisation occurred, but global topology was erratic.

The cause of this result was theoretically demonstrated by Amari (1980). In this proof, it was demonstrated that a single peak of excitation will result, provided the domain of the lateral connection function spans the entire map. Conversely, a lateral connection function with a smaller range will result in multiple peaks occurring on a map. During Von der Malsburg's experiments, the range of the lateral connection function was only a small number of neurons, and was not decayed. Consequently, on small maps, the connection function spanned the entire map and global organisation was observed. On larger maps, the small connection function did not span the map and many areas of excitation occurred, resulting in erratic global topology.

Another interesting feature of these results is that the primitive topologies develop without the need for a complex learning gain decay mechanism. This seems to suggest that the role played by learning gain is not as significant as previously suggested in the literature. The use of a single constant value for learning gain would significantly simplify the learning process, by removing the need to establish another set of decay parameters.

5.3 Optimizing Learning in Kohonen's SOM

Having established the role played by large and small neighbourhood sizes during the learning process, it is possible to propose a parameter decay scheme which exploits these characteristics to yield improved training times.

In this section, the results obtained in the previous sections are applied to Kohonen's algorithm, by way of a novel method for decaying neighbourhood size during training. This decay scheme provides training times which are equivalent to the best training times possible using traditional decay schemes. However, this technique has the additional benefit of removing the need for extensive tuning of key learning parameters or *a priori* knowledge of the likely training time.

5.3.1 Step Neighbourhood Decay

The previous experiments demonstrate that SOM learning can be broadly characterised by two behaviours: global topological organisation and local topological fine tuning. These two behaviours can be obtained by using a large and small neighbourhood size, respectively. However, both of these behaviours are desirable during the learning process.

It therefore seems reasonable to propose a scheme for decaying neighbourhood size during training which utilises both of these characteristics. Such a scheme would consist of 2 phases:

Phase 1: $\sigma = \lceil \sqrt{2N^2}/2 \rceil$, for T epochs; and

Phase 2: $\sigma = R$, for as many epochs as are required to converge on a solution.

In this formulation, the map undergoing training is an $N \times N$ square map of neurons, and R is the resolving distance of the SOM; that is, the distance between winners that would be observed if every data point was spread evenly over the map.

An additional modification to this simple step may be required. Maximal SOMs are much more sensitive learning algorithms than their minimal counterparts. Unlike the minimal SOM, maximal SOMs have no quantifiable goal state, and consequently, the update rule cannot be directed towards a precise optimal goal.

As a result of this sensitivity, the rapid change in neighbourhood size required by the step decay scheme may cause some difficulty to the learning process, resulting in optimisation towards suboptimal map states. To compensate for this, a smoothed step scheme based upon the Butterworth filter is proposed. This filter is a continuous approximation of the ideal step function, similar to that used in image processing in the continuous approximation of 'Top Hat' filters. An nth order Butterworth step from X to X' at epoch T is given by the equation:

$$\sigma(t) = X - \frac{X - X'}{1 + (\sqrt{2} - 1)((t/T)^{2n})}$$

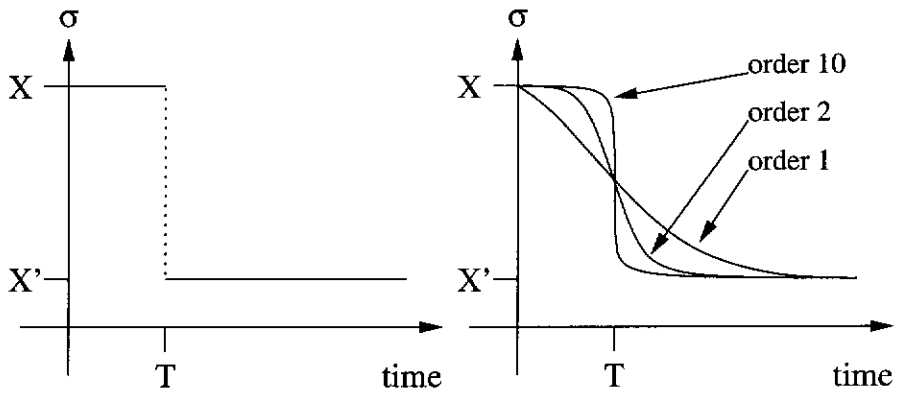
A comparison of parameter decay under the Butterworth scheme and under normal step decay can be found in Figure 5.13. Butterworth filters are used in image processing to remove high frequency 'ringing' effects associated with hard boundaries in image filters. Similarly, the smoothed edge of the Butterworth step decay prevents the rapid development of local topologies, instead encouraging the development of local topologies which are extensions of the global topology established in the early phases of training. This does not detract from the goals of the original step decay scheme; it merely smoothes the transition from large to small neighbourhood size.

In addition to this neighbourhood decay scheme, a scheme for decaying learning gain is required. Based upon the preliminary result obtained in the previous section, a minimalist approach to gain decay will be taken — gain will be held constant at a value of 0.2 throughout experimentation. Experimental verification of this assumption will be presented in the following sections.

5.3.2 Experimental Results

Two experiments were performed to establish the capabilities of the proposed step decay algorithm. In the first experiment, simple grid data sets (see Section A.3) of various sizes were learned by large and small maximal SOMs, in order to establish an appropriate value for T, the length of phase 1 of training in the Butterworth Step decay algorithm.

In the second experiment, simple grid data sets were again used, this time to test



(a) Step decay (b) Butterworth Decay

Figure 5.13: (a) Normal step decay from A to B at epoch T. (b) Butterworth decay from X to X', decay time T. Filters of order 1,2 and 10 are shown; in the infinite limit, Butterworth decay approaches step decay.

the assertion that a sophisticated gain decay mechanism is required when using the step decay algorithm. As with the first experiment, tests were performed on a range of map sizes and grid sizes.

A test set consisting of 500 training points was generated and used to evaluate the $\mathcal{Q}$ metric after each epoch of training. In this way, it was possible to establish the rate at which convergence took place.

Establishing decay period

In this experiment, a maximal SOM consisting of a square grid of neurons is used. The initialisation and update rules for these maps are as described in Section 2.5.1. The Butterworth Step Decay algorithm was used to decay neighbourhood size. In each experiment, learning gain was held at a constant value of 0.2.

Simple grid data sets of size 5×5, 10×10 and 20×20 were trained on these maps. Each data set was trained on a large and a small map. The size of the small map was set as the smallest map which would accommodate the data set (a 9×9 map for the 5×5 data set; a 19×19 map for the 10×10 data set; and a 39×39 map for the 20×20 data set). A 73×73 map was used as the large map for each data set.

For each map-grid combination, six trials were performed. The purpose of these trials was to test a range of values for T, the step period in the Butterworth Step Decay algorithm. T values of 2, 10, 20, 30, 40 and 50 were used. This establishes behaviour

of the Butterworth Step algorithm from the minimum possible period, to a reasonable maximum. Each trial was performed once.

The results of this experiment can be found in Figures 5.14. In each example, the quality of each map rapidly rises to a suboptimal value. As the neighbourhood size is reduced to the lower level, the average $\mathcal{Q}$ metric for the map decreases rapidly, becoming asymptotic at a low value. This value varies with the choice of data set but observation of trained maps showed the values to be representative of well formed maps. Despite the wide range of map and data set sizes utilised in these tests, consistent behaviour is observed in every case.

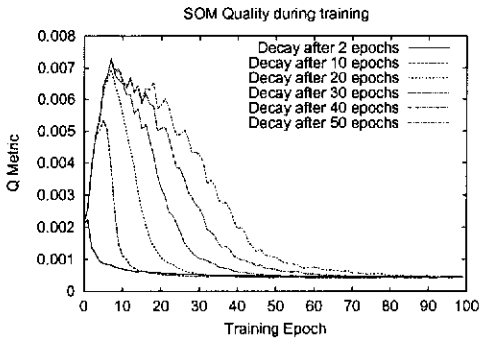
These results suggest that there is no minimum requirement for global topological ordering. The only requirement seems to be that *some* organisation is performed using a large neighbourhood size. However, a decay period of only 2 epochs is sufficient to ensure that global topological ordering takes place. In order to ensure that global ordering does occur, a decay period of 10 epochs will be used and tested in the following experiments.

Establishing gain level

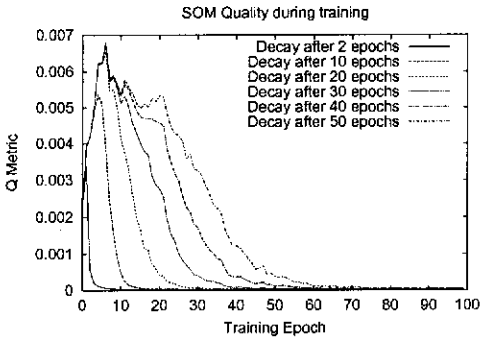
In this experiment, a maximal SOM consisting of a square grid of neurons is used. The initialisation and update rules for these maps are as described in Section 2.5.1. The Butterworth Step decay mechanism was used to decay neighbourhood size, using a decay period of 10 epochs.

As with the decay period experiments, simple grid data sets of size 5×5, 10×10 and 20×20 were trained on these maps. Each data set was trained on a large and a small map. The size of the small map was set as the smallest map which would accommodate the data set (a 9×9 map for the 5×5 data set, a 19×19 map for the 10×10 data set, and a 39×39 map for the 20×20 data set). A 73×73 map was used as the large map for each data set.

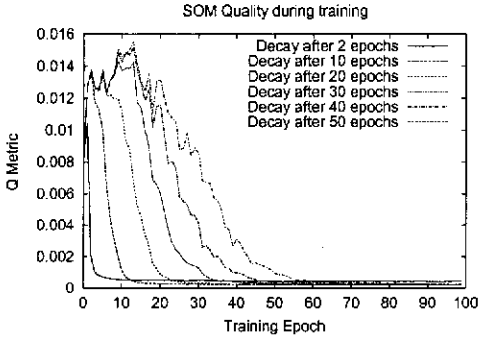
For each map-grid combination, seven trials were performed. The purpose of these trials was to test a range of values for learning gain while decaying neighbourhood size using the Butterworth Step algorithm. Constant gain values of 0.01, 0.1, 0.2, 0.3, 0.4 and 0.5 were used. In addition, a linear gain decay from 1.0 to 0 over a period of 50 epochs was tested. This establishes the behaviour of the SOM algorithm using a range of constant gain values, as well as providing a comparative example of linear gain decay. Each trial was performed once.



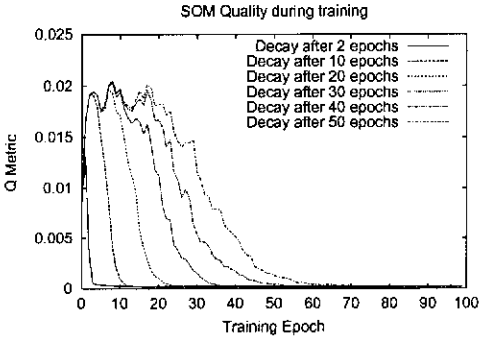
(a) 5 × 5 simple grid, 9 × 9 (small) map. Neighbourhood decay from 6 → 2



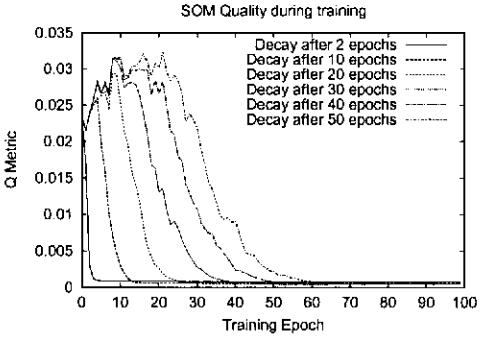
(b) 5 × 5 simple grid, 73 × 73 (large) map. Neighbourhood decay from 51 → 18



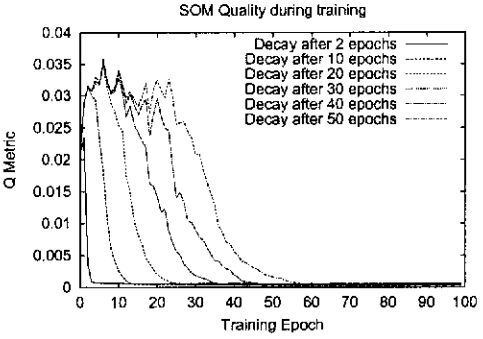
(c) 10 × 10 simple grid, 19 × 19 (small) map. Neighbourhood decay from 13 → 2



(d) 10 × 10 simple grid, 73 × 73 (large) map. Neighbourhood decay from 51 → 8

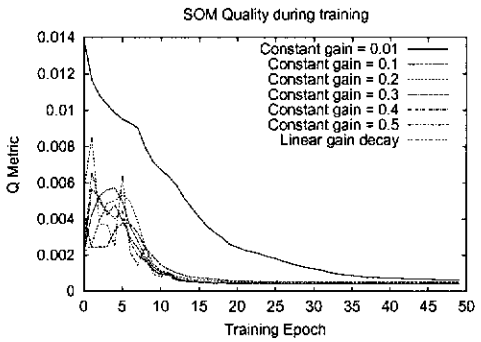


(e) 20 × 20 simple grid, 19 × 19 (small) map. Neighbourhood decay from 27 → 2

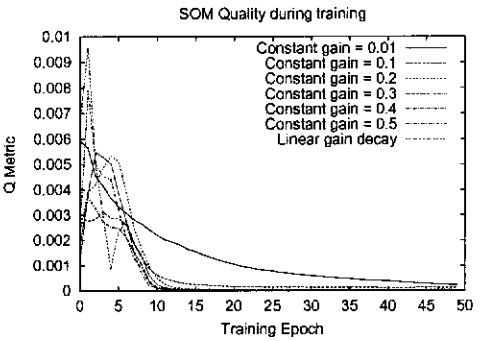


(f) 20 × 20 simple grid, 73 × 73 (large) map. Neighbourhood decay from 51 → 8

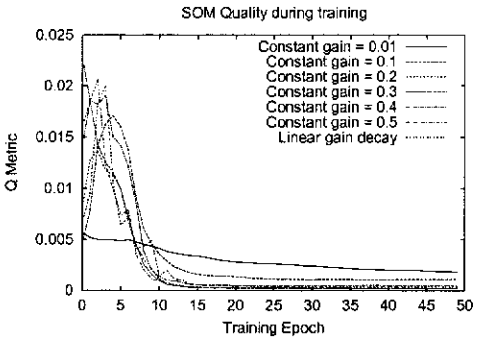
Figure 5.14: Testing the optimal neighbourhood decay period in the Butterworth Step decay algorithm. Each plot shows SOM Quality during the learning of a simple grid, using a range of step sizes.



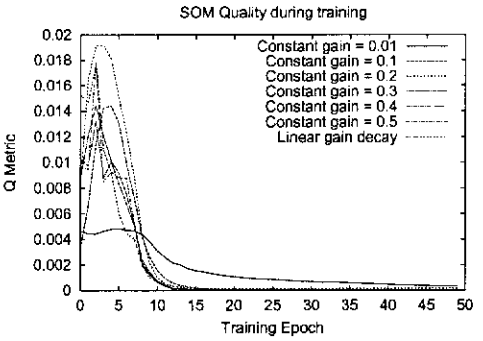
(a) 5 × 5 simple grid, 9 × 9 (small) map. Neighbourhood decay from 6 → 2



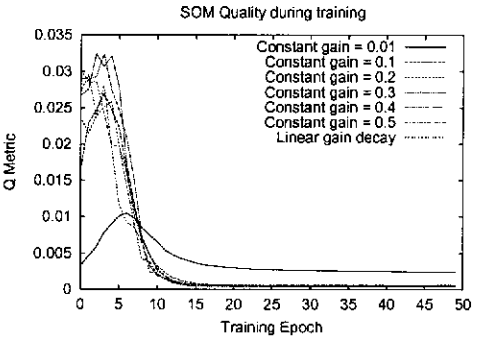
(b) 5 × 5 simple grid, 73 × 73 (large) map. Neighbourhood decay from 51 → 18



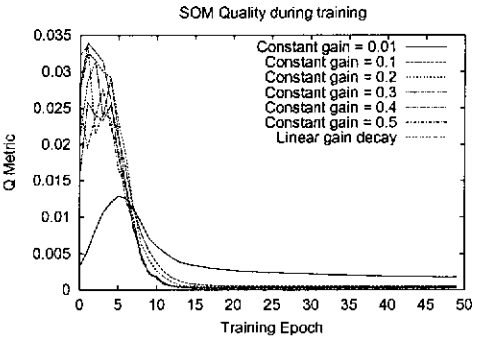
(c) 10 × 10 simple grid, 19 × 19 (small) map. Neighbourhood decay from 13 → 2



(d) 10 × 10 simple grid, 73 × 73 (large) map. Neighbourhood decay from 51 → 8



(e) 20 × 20 simple grid, 19 × 19 (small) map. Neighbourhood decay from 27 → 2



(f) 20 × 20 simple grid, 73 × 73 (large) map. Neighbourhood decay from 51 → 8

Figure 5.15: Testing the optimal gain level for the Butterworth Step decay algorithm. Each plot shows SOM Quality during the learning of a simple grid, using a range of constant gain values, and a linear decay of gain from 1 → 0.

The results of this experiment can be found in Figures 5.15. Similar results are observed in each experiment. The characteristic rapid convergence of the Butterworth Step decay algorithm is largely unaffected by the choice of gain size, or the decision to use a constant, rather than a linearly decaying value of gain. Using a constant gain value, good topologies are achieved quickly.

The only significant feature which seems to affect the choice of gain is that it exceeds a minimum critical value. The only plots which do not follow the consistent behaviour are the plots corresponding to very low constant gain values — values of 0.01 and 0.1. When these low constant values are utilised, the SOM is not reinforced sufficiently after each training epoch. Consequently, during the first 10 epochs, the map does not develop a strong global topology, and the resulting map is a poor global topological organisation of the training data.

These results suggest that the choice of gain value is largely insignificant in controlling the organisation process. These experiments produced identical results with a wide range of gain values. In addition, no gain decay scheme was necessary to stimulate the organisation process. This challenges the notion that gain decay is the strongest controlling factor on the organisation process.

Following the results presented in this section, the self-organisation experiments performed in this thesis will utilise a constant gain value of 0.2, coupled with a neighbourhood decayed using the Butterworth Step method.

5.4 Comparison with Existing Schemes

Having established the values required for the key parameters, it is now possible to compare performance of this algorithm to existing parameter decay algorithms. The experiments presented in this section compare the performance of the step decay technique with conventional decay mechanisms. Three different data sets and five different decay schemes (including simple step decay and Butterworth Step Decay) are used to form this comparison. In these experiments, the Q metric was evaluated throughout training in order to establish the rate at which convergence took place.

5.4.1 Simple Grid

In this experiment, a maximal SOM consisting of a square grid of 9×9 neurons is used. A simple grid of size 5×5 was then used to train the map. The initialisation and update

rules for these maps are as described in Section 2.5.1. Five different neighbourhood decay schemes were tested:

1. $\alpha(t) = 1.0 - \frac{0.9}{300}t$, $\sigma(t) = 9.0 - \frac{9.0}{300}t$;
2. $\alpha(t) = 1.0 - \frac{0.9}{100}t$, $\sigma(t) = 9.0 - \frac{9.0}{100}t$;
3. $\alpha(t) = 0.2$, $\sigma(t) = 9.0 - \frac{9.0}{100}t$;
4. $\alpha(t) = 0.2$, $\sigma(t) = 6$, $t < 10$, otherwise $\sigma(t) = 2$; and
5. $\alpha(t) = 0.2$, $\sigma(t) = 6 - \frac{4}{1+(\sqrt{2}-1)(t/10)^4}$.

Schemes 1 and 2 are classic Kohonen linear decay schemes, over a period of 300 and 100 epochs respectively. Scheme 3 uses a classic linear decay for σ over 100 epochs, but keeps α at a constant value of 0.2 throughout training. Scheme 4 is a simple step decay scheme, and Scheme 5 is an example of Butterworth Step Decay. Following the results of the Section 5.3, a step decay period (T) of 10 epochs is used in schemes 4 and 5. The step decay schemes use a large neighbourhood size of 6 and a small neighbourhood size of 2. A constant gain of 0.2 is used in schemes 3–5.

Each neighbourhood decay scheme was tested 10 times, and the results of these trials were averaged. Uncertainty in each mean value was calculated using a single value two tailed Student's t-test on a 95% confidence interval. The test set used to evaluate $\mathcal{Q}$ metric values consisted of 500 vectors.

A table of the asymptotic value reached by each scheme can be found in Table 5.1. The decay in $\mathcal{Q}$ metric value for each scheme is shown graphically in Figure 5.16. These results demonstrate that all five schemes eventually reach the same asymptotic value (allowing for experimental uncertainty). This value represents a well formed topological representation of a grid. Scheme 1 reach this value after $\approx$ 250 epochs, Schemes 2 and 3 after $\approx$ 75 epochs, and Schemes 4 and 5, the step decay, after $\approx$ 20 epochs.

Upon first inspection, these results would seem to suggest that the step decay algorithm, in either form, yields superior training times to the linear decay mechanisms. However, this is not necessarily the case. The factor restricting the convergence time of the linear decay examples is the selection of an appropriate decay time. A linear decay scheme may be capable of matching the performance of the step algorithm by selecting a linear decay period of 20 epochs. However, linear (and many other decay schemes) require *a priori* knowledge of the time required for decay or an appropriate rate of decay. This knowledge is not available for most problems without empirical

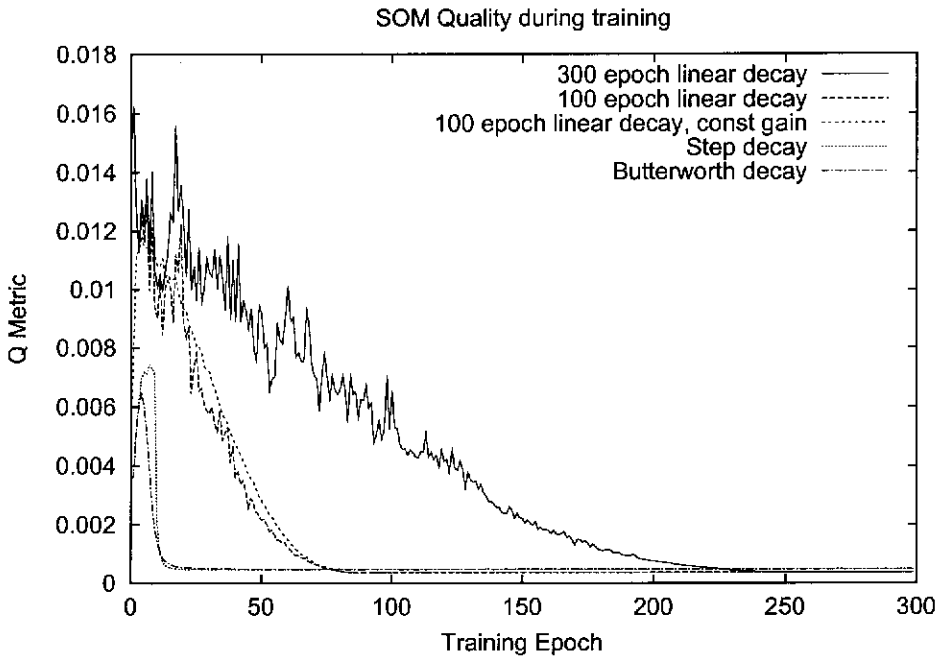


Figure 5.16: SOM Quality during learning of a 5×5 simple grid on a 9×9 map. Each curve represents a different decay scheme.

study. The greatest advantage of the step algorithm is that it requires no such *a priori* knowledge, only a knowledge of the size of the map, and the probable density of data points on that map. Given this knowledge, which is available prior to training, without empirical testing, it is possible to use the step decay technique to attain training times comparable to the best possible using linear decay.

Another feature of note is the fact that Schemes 2 and 3 have almost identical convergence times, and converge to a statistically identical asymptotic value. In addition, Scheme 3 is observed to demonstrate much less erratic behaviour during convergence. This suggests that the role played by decaying learning gain during training is much less significant than the literature suggests. The erratic convergence of Scheme 2 could even lead to the conclusion that decaying α *impedes* learning by preventing stable convergence on an optimal solution.

5.4.2 Questionnaire

In this experiment, a maximal SOM consisting of a square grid of 15×15 neurons is used. The initialisation and update rules for these maps are as described in Section 2.5.1. A questionnaire data set (see Section A.4) consisting of 4 questions was used as the training data. The $\mathcal{Q}$ metric was evaluated using the same data set.

Scheme	Asymptotic $\mathcal{Q}$ metricvalue ($\times 10^{-4}$)
Linear decay, 300 epochs	3.6 ± 0.2
Linear decay, 100 epochs	3.7 ± 0.2
Linear decay, 100 epochs, constant gain	3.6 ± 0.2
Simple step, constant gain	4.1 ± 0.3
Butterworth step, constant gain	4.1 ± 0.3

Table 5.1: Asymptotic $\mathcal{Q}$ metric values at the end of training a 5×5 simple grid on a 9×9 map, using a variety of neighbourhood decay schemes.

Scheme	Asymptotic $\mathcal{Q}$ metricvalue ($\times 10^{-}3$)
Linear decay, 300 epochs	2.2 ± 0.3
Linear decay, 100 epochs	2.2 ± 0.3
Linear decay, 100 epochs, constant gain	1.9 ± 0.4
Simple step, constant gain	1.9 ± 0.2
Butterworth step, constant gain	2.1 ± 0.2

Table 5.2: Asymptotic $\mathcal{Q}$ metric values at the end of training a 4 question questionnaire on a 15×15 map, using a variety of neighbourhood decay schemes.

Five different neighbourhood decay schemes were tested:

1. $\alpha(t) = 1.0 - \frac{0.9}{300}t$, $\sigma(t) = 15.0 - \frac{15.0}{300}t$;
2. $\alpha(t) = 1.0 - \frac{0.9}{100}t$, $\sigma(t) = 15.0 - \frac{15.0}{100}t$;
3. $\alpha(t) = 0.2$, $\sigma(t) = 15.0 - \frac{15.0}{100}t$;
4. $\alpha(t) = 0.2$, $\sigma(t) = 11$, $t < 10$, otherwise $\sigma(t) = 2$; and
5. $\alpha(t) = 0.2$, $\sigma(t) = 11 - \frac{9}{1+(\sqrt{2}-1)(t/10)^4}$.

These decay schemes are analogous to the schemes used in the grid experiment. Schemes 1 and 2 are classic Kohonen linear decay schemes, over a period of 300 and 100 epochs respectively. Scheme 3 uses a classic linear decay for σ over 100 epochs, but keeps α at a constant value of 0.2 throughout training. Scheme 4 is a simple step decay scheme, and Scheme 5 is an example of Butterworth Step Decay. Following the results of the Section 5.3, a step decay period (T) of 10 epochs is used in schemes 4 and 5. The step decay schemes use a large neighbourhood size of 11 and a small neighbourhood size of 2. A constant gain of 0.2 is used in schemes 3–5.

Figure 5.17 shows the results of training using the questionnaire data set with the 5 neighbourhood decay schemes. Table 5.2 shows the asymptotic $\mathcal{Q}$ metric values

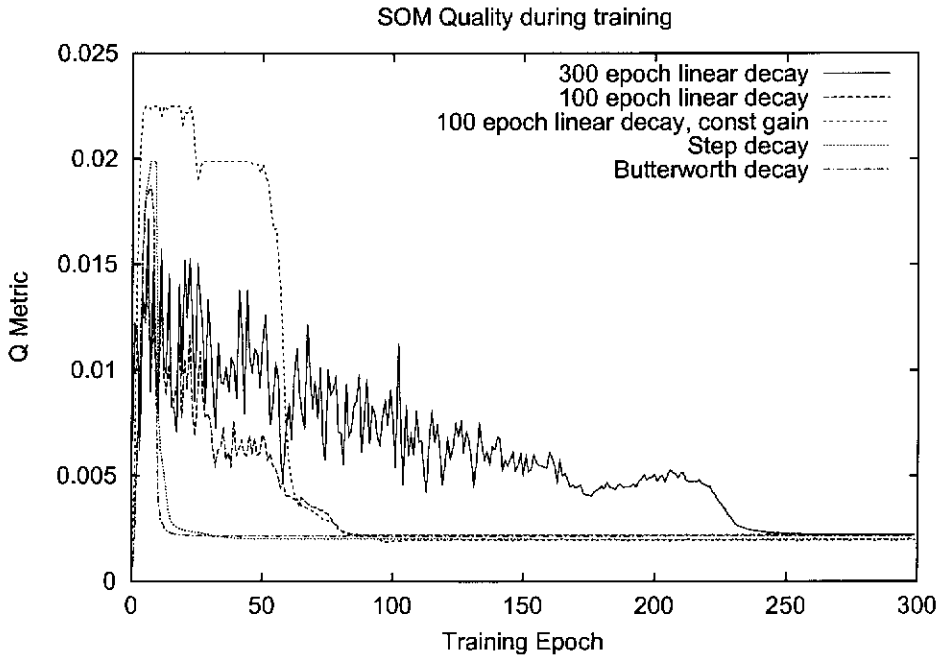
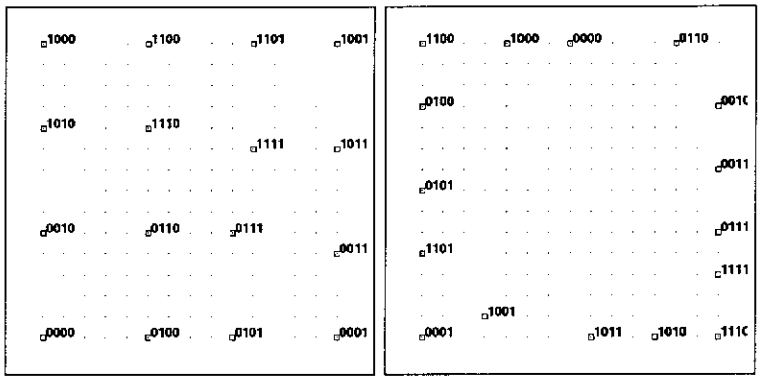


Figure 5.17: SOM Quality during learning of a 4 question questionnaire on a 15×15 map. Each curve represents a different decay scheme.

achieved by each scheme. The results obtained in this experiment are analogous to those obtained in the Grid experiment. In each trial, the map converges to an asymptotic $\mathcal{Q}$ metric value. The same $\mathcal{Q}$ metric value is obtained in each trial. The only significant difference between each trial is the time taken to reach this asymptotic value. Using linear decay schemes, the time taken to converge is constrained by the decay parameter. As a result, training takes ≈ 250 epochs in the first trial, and ≈ 80 epochs in the second and third trials. However, the step decay schemes reach an asymptotic representation after only 20 epochs, without the need to specify an appropriate decay period.

An interesting feature of these results is the range in topologies which are generated by the decay scheme. The 4-question questionnaire data set is a 4 dimensional data set (with each dimension represented using two inputs). However, in presenting this data on a 2D SOM surface, 2D of the 4D data space must be lost. If the preserved topologies correspond exactly with the obvious input topology, there are ${}^4C_2 = 6$ possible ways to organise this 4D data. Figure 5.18(a) is an example of one of these simple topological preservations. In this example, the answers for the first and last questions form the principal topology — the response to question 1 on the vertical axis, and question 4 on the horizontal axis. A secondary topology can be observed in the second and third questions. However, this topology is localised to regions within the principal topology.



(a) Mapping 1 (b) Mapping 2

Figure 5.18: Two example mappings of a 4 question questionnaire on a 15 × 15 map. These two mappings have almost identical $\mathcal{Q}$ metric values, even though Mapping 1 is the more obvious and useful representation.

This is not the only way which the SOM algorithm can organise this data set. The 2 dimensions which must be lost during organisation need not be orthogonal to the 4 obvious dimensions of the data space. Rather than completely losing 2 dimensions and completely preserving another 2, the SOM can organise into a state which preserves half of every dimension, losing the other half. This results in an output state whose topology is not visually obvious; however, the $\mathcal{Q}$ metric value resulting from such an organisation is virtually identical to that produced by the more obvious topology. Figure 5.18(b) is an example of such an eccentric topology.

5.4.3 Scotch Whisky

In this experiment, a maximal SOM consisting of a square grid of 25 × 25 neurons is used. The initialisation and update rules for these maps are as described in Section 2.5.1. The Scotch Whisky data set described in Section A.6 was used as a training set. The $\mathcal{Q}$ metric was evaluated using the same data set.

Five different neighbourhood decay schemes were tested:

1. $\alpha(t) = 1.0 - \frac{0.9}{300}t$, $\sigma(t) = 25.0 - \frac{25.0}{300}t$;
2. $\alpha(t) = 1.0 - \frac{0.9}{100}t$, $\sigma(t) = 25.0 - \frac{25.0}{100}t$;
3. $\alpha(t) = 0.2$, $\sigma(t) = 25.0 - \frac{25.0}{100}t$;
4. $\alpha(t) = 0.2$, $\sigma(t) = 18$, $t < 10$, otherwise $\sigma(t) = 2$; and

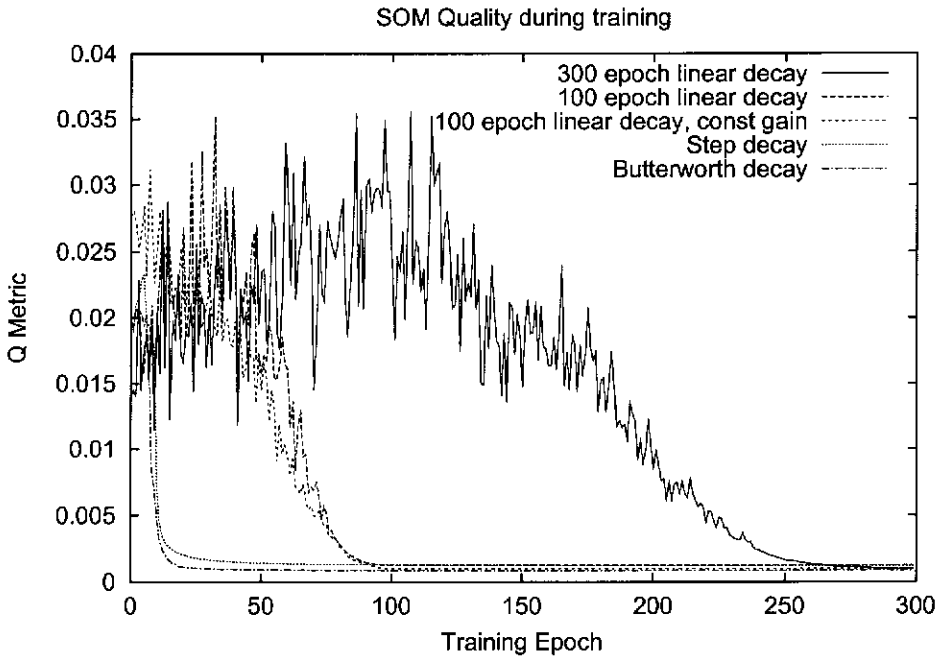


Figure 5.19: SOM Quality during learning of the Scotch Whisky data set on a 25×25 map. Each curve represents a different decay scheme.

Scheme	Asymptotic $\mathcal{Q}$ metricvalue ($\times 10^{-3}$)
Linear decay, 300 epochs	0.9 ± 0.3
Linear decay, 100 epochs	1.3 ± 0.3
Linear decay, 100 epochs, constant gain	1.0 ± 0.5
Simple step, constant gain	1.2 ± 0.3
Butterworth step, constant gain	0.8 ± 0.2

Table 5.3: Asymptotic $\mathcal{Q}$ metric values at the end of training the Scotch Whisky data set on a 25×25 map, using a variety of neighbourhood decay schemes.

5. $\alpha(t) = 0.2$, $\sigma(t) = 18 - \frac{16}{1+(\sqrt{2}-1)(t/10)^4}$.

These decay schemes are analogous to the schemes used in the Simple Grid experiment. Schemes 1 and 2 are classic Kohonen linear decay schemes, over a period of 300 and 100 epochs respectively. Scheme 3 uses a classic linear decay for σ over 100 epochs but keeps α at a constant value of 0.2 throughout training. Scheme 4 is a simple step decay scheme and Scheme 5 is an example of Butterworth Step Decay. Following the results of the Section 5.3, a step decay period (T) of 10 epochs, and a constant gain of 0.2 is used for schemes 4 and 5. The step decay schemes use a large neighbourhood size of 18 and a small neighbourhood size of 2.

Figure 5.19 shows the results of training using the Scotch Whisky data set with the

five neighbourhood decay schemes. This real-world example confirms the results of the previous experiments. Again, all neighbourhood decay schemes are able to bring the SOM to a good topological representation of the data but the step decay mechanisms are able to do so without the need for *a priori* knowledge of the time required to train the data set. A table of the asymptotic $\mathcal{Q}$ metric values obtained in this experiment can be found in Table 5.3.

As with the questionnaire data set, the dimensionality of the Scotch Whisky data vastly exceeds the 2 dimensions available on the SOM. Consequently, many dimensions of data are not represented in the final topology. However, unlike the questionnaire data set, there is not obvious organisation of the Scotch Whisky data. A wide range of equally valid 2D topological organisations of this data set are observed; however, there is currently no method for forcing a given topology to express in preference to another.

5.5 Discussion

These experiments clearly demonstrate the principal benefit of the Butterworth Step decay scheme. Training times using the technique are comparable with the best possible training times using traditional decay techniques. However, these training times can be achieved using parameters that can be easily derived from the data set and map to be used. No *a priori* knowledge or empirical studies of likely training time are required to establish the key parameters of the step decay scheme.

Perhaps the strongest evidence supporting use of the step decay mechanism comes through observing the goodness of fit curves for the 300 epoch linear decay in Figures 5.16, 5.17 and 5.19. The behaviour of this curve can be classified into two types; erratic oscillation (during epochs 1-150) and stable decay (during epochs 150-300). The switch between states in both cases occurs at the point where neighbourhood size decays to a value equivalent to the resolving power of the map. Once the neighbourhood size reaches this point, a stable topographical representation is rapidly achieved rapidly. The step mechanism represents the distillate of this process. Rather than engage in a long period of erratic representations, the step decay mechanism uses a large neighbourhood size for the minimum possible time — just long enough to establish a primitive global topology. Neighbourhood size is then dropped to R (the desired resolution of the data set), to allow precise local ordering.

It is also interesting to note that this fast training time was achieved *without* the need for a sophisticated scheme of learning gain decay. This represents a significant

difference from the perspective presented in the vast majority of literature (Kohonen, 2001) (Cottrell *et al.*, 1998) (Mulier and Cherkassky, 1994). These sources emphasize the importance of learning gain, leaving neighbourhood size as a peripheral concern. This study suggests that neighbourhood size is at least an equal, if not more important contributor to the process of learning in self-organisation.

Why is gain seemingly unimportant in these experiments? As an *extremely* simplistic analysis, this can be attributed to the fact that a learning step with $\alpha = 0.4$ is essentially equivalent to 2 learning steps at $\alpha = 0.2$ (however, as $\alpha \to 1$, this relation becomes increasingly tenuous). Allowing initially large gain permits rapid convergence from random weights to a reasonable topology. However, there is no guarantee that if a good topology will be retained if found. This is observed in the wildly ranging $\mathcal{Q}$ metric values in the initial epochs of Figures 5.16, 5.17 and 5.19. During these early epochs a good topology acquired in one epoch is often lost in the next. By using a smaller learning gain, it may take slightly longer to attain a global topological representation but oscillations about this representation will be minimal. The net difference in time taken to reach a stable representation is therefore minimal. Sophisticated schemes of learning rate decay would seem to be unnecessary.

One possible benefit of an initially large neighbourhood size is to provide momentum to escape local minima representations of a high dimensionality input space. To avoid this eventuality, the step decay method could be combined with an optimal learning rate decay scheme (such as those in Chapter 3.6 of Kohonen (2001)). However, the experiments performed here seem to suggest that this is not required to achieve good training times for many data sets.

The proposed neighbourhood size decay scheme has one additional benefit: the potential for use in a continuous learning framework. Using conventional learning decay schemes, learning gain and neighbourhood size are, in most cases, decayed to 0 at the conclusion of training. Under these conditions, the SOM has reached a steady state and further learning cannot take place. The SOM is then used in an 'off-line' manner, identifying winners without updating the map weights. However, using the proposed training scheme, there is no limit on the time for which training can take place. Once the map has arrived at an asymptotic representation with the small neighbourhood size, the map can be used to identify winners within a precise global topology. However, by retaining the ability to update, the SOM will continue to respond to changes in input data, becoming a dynamic model of the input data set. While this facility could be

built into conventional parameter decay schemes, it would require *a priori* evaluation of yet more parameters. This issue will be addressed in Chapter 6.

The partial preservation of complex topologies is also a matter of some interest. The results presented in Sections 5.4.2 and 5.4.3 show that when a complex (i.e., > 2D) topology exists in the training data, only 2D of the complete topology will be represented in the SOM. The loss of topology can occur either as the complete loss of certain topological dimensions or as a partial loss over all dimensions. However, these two modes of topological loss generate equivalent asymptotic $\mathcal{Q}$ metric values.

This can be attributed to the fact that the $\mathcal{Q}$ metric method assumes that all dimensions in a data set are equally significant. This is not necessarily the case. *A priori* knowledge about the data set could indicate that specific dimensions of the data set are of greater importance than others. If this is the case, it would desirable to be able to force the SOM into a given principal topology. However, the traditional SOM algorithm provides no method for preferentially selecting dimensions for preservation — all dimensions are treated equally during training. This issue will be addressed in Chapter 7.

5.6 Conclusion

In this chapter, the role played by large and small neighbourhood sizes was established using an empirical study. It was found that large neighbourhood sizes are responsible for establishing global map topology, whereas small neighbourhood sizes are responsible for developing local map topologies. A combination of large and small neighbourhood sizes is required during the training process to establish a good global topology with local precision.

Based upon these findings, a scheme for decaying parameters during SOM training was proposed. This scheme, known as Butterworth Step Decay, involves a smooth step between a large and small neighbourhood size. This decay scheme provides training times comparable to the best training times possible using traditional linear decay schemes but precludes the need for *a priori* knowledge of likely training times.

In addition, the use of this scheme removes the need for sophisticated learning gain decay schemes. Butterworth step decay utilises a constant value for gain. Empirical evidence presented in this chapter demonstrates that the performance of the Butterworth scheme is largely unaffected by the choice of gain. This demonstrates that learning gain is not the most significant attribute contributing to the learning of a SOM topology.

An additional benefit of the proposed parameter decay scheme is its potential for use in continuous learning situations. As a result of the fact that the Butterworth Decay scheme maintains nonzero learning parameters, a map trained using the scheme can remain receptive to changes in training data over time. This feature is not available to conventional parameter decay schemes which decay all learning parameters to zero at the end of training. This feature will be explored in the next chapter.

Chapter 6

Continuous Learning

> He [Arthur] knew that one of the things he was supposed to do as a parent was to show trust in his child, to build a sense of trust and confidence into the bedrock of relationship between them. He had a nasty feeling that that might be an idiotic thing to do, but he did it anyway, and sure enough it had turned out to be an idiotic thing to do. You live and learn. At any rate, you live.
>
> You also panic.
>
> — *Mostly Harmless* (Adams, 1992)

6.1 Introduction

Most conventional algorithms maintain a two stage approach to the learning process; an initial phase of learning, followed by a prolonged period of off-line use, in which no new information is learned. This separation of the learning process into two stages enforces a hard endpoint to the period of training. Kohonen's SOM is one example of a learning algorithm which follows such a two-stage approach to training. This two stage approach is enforced through the selection of parameter decay schemes. Traditional parameter decay schemes for Kohonen's SOM involve decaying SOM parameters to zero at the end of the training phase. Once parameters have decayed to this level, the SOM remains static and cannot adapt to new training data.

However, in real-world learning problems, it is difficult to perform this division. Learning systems do not usually have access to a complete and canonical set of training examples prior to the commencement of training. Rather, learning is a continuous process with input stimuli that are highly non-stationary (i.e., the probability distribution of training cases varies over time). This necessitates the development of algorithms which can learn continuously.

In this chapter, a continuous learning extension to Kohonen's SOM algorithm is presented. This extension is achieved by exploiting the fact that the Butterworth Step

Decay scheme presented in the previous chapter does not decay learning parameters to zero. As a result, the SOM retains the ability to adapt to changes in the training set indefinitely. This avoids the need to divide the life of the SOM into discrete phases of training and use.

This extension is tested empirically using the Simple Grid, Questionnaire, and Scotch Whisky data sets. These data sets are allowed to 'drift' over time, introducing new training examples from outside the original domain of training. Performance of the continuously learning SOM is compared to traditional parameter decay schemes to show the manner in which new training data can be integrated into the SOM representation.

6.2 Continuous Learning in Kohonen's SOM

Using conventional learning decay schemes within Kohonen's algorithm, parameters learning gain and neighbourhood size are decayed to 0 at the conclusion of training. Under these conditions, the SOM reaches a steady state at the end of training, and further learning cannot take place. The SOM is then used in an 'off-line' manner, identifying winners without updating the map weights.

This approach requires that the user has an extensive *a priori* knowledge of the characteristics of the data set in order to set appropriate decay parameters. In a continuous learning situation, it is hard or impossible to obtain this information, as changes in the characteristics of the data set over time cannot be predicted. Alternatively, the user must expend significant computational effort to use dynamic filters (such as Kalman filters) to update learning parameters.

One technique that can circumvent this problem is the use of Butterworth Step Decay, introduced in Chapter 5. Under this scheme, SOM learning is divided into two phases. In the short initial phase ($\approx$ 10 epochs), a large neighbourhood size (approximately half the size of the map) is used. The map is trained using this large neighbourhood size to establish an initial global map topology. After this short initial period of training, a prolonged period of learning with a small neighbourhood size takes place. The neighbourhood size selected for this phase of training is dependent on the expected density of the data set on the trained map. This smaller neighbourhood size is used for training until an adequate topological organisation has developed. The transition between the large and small neighbourhood sizes is moderated by a second order Butterworth approximation of the hard step. This prevents the introduction of false local topologies by encouraging the local topologies suggested by the global

topology. Learning gain is held constant throughout training; Chapter 5 found a value of gain value of 0.2 to be suitable.

Using this parameter decay scheme, there is no limit to the time over which training can take place. Once the map has arrived at an asymptotic representation with the small neighbourhood size, the map can be used to identify winners within a precise global topology. Consequently, the map reaches a stable representation when it is ready, rather than being restricted by an artificially imposed parameter decay rate.

This characteristic can be exploited to achieve continuous learning on a SOM. By retaining the ability to update, the SOM remains capable of responding to changes in input data. When using a static data set, the SOM will eventually reach a stable representation and, although the weights are capable of updating, they tend not to change by any significant amount. However, if we allow the training set to change over time, the SOM becomes a dynamic model of the input data set at any given time — that is, the SOM can learn continuously. While this facility could be built into conventional parameter decay schemes, it would require *a priori* evaluation of yet more parameters.

One advantage of the use of SOMs for continuous learning is the control of over-specification. Overspecification is a key problem with many continuous learning architectures (see Section 2.7 for a full discussion). However, SOMs are an unsupervised learning mechanism — that is, they do not learn an output classification state, only an internal representation. The specificity of this internal representation is limited by the resolution (i.e., the number of neurons) of the self-organising neural layer. All training data presented to the map is fitted to the existing elements on this neural layer. Consequently, the user has complete control over specificity through the initial selection of map size.

6.3 Testing Continuous Learning

In this section, an experiment is presented which demonstrates the way in which the Butterworth Step decay mechanism can be used to implement a scheme of continuous learning. This experiment is repeated three times, using three different training sets: the Simple Grid data set (see Section A.3), the Questionnaire data set (see Section A.4), and the Scotch Whisky data set (see Section A.6).

In order to demonstrate the continuous learning capabilities of the Butterworth Step Decay mechanism, each data set is split into two overlapping subsets. The first subset is

presented to the SOM, and learned completely. At the completion of this initial phase of learning, the second subset is presented to the SOM. This new data set contains some familiar training examples and some new examples from the same domain. A continuously learning system should incorporate the new training examples into the topology established in the initial training phase. For the purposes of comparison, a second SOM is trained using the same training data regimen. However, this map uses traditional Kohonen linear decay to 0 of neighbourhood size. This shows the effect of a changing data source on a traditional Kohonen SOM, and provides a basis for comparison with the continuous learning technique.

The performance of the continuously learning SOM is quantified using the $\mathcal{Q}$ metric defined in Chapter 3. The evaluation of the $\mathcal{Q}$ metric value is performed with respect to a specific data set. By using test data drawn from a specific data set (or subset) it is possible to use the $\mathcal{Q}$ metric to evaluate the representation of each of the overlapping data subsets, and the combined data set.

6.3.1 Simple Grid

In the first experiment, a Simple Grid data set (see Section A.3) is used. Change in the source data is achieved by generating a rectangular grid of inputs (instead of the normal square) and selecting two overlapping square subsets of this rectangular data set.

To simulate a shift of n units in a Simple Grid data set, a $7 \times 7 + n$ rectangular Simple Grid data set was generated. The two 7×7 subsets were drawn from this rectangular data set as training subsets. Using these two subsets of the original data set it is possible to test the response to new data in the data source (the columns of data in subset 2 which are not in subset 1), and the response to data disappearing from the data source (the columns of data in subset 1 which are not in subset 2).

This process of data set generation is demonstrated graphically in Figure 6.1. Data subsets representing shifts of size 1, 4 and 8 were generated for this experiment. This represents a range of shifts from a minor shift, to a complete change in the data source.

These data subsets were trained on a 13×13 neuron map. The first period of training, using data subset 1, lasted for 75 epochs. After this initial period of training, data subset 2 was used for a further 75 epochs of training.

When using Butterworth neighbourhood decay, neighbourhood size is decayed from 9 to 2, with the step occurring after 10 epochs. The final step value of 2 is maintained

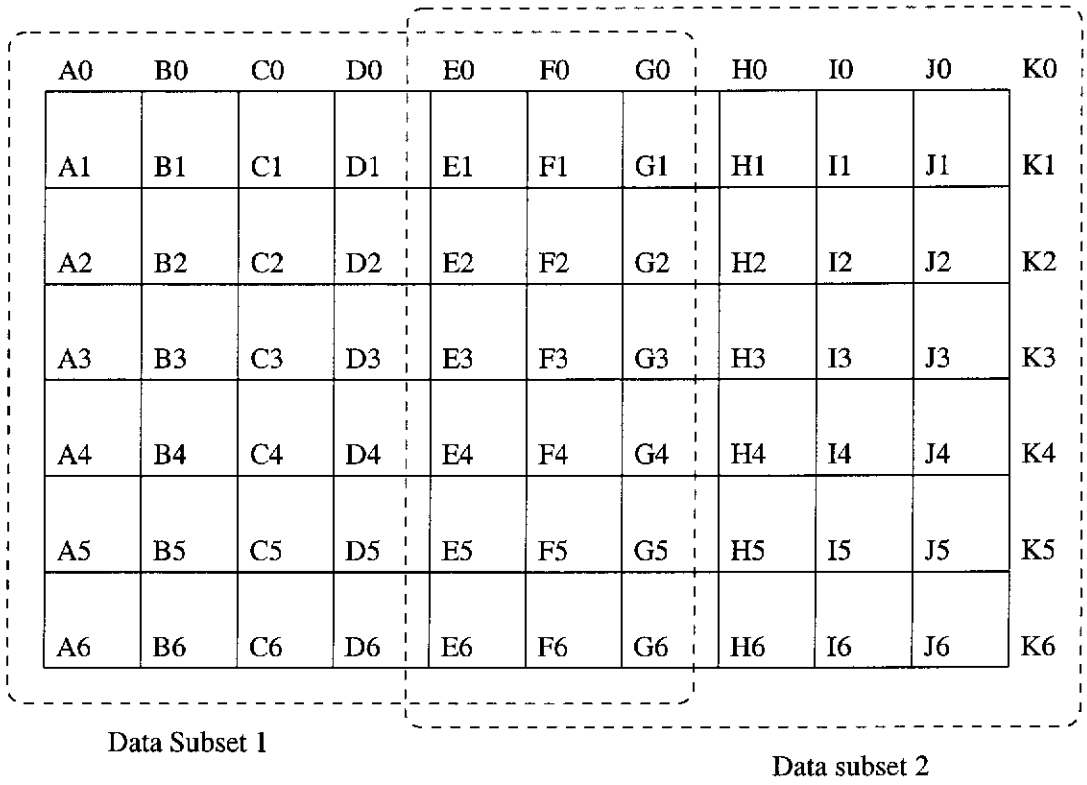


Figure 6.1: Generating subsets of the Simple Grid data set. This figure shows a two 7×7 subsets, with a shift of 4 units.

until the end of training with data subset 2. When using linear neighbourhood decay, neighbourhood size is decayed from 13 to 0 over a period of 75 epochs (the end of the initial training phase). During the second training phase, a neighbourhood size of 0 is used. In both Butterworth and linear decay experiments, a constant learning gain of 0.2 is used.

To eliminate statistical variation, each experiment was performed 10 times, and the $\mathcal{Q}$ metric for each trial averaged. The value of each $\mathcal{Q}$ metric was based upon a randomly selected set of 500 test points from within the domain of the relevant data subset.

The results of these trials can be found in Figures 6.2–6.7. Although there is wide variation in the extent of the shift used in these trials, analogous behaviour is observed for all values of shift. The results obtained using a shift of 4 units (Figures 6.4 and 6.5) will be used as specific examples for the purposes of discussion in this section; results using a shift of 1 (Figures 6.2 and 6.3) and a shift of 8 (Figures 6.6 and 6.7) can be extrapolated from this discussion.

The $\mathcal{Q}$ metric plots (Figures 6.4(g) and 6.5(g)) of each trial show three curves: the

$\mathcal{Q}$ metric representing the topological quality of each data subset, and the $\mathcal{Q}$ metric representing the topological quality of the original source data set. The performance of each SOM is functionally identical at the end of the first period of training. The $\mathcal{Q}$ metric for data subset 1 reaches an asymptotic value of 0.0003 using Butterworth decay and a value of 0.0002 using linear decay. The trials utilising linear neighbourhood decay reach these asymptotic $\mathcal{Q}$ metric values at a slower rate than their Butterworth counterparts; this is a function of the period selected for linear neighbourhood decay. The output state resulting from this initial period of training can be seen in Figures 6.4(a) and 6.5(a); these clearly show a well ordered, evenly spread map topology.

Since data subset 2 shares some training vectors with data subset 1, this initial period of training yields a SOM which contains a partial representation of data subset 2. Consequently, the $\mathcal{Q}$ metric value for data subset 2 is also asymptotic at epoch 75. A value of 0.0018 is reached using Butterworth decay, and 0.0016 is reached using linear decay. The practical consequence of this can be seen in Figures 6.4(b) and 6.5(b). The part of data subset 2 which is shared with data subset 1 forms a good grid topology. However, very poor spreading is observed in the new training vectors. These vectors are compressed into a single column on the edge of the SOM — a topologically consistent, but functionally useless representation.

The $\mathcal{Q}$ metric for the combined data set (i.e., the full grid) is a combination of these two values; 0.0018 using Butterworth decay and 0.0016 using linear decay. Figures 6.4(c) and 6.5(c) show the output state of the combined data set. This output is a combination of the output of the two data subsets, with good representation of most of the data set but compression on the edge of the map where the new training vectors from data subset 2 should fall.

Regardless of the neighbourhood decay scheme used, the $\mathcal{Q}$ metric for data subset 1 has reached the best topological representation by epoch 75; this should not be surprising, as up to this point, the map has been trained using data exclusively from data subset 1. The $\mathcal{Q}$ metric values for data subset 2 and the full data set are somewhat higher but do reach an asymptotic value representative of their partial representation in the SOM topology.

However, after epoch 75, a dramatic change is observed. Trials utilising Butterworth decay exhibit a switch between asymptotic values. The $\mathcal{Q}$ metric value for data subset 2 improves to the level previously observed for data subset 1. Conversely, the $\mathcal{Q}$ metric value for data subset 1 rises to the level previously demonstrated by data subset 2.

By the end of training at epoch 150, the $\mathcal{Q}$ metric values for the two data subsets have again reached asymptotic values; these asymptotic values are reversed from their condition at epoch 75. This reflects the change in training source. The distribution of source data has shifted, and the SOM has adapted to represent this new training source.

Although the $\mathcal{Q}$ metric value for the data subsets change as the training source changes, the asymptotic $\mathcal{Q}$ metric value for the combined data set does not change substantially between epoch 75 and 150. Just after epoch 75, there exists a short period of time during which the data learned during the initial training period have not been lost but the new data has not been fully reinforced. During this period, which lasts no more than 30 epochs, the $\mathcal{Q}$ metric value for the combined data set slightly improves. However, this improvement is lost as data subset 2 is further reinforced. The benefit of training with data subset 1 is lost as training with data subset 2 persists.

The Butterworth step decay algorithm is able to adapt to changes in the data subsets regardless of the size of the shift between the data subsets. However, the performance of the algorithm does decay slightly as the size of the shift is increased. The worst observed performance occurs with a shift of 8. Using a shift of this size, the $\mathcal{Q}$ metric value for data subset 2 greatly improves but never reaches the level obtained by data subset 1. This is not entirely surprising: when a shift of 8 units is used, subset 2 shares no data with subset 1. Data subset 2 is effectively an entirely new data set, which shares a gross topology, but nothing else, with the original data subset. Consequently, phase 2 of the experiment becomes an attempt to learn an entirely new topology. This learning can be guided by the gross topology established while training data subset 1 but, as the shift increases, the benefit of this guidance decreases.

While this decay in performance is less than ideal, it is important to note that it is a dramatic improvement over the performance observed in systems utilising linear decay (Figures 6.3, 6.5 and 6.7). These systems do not adapt at all. Systems trained using linear decay reach their asymptotic $\mathcal{Q}$ metric values at the point at which neighbourhood size decays to 0. Beyond this point, the $\mathcal{Q}$ metric value does not substantially change, regardless of changes to the training data source. This asymptotic value is not perfectly constant — a small amount of localised distortion is observed. This can be attributed to two sources. Firstly, the $\mathcal{Q}$ metric is the result of calculations performed using a randomly generated test set; and secondly, although neighbourhood size is 0, gain is not — weights on the winning neuron are still updated, and as a result, the

winning neuron (and therefore the topology) can change. However, the extent of this fluctuation is substantially less than the dramatic adaptation observed in the trials utilising Butterworth Decay.

Comparing results with other values of shift in the data set show that the difference between the asymptotic $\mathcal{Q}$ metric values is a function of the shift between the subsets. When the shift is small, there is substantial overlap between the contents of the two data subsets. Consequently, the training provided by using data subset 1 is substantially similar to the training provided by using data subset 2. Training using data subset 1 can therefore generate a topology which is adequate for describing data subset 2, resulting in similar asymptotic values. As the shift between the subsets is increased, the overlap in the subsets decreases. As a result, data subset 1 is decreasingly representative of data subset 2, and the difference between asymptotic values increases.

One interesting feature of all the trials is the spike in the $\mathcal{Q}$ metric value observed at epoch 75. This spike increases with the size of the shift between the data subsets. This characteristic is essentially an edge effect, associated with the sudden change in the training source. When the training data subset changes, the new subset contains training examples which stimulate inputs which were not previously undergoing stimulation. As the shift increases in size, the number of novel inputs also increases, resulting in a larger $\mathcal{Q}$ metric spike. In the Butterworth decay trials, the map is able to rapidly adapt to this edge effect, the spike being only a temporary setback. Linear decay trials, however, are limited in their capacity to adapt. If the shift (and resulting spike) is small, the presence of nonzero gain can overcome the spike. However, if the shift is large, it is difficult for the map to compensate. Figure 6.7 shows the effect of a map which suffers catastrophic failure with the presentation of a substantially shifted data subset. This failure is due to the extremely poor representation of data subset 2, at the edge of the map.

6.3.2 Questionnaire

In the second experiment, a Questionnaire data set (see Section A.4) is used. Changes in the source data is achieved in a similar manner to that used for the Grid data set. The complete data set consists of $5 + n$ questions; this data space is split into two overlapping subsets of 5 questions each. The changing data source can be thought of as a Questionnaire that changes over time as new questions are added and old questions are removed. Data subsets representing shifts of 1, 3 and 5 questions were generated

for this experiment. This represents a range of shifts from a minor shift, to a complete change in the data source.

These data subsets were trained on a 15×15 neuron map. The first period of training lasted for 75 epochs, using data subset 1. After this initial period of training, data subset 2 was used for a further 75 epochs of training. When using Butterworth neighbourhood decay, neighbourhood size is decayed from 11 to 2, with the step occurring at 10 epochs. When using linear neighbourhood decay, neighbourhood size is decayed from 15 to 0 over a period of 75 epochs. In both Butterworth and linear decay experiments, a constant learning gain of 0.2 is used. To eliminate statistical variation, each experiment was performed 10 times, and the $\mathcal{Q}$ metric for each trial averaged.

The results of these trials can be found in Figures 6.8–6.10. As with the Simple Grid data set experiments, the Butterworth Step method allows for continuous learning of the new data set. $\mathcal{Q}$ metric plots for maps using Butterworth neighbourhood decay can be seen in Figures 6.8(a), 6.9(a) and 6.10(a). Prior to epoch 75, each $\mathcal{Q}$ metric plot asymptotes at a state where data subset 1 has a better representation than data subset 2. The $\mathcal{Q}$ metric plot for the combined data set asymptotes at a value which is the approximate average of these two subset $\mathcal{Q}$ metric values. At epoch 75, the new data subset is introduced, the map quickly responds, and asymptotes at a new state in which data subset 2 holds a better representation that data subset 1. The combined data set maintains an asymptotic value consistent with that observed prior to the change in data subset.

The behaviour of maps utilising linear decay is also consistent with previously observed behaviour. $\mathcal{Q}$ metric plots for maps using linear neighbourhood decay can be seen in Figures 6.8(b), 6.9(b) and 6.10(b). In each trial, the map reaches a steady state by epoch 75. However, when the data subset is changed, the map is unable to adapt to the new data set, instead settling at a suboptimal asymptotic level.

There is one significant difference between the behaviour observed with the Simple Grid data set and the Questionnaire data set. In the Questionnaire experiments, the performance of the continuous learning method *improves* as the size of the shift is increased. This disparity is caused by a difference in the fundamental topology of the underlying data set. Although the Simple Grid data set consists of $7 \times (7 + n)$ inputs, the underlying topology of these inputs is 2 dimensional. The shift in the data set does not introduce any new topologies: the data from data subset 2 falls in a different range of the same dimension. When the data subset is shifted, the topological range expressed

by the weight space must also shift; a task which becomes increasingly difficult as the size of the shift increases.

However, the topology of the Questionnaire data set is $(5+n)$-dimensional, exactly equivalent to the number of questions. When the Questionnaire data set is learned, 2 of the 5 dimensions in data subset assert themselves as dominant (even though they are, in fact, no more significant). The remaining dimensions are underrepresented, or are not represented at all.

When the data set is shifted a small distance, some of these dimensions are removed from the training data. However, it is possible for a dominant topology to remain in the training set, and thus on the map. As a result, it is difficult for the $\mathcal{Q}$ metric value for data subset 2 to reach the same level as was expressed by data subset 1 before the shift.

When the shift is large, none of the original topologies exist in the new data subset; rather, the training data consists of an entirely new 5 dimensional data set. Without any old topologies attempting to assert themselves, it is an easy task for the SOM to adopt a new topological structure.

6.3.3 Scotch Whisky

Finally, the use of the Butterworth Decay for continuous learning was tested on a real-world data set — a tasting analysis of Scotch Whisky (see Section A.6). This data set is drawn from expert tasting analysis of single malt Scotch Whisky. This data set does have a topology, of sorts, but it is not a regular or evenly distributed topology.

The subset method is again used to generate change in the source data. One of the descriptors in the full data set classifies each whisky according to the location of the distillery: Highland, Lowland or Islay. The characteristics of Scotch Whisky are related to the locality in which they are distilled; by creating subsets of the complete data set which omit various localities, it is possible to generate overlapping subsets within the irregular topology of the data set.

The complete Scotch data set consists of one sample from each of the 109 Scotch Whisky distilleries. Of these, 89 are Highland whiskies, 12 are Lowland whiskies, 8 are from the Islay region. Three subsets were generated, with each subset omitting whisky originating from a single region. Any two of these subsets will have one locality overlap. By presenting one subset during the first phase of training and a different subset during the second phase, a shift in the training source can be simulated. The combination of

Data Subset	Locality omitted	N^o Vectors
A	Lowland	97
B	Islay	101
C	Highland	20

(a) Locality omissions used to generate data subsets

Trial	Training Subset 1	Training Subset 2
1	Data Subset A	Data Subset B
2	Data Subset B	Data Subset C
3	Data Subset A	Data Subset C

(b) Data subsets used in experimentation

Table 6.1: Generating subsets of the Scotch whisky data set.

pairs of subsets yields a range of shift sizes, due to the differing sizes of each subset. The formation of the data sets, and the three pairs of subsets used during each of the three trials, are shown in Table 6.1.

These data subsets were trained on a 25 × 25 neuron map. The first period of training lasted for 75 epochs, using data subset 1. After this initial period of training, data subset 2 was used for a further 75 epochs of training. When using Butterworth neighbourhood decay, neighbourhood size is decayed from 18 to 2, with the step occurring at 10 epochs. When using linear neighbourhood decay, neighbourhood size is decayed from 25 to 0 over a period of 75 epochs. In both Butterworth and linear decay experiments, a constant learning gain of 0.2 is used. To eliminate statistical variation, each experiment was performed 10 times, and the $\mathcal{Q}$ metric for each trial averaged.

The results of these trials can be found in Figures 6.11–6.13. Figure 6.11 is trained with a data set initially consisting of no Lowland whiskies; secondary training contains Lowland whiskies, but no Islay whiskies. Figure 6.12 is trained with a data set initially consisting of no Islay whiskies; secondary training contains Islay whiskies, but no Highland whiskies. Figure 6.13 is trained with a data set initially consisting of no Lowland whiskies; secondary training contains Lowland whiskies, but no Highland whiskies. Although these results are not as clear as those observed using synthetic data sets, the continuous learning nature of the Butterworth Step method can still be observed. Linear neighbourhood decay is again unable to adapt to changes in the training data source.

When using Butterworth decay (Figures 6.11(a), 6.12(a), and 6.13(a)), the $\mathcal{Q}$ metric value for data subset 2 improves after the change in training subset at the epoch 75. Conversely, the $\mathcal{Q}$ metric value for data subset 1 becomes worse. This behaviour is not observed when using linear neighbourhood decay (Figures 6.11(b), 6.12(b), and 6.13(b)). Although the map is able to form a good representation of the initial data subset, the map is unable to adapt to the shifted data set which is presented after epoch 75. This is consistent with a map which has reached a static state at the end of the first 75 epochs of training.

The extent of changes in $\mathcal{Q}$ metric value in the Butterworth decay experiments is moderated by the size of the shift between the data subsets. In Figure 6.11, the overlap between subset 1 and subset 2 consists of 89 training examples (of 109). The presentation of data subset 2 therefore involves the re-presentation of many of the vectors in data subset 1. As a result, the magnitude of change in $\mathcal{Q}$ metric values is not particularly large. However, in Figure 6.12, the overlap between the two data subsets is a mere 12 vectors. As a result, the learning of data subset 2 is effectively a completely new training task, with the topology representing data subset 1 being lost in the process of learning a topology for data subset 2.

One interesting characteristic of these results is the extent of decay in the representation of data subset 1 during the second phase of training in Figures 6.12(a) and 6.13(a). In the experiments with synthetic data sets, $\mathcal{Q}$ metric values are observed to 'swap' during the second phase of training; that is, data subset 2 improves to a $\mathcal{Q}$ metric value comparable to that previously held by data subset 1 and vice versa. However, in these trials, the representation of data subset 1 is observed to dramatically decay after the introduction of data subset 2.

This can be explained by considering the composition of the training subsets. In the Simple Grid and Questionnaire experiments, the two data subsets are of equivalent size and topological density (i.e., inputs are evenly distributed over their domain. Consequently, during the second phase of training, data subset 1 is replaced with a new data subset of equal size and density. As a result of the symmetry between the two data sets in the two training phases, the extent to which data subset 2 is not represented in the first training phase is equivalent to the extent to which data subset 1 is not represented in the second training phase. This is observed as a 'swap' in $\mathcal{Q}$ metric values for the two data subsets.

However, the Scotch Whisky data does not share this characteristic; data is not

evenly distributed over the problem domain, and the data subsets are not of equivalent size. Consequently, there is no relationship between the $\mathcal{Q}$ metric values for the two data sets after training data is swapped. The extent of the decay in $\mathcal{Q}$ metric is exacerbated by the size of the two training subsets. In the second two Scotch Whisky trials, the second data subset contains only 20 training vectors; under the Butterworth scheme, these vectors are allowed to reinforce their topology without restraint. As a result, the topology of the much larger initial training set is substantially lost.

This skew in the data set also affects the $\mathcal{Q}$ metric value for the full data set. In the synthetic experiments, the full data set was equally composed of the two data subsets. As a result, the overall $\mathcal{Q}$ metric value is largely unaffected by a change in training subset. The Scotch Whisky data subsets are not similarly balanced. The Highland whisky subset comprises a major part of the full data set; when this data set is removed from the training process, a much larger drop in overall performance is observed.

6.3.4 Discussion

The examples presented in this chapter clearly demonstrate that a Kohonen style SOM can be made to learn continuously. The power of this technique is based solely upon maintaining non-zero neighbourhood size. Kohonen's algorithm is unable to expand its knowledge representation beyond the initial training data set purely because after initial training, the neighbourhood size is decayed to 0; as a result, the potential for further learning is limited. By maintaining non-zero training parameters in the SOM, it is possible for the SOM to adapt to changes in the training space of the SOM. At any given time, the state of the map will represent the training set currently in use. Given a significant period of stationarity in this training set, the map will stabilize to a consistent topology of the current training set. Therefore, it is not necessary to have a canonical training set at the commencement of training. All that is required is a sufficiently spanning data set to establish an initial global topology.

It is also worthy of note that this technique allows the use of training sets which do not fall into the traditional 'batch of data' category. Although the examples presented in these experiments all train upon static sets of data, the continuous learning mechanism has the potential to allow training data to be drawn from a non-repeating source. An example of such a source would be a real world questionnaire; rather than collecting 10 examples of questionnaire data, and repeatedly presenting them, a continuous stream of

unique examples could be collected and presented once to the SOM. The map topology then become a function of the statistical distribution of training vectors drawn from the training source. If this training source is non-stationary, the topology represented by the map will evolve over time to represent the current statistical distribution from the training source.

Although the rapid adaptation of the SOM is a desirable characteristic, it does raise one significant problem. The SOM algorithm effectively forms a probability density map of the input set. The area given over to a concept on a trained SOM is directly related to the frequency of the training patterns representing that concept in the training set. This is a useful characteristic when using static data sets. If the user wishes to have high resolving power of a given concept, they need only bias the training set to give an emphasis to that concept.

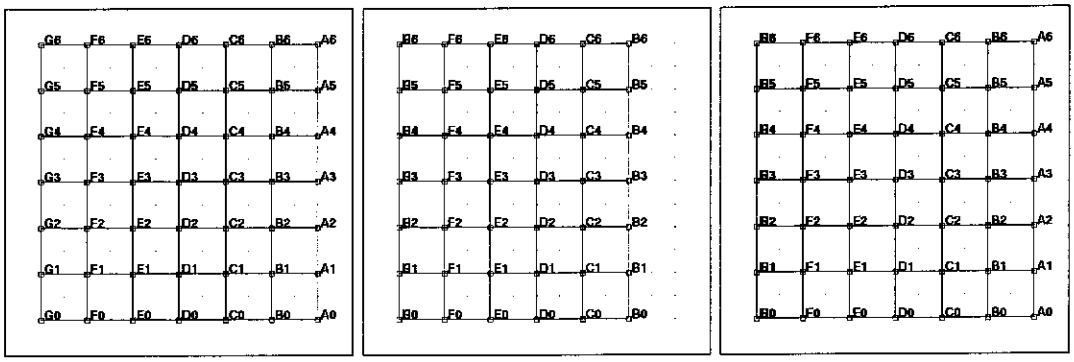
However, in a dynamic learning environment, the training set changes constantly, and consequently, so does the probability distribution underlying the training set. If the SOM adapts to a dynamic data set, the probability density map provided by the SOM will, at any given time, represent only the most recent training examples, not the range of data presented over the entire history of training. Training examples which are not repeated will, over time, lose their representation in the map, regardless of the significance of these training examples. That is, the continuously learning SOM has no long-term memory.

While a continuous learning mechanism capable of short-term, forgetful memory (such as a SOM using Butterworth Step Decay and a dynamic training set) may be useful for some applications, long term memory also has significant uses. Applications which need to learn from experience cannot afford to lose the benefit of significant training examples simply because they occurred early in training and have not been repeated recently. To be of general use, a method for overcoming this forgetfulness must be found. This issue will be addressed in Chapter 8.

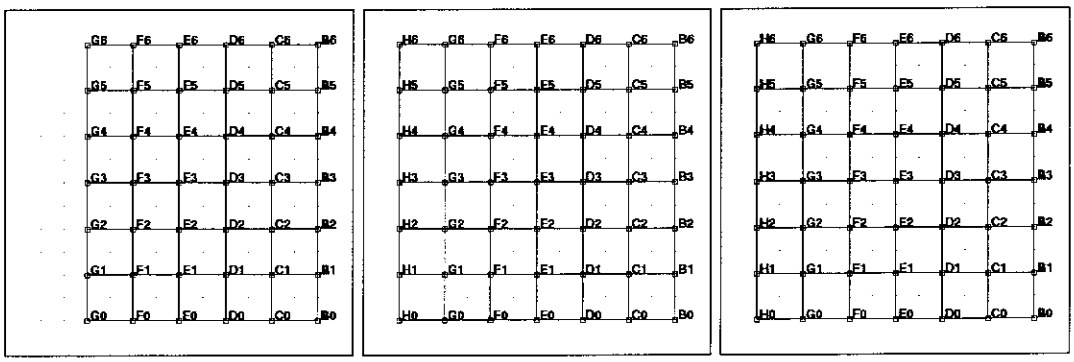
6.4 Conclusion

In this chapter, it was shown that by decaying learning parameters to non-zero levels, it is possible for a Kohonen-style SOM to adapt to changes in the training data source. By maintaining non-zero neighbourhood size, the SOM becomes a constantly adjusting representation of a data source. If the data source changes, so does the representation. It is therefore possible for a Kohonen-style SOM to learn continuously.

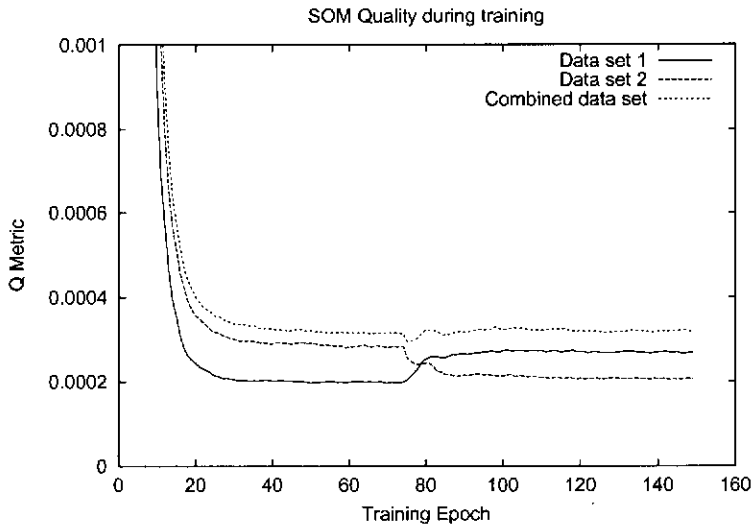
This method of using non-zero neighbourhood size does raise the issue of forgetfulness. By remaining flexible to changes in the training source, the SOM will forget training examples which are not reinforced. Applications which need to learn from experience cannot afford to lose the benefit of significant training examples simply because they occurred early in training and have not been repeated recently. To be of general use, a method for overcoming this forgetfulness must be found. This issue will be addressed in Chapter 8.



(a) Epoch 75: subset 1 (b) Epoch 75: subset 2 (c) Epoch 75: full set

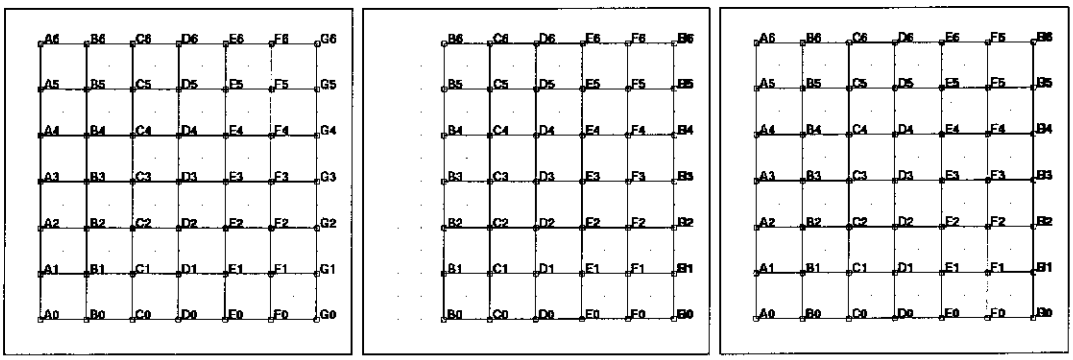


(d) Epoch 150: subset 1 (e) Epoch 150: subset 2 (f) Epoch 150: full set

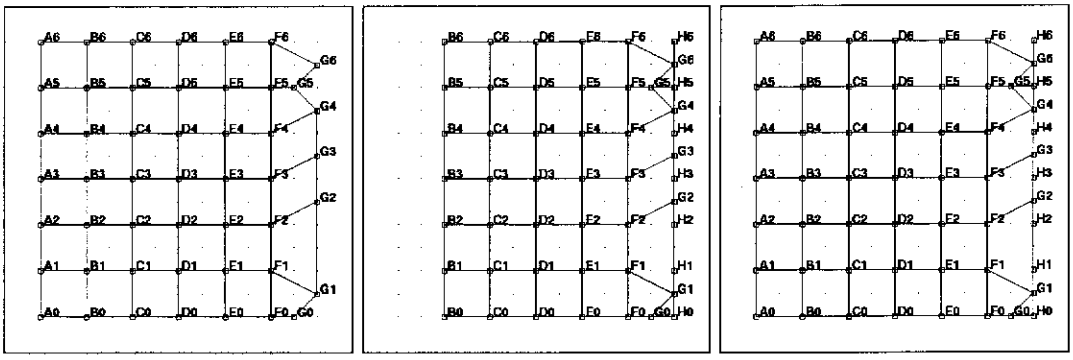


(g) Map Quality during training

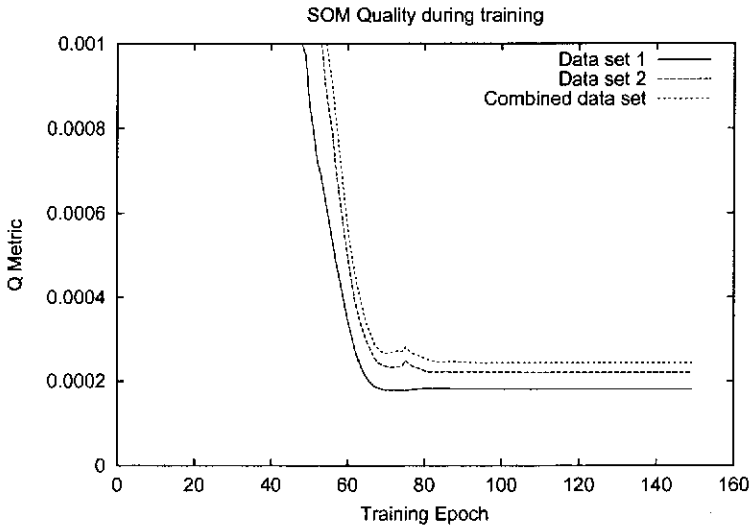
Figure 6.2: Continuous learning of a Simple Grid data set with a Kohonen-style SOM, using Butterworth step neighbourhood decay. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 2 is formed by shifting subset 1 by 1 unit. The full data set is the union of these two subsets.



(a) Epoch 75: subset 1 (b) Epoch 75: subset 2 (c) Epoch 75: full set

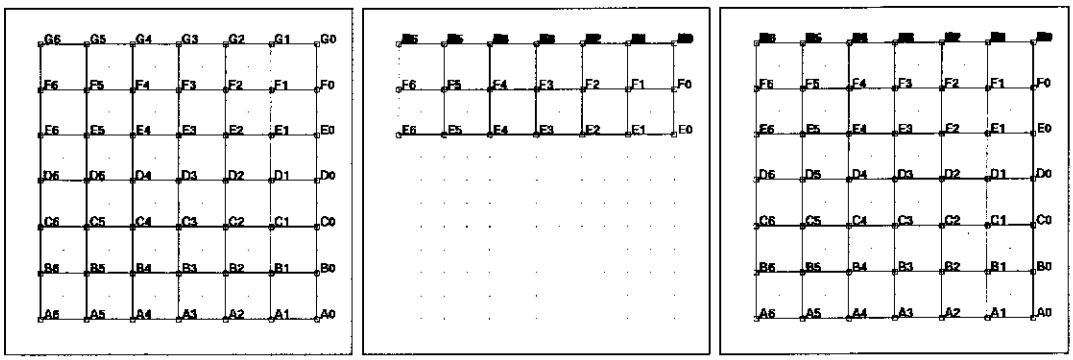


(d) Epoch 150: subset 1 (e) Epoch 150: subset 2 (f) Epoch 150: full set

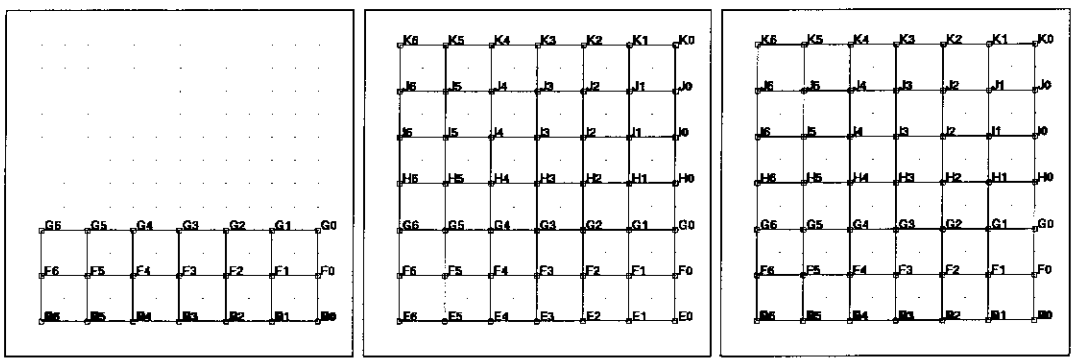


(g) Map Quality during training

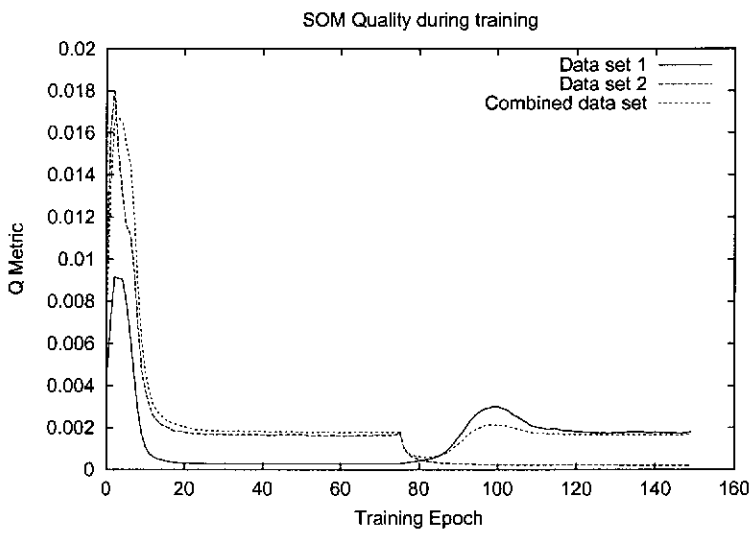
Figure 6.3: Continuous learning of a Simple Grid data set with a Kohonen-style SOM, using linear neighbourhood decay. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 2 is formed by shifting subset 1 by 1 unit. The full data set is the union of these two subsets.



(a) Epoch 75: subset 1 (b) Epoch 75: subset 2 (c) Epoch 75: full set

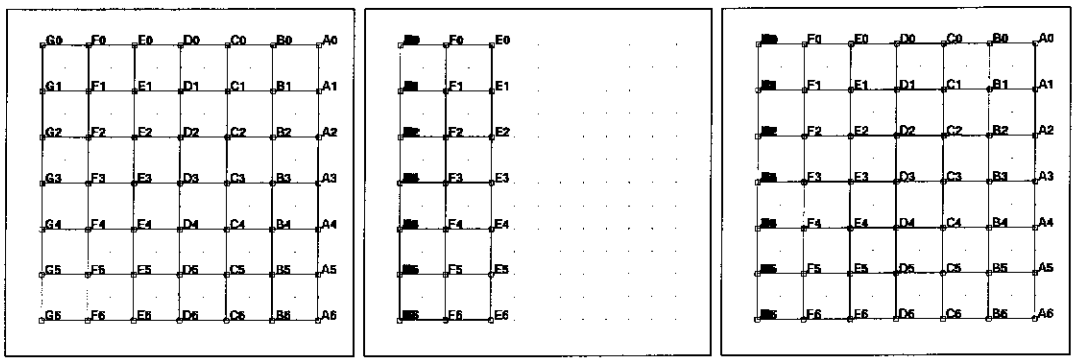


(d) Epoch 150: subset 1 (e) Epoch 150: subset 2 (f) Epoch 150: full set

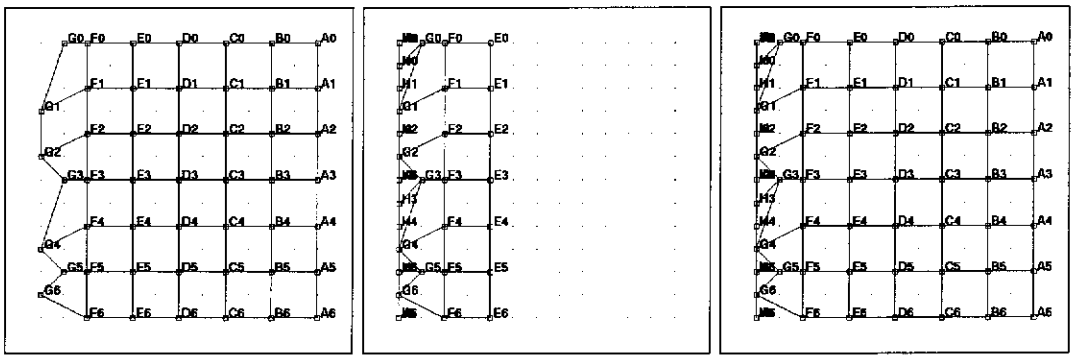


(g) Map Quality during training

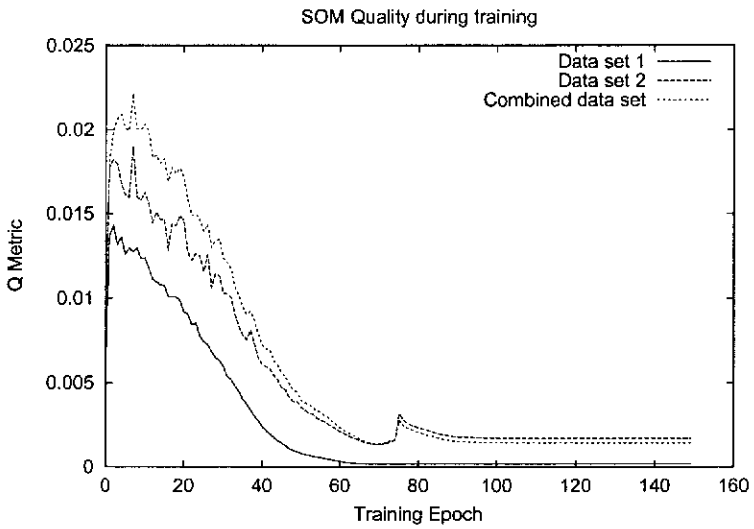
Figure 6.4: Continuous learning of a Simple Grid data set with a Kohonen-style SOM, using Butterworth step neighbourhood decay. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 2 is formed by shifting subset 1 by 4 units. The full data set is the union of these two subsets.



(a) Epoch 75: subset 1 (b) Epoch 75: subset 2 (c) Epoch 75: full set

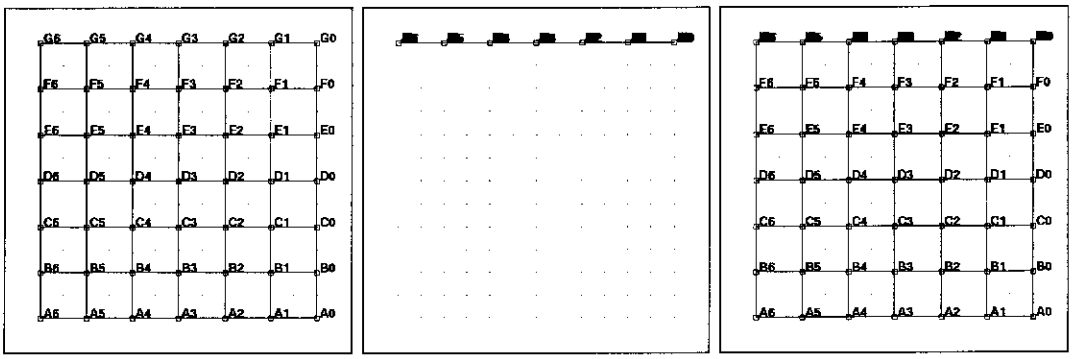


(d) Epoch 150: subset 1 (e) Epoch 150: subset 2 (f) Epoch 150: full set

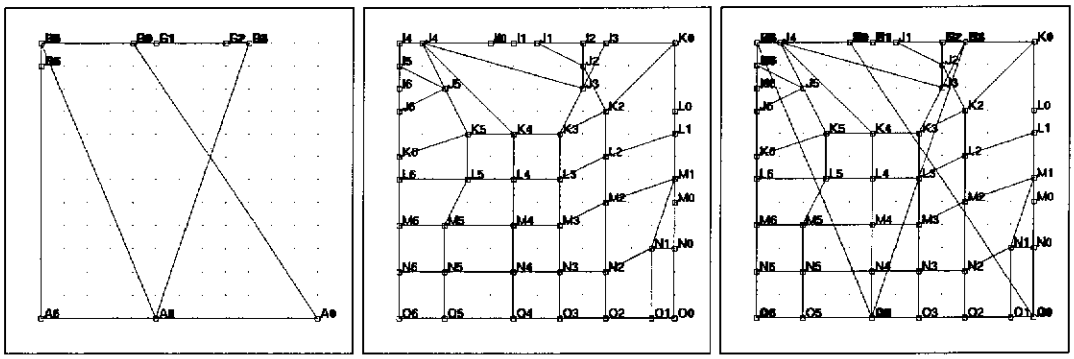


(g) Map Quality during training

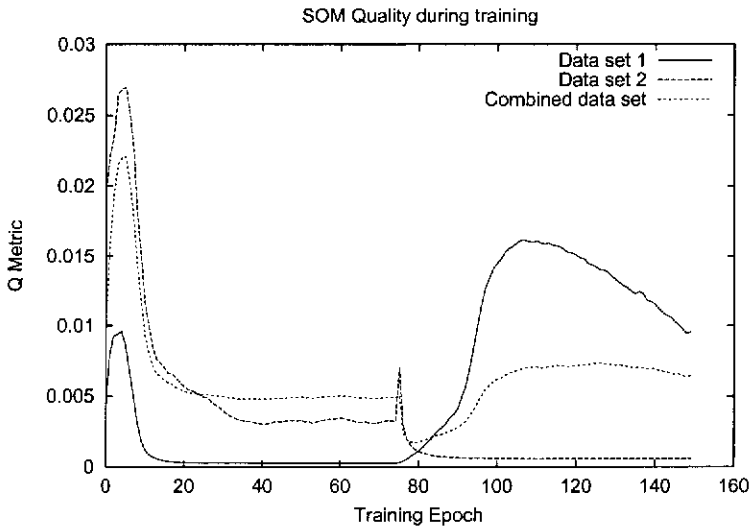
Figure 6.5: Continuous learning of a Simple Grid data set with a Kohonen-style SOM, using linear neighbourhood decay. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 2 is formed by shifting subset 1 by 4 units. The full data set is the union of these two subsets.



(a) Epoch 75: subset 1 (b) Epoch 75: subset 2 (c) Epoch 75: full set

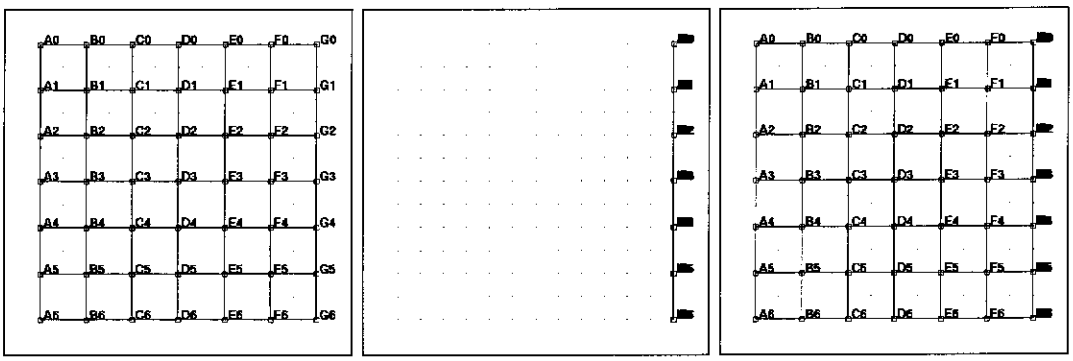


(d) Epoch 150: subset 1 (e) Epoch 150: subset 2 (f) Epoch 150: full set

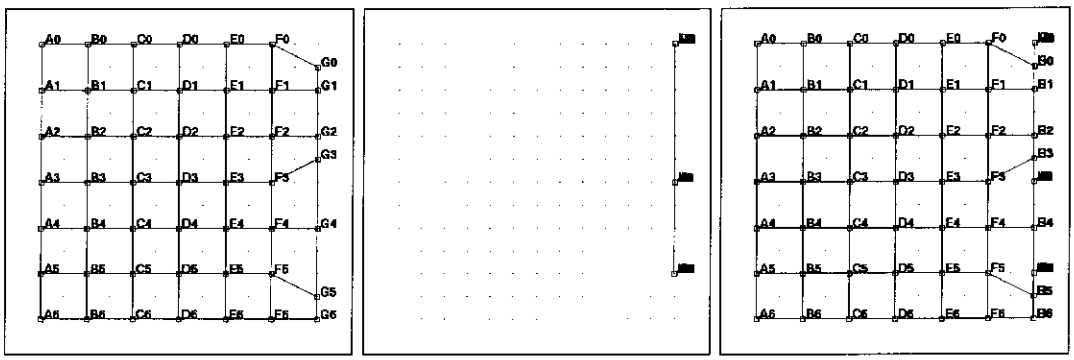


(g) Map Quality during training

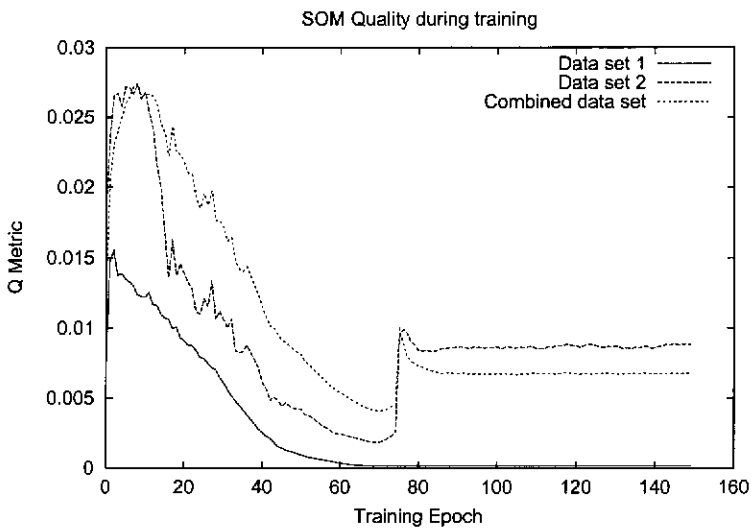
Figure 6.6: Continuous learning of a Simple Grid data set with a Kohonen-style SOM, using Butterworth step neighbourhood decay. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 2 is formed by shifting subset 1 by 8 units. The full data set is the union of these two subsets.



(a) Epoch 75: subset 1 (b) Epoch 75: subset 2 (c) Epoch 75: full set

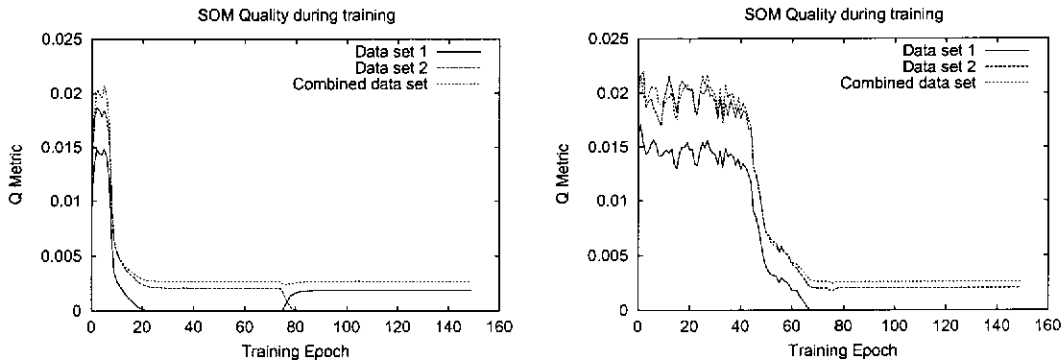


(d) Epoch 150: subset 1 (e) Epoch 150: subset 2 (f) Epoch 150: full set



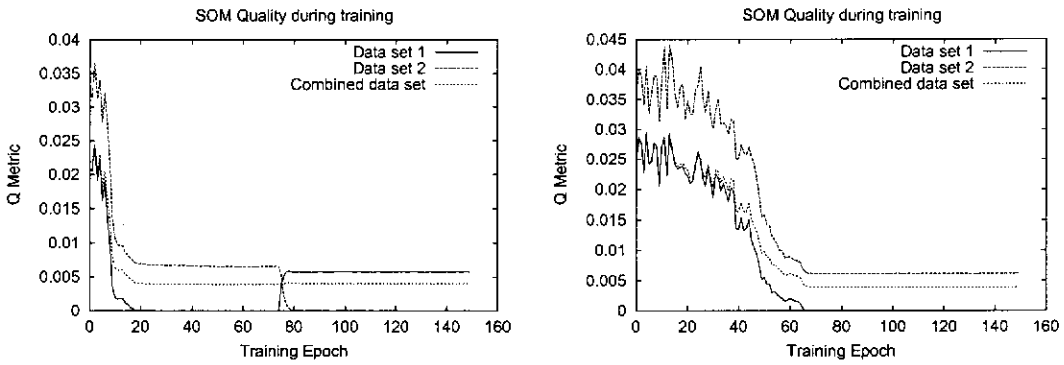
(g) Map Quality during training

Figure 6.7: Continuous learning of a Simple Grid data set with a Kohonen-style SOM, using linear neighbourhood decay. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 2 is formed by shifting subset 1 by 8 units. The full data set is the union of these two subsets.



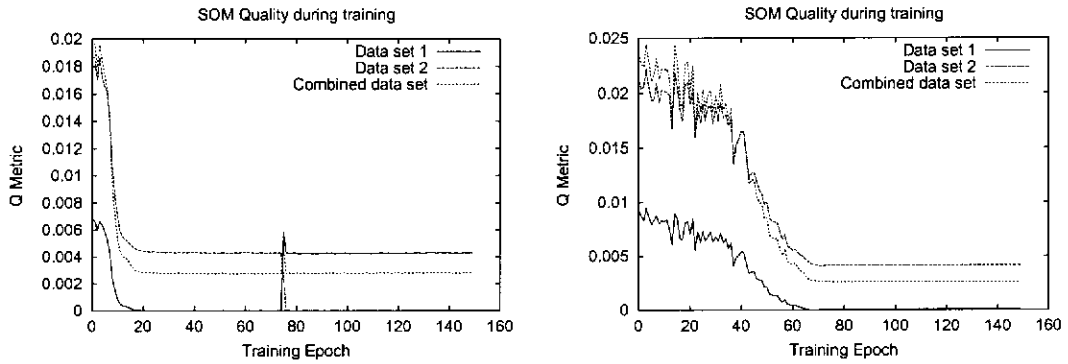
(a) Butterworth Decay (b) Linear Decay

Figure 6.8: Continuous learning of the Questionnaire data set with a Kohonen-style SOM. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Data subset 2 is created by shifting data subset 1 by 1 question.



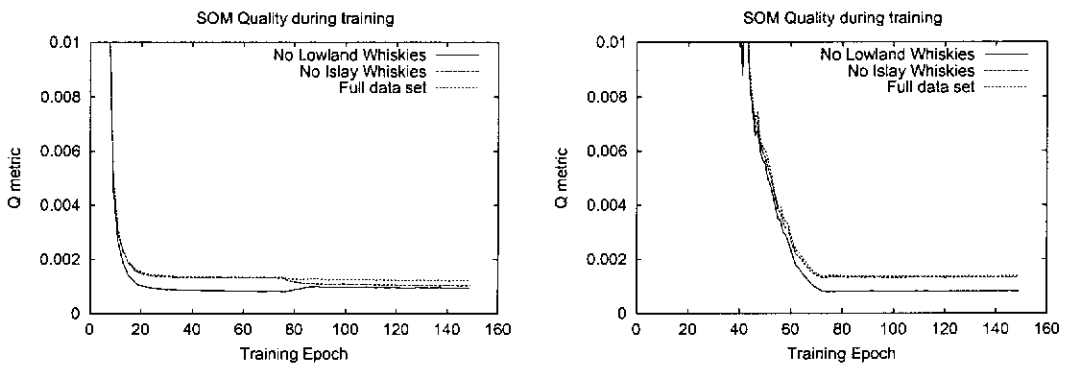
(a) Butterworth Decay (b) Linear Decay

Figure 6.9: Continuous learning of the Questionnaire data set with a Kohonen-style SOM. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Data subset 2 is created by shifting data subset 1 by 3 questions.



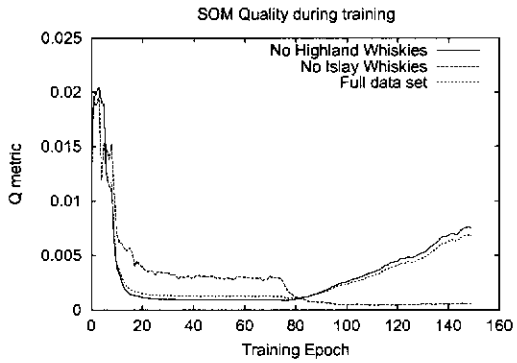
(a) Butterworth Decay (b) Linear Decay

Figure 6.10: Continuous learning of the Questionnaire data set with a Kohonen-style SOM. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Data subset 2 is created by shifting data subset 1 by 5 questions.

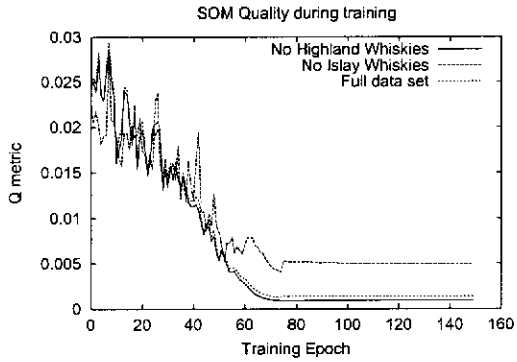


(a) Butterworth Decay (b) Linear Decay

Figure 6.11: Continuous learning of the scotch data set with a Kohonen-style SOM. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 1 contains no Lowland whiskies; Training subset 2 contains no Islay whiskies. The full data set contains all whiskies.

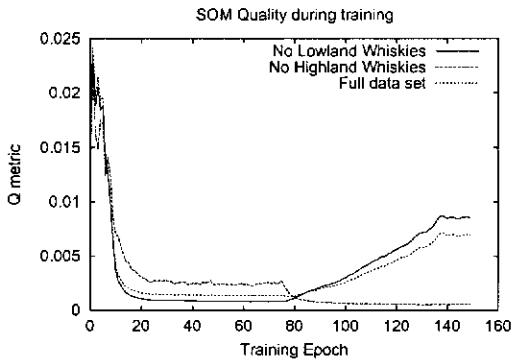


(a) Butterworth Decay

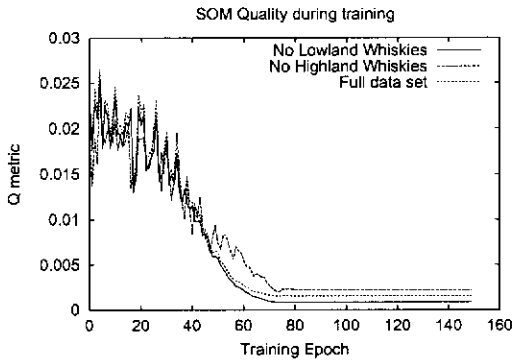


(b) Linear Decay

Figure 6.12: Continuous learning of the scotch data set with a Kohonen-style SOM. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 1 contains no Islay whiskies; Training subset 2 contains no Highland whiskies. The full data set contains all whiskies.



(a) Butterworth Decay



(b) Linear Decay

Figure 6.13: Continuous learning of the scotch data set with a Kohonen-style SOM. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 1 contains no Lowland whiskies; Training subset 2 contains no Highland whiskies. The full data set contains all whiskies.

Chapter 7

Learning from a Syllabus

> "You know, it's at times like this, when I'm trapped in a Vogon airlock with a man from Betelgeuse, and about to die of asphyxiation in deep space, that I really wish I'd listened to what my mother told me when I was young."
> "Why, what did she tell you?"
> "I don't know, I didn't listen."
> — *The Hitch-Hikers Guide to the Galaxy* (Adams, 1979)

7.1 Introduction

In the previous chapter, the basic SOM algorithm was extended so as to enable a self-organising map to add knowledge continuously to a basic representation. However, this does not account for the initial establishment of a representation.

The establishment of a good fundamental representation of a problem domain is of critical importance in almost all learning systems. A poor fundamental representation will restrict the ability of the system to develop general representations of a problem space. This is especially important for continuous learning problems. A misleading or misrepresentative fundamental representation will restrict the long term performance of a learning system, as all future representations must be based upon the initial lessons learned by the system.

In this chapter, a method for developing a strong fundamental topology is presented. This technique is known as *Syllabus Presentation*; it is similar in nature to incremental presentation techniques which have been successfully applied to feedforward networks learning grammatical data (Plunkett and Marchman, 1990)(Elman, 1993). A method for developing a training syllabus is also presented. This technique, known as *Percept Masking* is based upon masking training inputs as a result of prior domain knowledge. The syllabus presentation technique is demonstrated with SOMs using two simple data

sets: a Grid-in-a-Grid, and a 4 question Questionnaire. The technique is then demonstrated on the Scotch Whisky training data.

7.2 Syllabus Presentation

A common characteristic of machine learning systems is the significant effect that early training samples have on the fundamental structure of the system. For example, kittens raised in an environment which contains only vertically oriented visual stimuli are unable in adulthood to see horizontally oriented features (Blakemore and Cooper, 1970). The composition of training of data during the formative period of development is therefore capable of skewing the decision making power of a system over its entire lifetime.

Furthermore, the effect of false or misleading training information is more significant during the initial stages of training. After a prolonged period of training, a system has sufficient experience to identify and ignore (or at least reduce the significance of) training data which deviate from expectation. During initial training, this experience has not yet been accumulated and all training information has the illusion of significance. For example, the root partition of a decision tree is formed from the information available at the start of training. The validity of this root partition is never subsequently challenged. If the data set changes in such a way as to introduce a new significant partition, or to reduce the significance of an initially strong partition, the decision tree cannot and does not compensate. Pruning and merging strategies can help to overcome partitioning discrepancies on leaf nodes. However, if a partitioning problem is caused by the root partition, the decision tree cannot be fixed without rebuilding the entire decision tree. Rebuilding from the root node is extremely inefficient, as it requires the loss of all previous training knowledge.

The significance of initial data partitions suggests a method for overcoming one of the key aspects of the stability-plasticity dilemma: the difference between infrequent but important and infrequent, unimportant training information. In an ideal system, important information would not be lost, regardless of presentation frequency. Conversely, unimportant information would never be integrated into the knowledge representation. The oversensitivity of systems to initial training examples provides a mechanism by which the preservation of important knowledge can be ensured.

The continuous learning problem is generally considered to involve an ability to learn and operate after the first training pattern is presented (Protzel *et al.*, 1998). Under

these conditions, the suitability of initial training examples cannot be determined or controlled. As a result, if the initial training data is not representative of the overall problem space, the initial partitioning of the data space will not be representative of the overall problem. This could have a significant detrimental effect on the long term performance of the system.

An alternate approach is to provide a short period of guided learning at the start of training. During this guided training period, all training knowledge comes from a well established training set, embodying core knowledge from a syllabus consisting of a known subset of the full problem domain. If this syllabus consists solely of knowledge known to be significant, this knowledge will form the fundamental knowledge representation of the system. Consequently, this knowledge will be difficult to 'unlearn', without a complete reorganisation of the system. Later training can introduce less significant knowledge/ or knowledge from a different part of the problem domain, without the risk of losing a significant representational component.

This approach has been successfully applied to the learning of static language and grammar problems by traditional feedforward networks. Plunkett and Marchman (1990) have shown that neural architectures demonstrate better overall learning when the training corpus is allowed to slowly grow in size. Elman (1993) demonstrated similar results using a series of staged training sets. In these experiments, the neural network was unable to learn a grammar when the training data was presented all at once.

The syllabus approach to learning is also a feature of everyday experience. For example, consider the problem of teaching a child integral calculus. In attempting to teach a child, the teacher does not give the child a book full of calculus problems and hope that the child will be able to 'maximise entropy' over this training set and develop an optimal representation — and therefore understanding — of calculus. Rather, the teacher starts by teaching simple arithmetic problems, progresses to simple algebra, eventually culminating in calculus. This progression of training occurs over a prolonged period of time. The change from a sparse, simple training set to a complex data set does not occur until the simple training data is mastered.

7.2.1 Syllabus Presentation for SOMs

As discussed in Chapter 5, the principal topology of a SOM is determined during the initial stages of the learning process. During this time, the neighbourhood size is large,

and a consistent global topology develops. The local topological relationships on the map are then refined during a protracted period of training with a small neighbourhood size.

Self-organising maps can also be applied to continuous learning problems. It was shown in Chapter 6 that by decaying neighbourhood size to a non-zero asymptotic value, a SOM is able to continuously incorporate new knowledge into its output representation. The chosen asymptotic value for neighbourhood size should reflect the desired resolving power of the SOM. However, if this value is kept small, fundamental changes in global map topology are prevented, as no single training exemplar is able to affect the entire map.

These features of SOM learning suggest that syllabus presentation would be well suited for use in the training of a continuously learning self-organising map. If, during the initial training period, the training set is restricted to a specific subset, the principal topology of the SOM will be defined by the data present in this subset. Once this initial topology is established, a reduced neighbourhood size can be used to refine the local topology and incorporate training data from new parts of the problem domain; however, the reduced neighbourhood size prevents major shifts in the global topology.

This leaves the problem of how to develop a training data set which can be used for syllabus training. In a lifelong learning and development context (such as the life-cycle model of Figure 3.1, the training syllabus can be identified by a 'parent' figure (or, in the case of an orphaned or parentless system, a mentor or trainer). This parent figure would extract the 'most useful' training examples, and use them to create a syllabus. However, in the context of a once-off learning system, no 'parent' exists. Rather, an automated system for developing a syllabus from an initial training batch is required.

7.3 Percept Masking

One method for developing an SOM syllabus is to exploit the nature of the topological mapping produced by a SOM. Regardless of the input space used for training, the output space of a SOM is a two dimensional topological map. Higher dimensional relationships are generally not represented on the fundamental map topology (unless there are significant folds in this topology). Rather, they are only observed at a local level. The output representation of a SOM is therefore a low dimensionality projection of a high dimensionality training set.

However, there is no mechanism by which to choose the axis of projection or to

force a given projection to take the principal topology. The only factor determining the fundamental topology which emerges is the underlying probability distribution of the training data. Dimensions which are dominant in the training data will form the principal topology. In the absence of a significant skew in the training data, the fundamental topological dimensions of a SOM are effectively chosen at random (through the randomly instantiated initial weights). This result was demonstrated empirically in Chapter 6.

This provides a way to control the emergence of principal topologies on a SOM during learning. If we know *a priori* that one specific subset of dimensions is of more importance to the output representation than another subset, we can use this knowledge to generate a skew in the training data space. Furthermore, if we assume that there is a correspondence between the components of a training vector and the underlying topology of the data set, we can mask individual inputs (or percepts) to generate a skew in the training data.

By applying an arithmetic mask to training examples, it is possible to limit the experience of a SOM to a specific subset of the full training set. This technique is referred to as *Percept Masking*. For example, consider a 4 element training vector [1.0 0.5 0.7 0.2], drawn from a data set with an underlying 4D topology. If we assume that the elements of the training vector correspond to the dimensions of the underlying topology and that elements 2 and 3 of this vector represent the most significant dimensions, we can use a percept mask of [0 1 1 0], resulting in a training vector during syllabus training of [0 0.5 0.7 0]. A training pattern element with strength 0 results in no update to weights connected to those inputs. Therefore, the initial random weights connected to these inputs are unaffected. However, as these random initial values are small, they contribute little to the activation, and thus winner selection, on the map.

Using this technique, it is possible to ensure that certain dimensions of the training set are never presented. As a result, it is possible to prevent these dimensions from being expressed in the fundamental topology of the SOM, as these dimensions are never present during the initial period of syllabus training. Upon completion of syllabus training (i.e., a stable representation of syllabus data has been achieved), the mask can be relaxed, and the full training set can be presented to the SOM. As a result of exposure to the unfiltered, higher dimensionality training set, secondary topologies will develop within the established fundamental topology.

The use of a syllabus training set produced in this manner introduces the need for

a minor change to the neighbourhood decay scheme. Consider a 4D binary feature vector, where the spanning training set would consist of 16 patterns. If two inputs in this data set are masked, there would appear to be only 4 unique training vectors. This poses a problem for the selection of an appropriate final neighbourhood size. On a 15×15 map, the full data set would require a final neighbourhood size of approximately 2, representing half the expected distance between winning neurons. However, when training using the syllabus, a neighbourhood size of approximately 7 would be appropriate for spacing the sparse vectors of the syllabus training set.

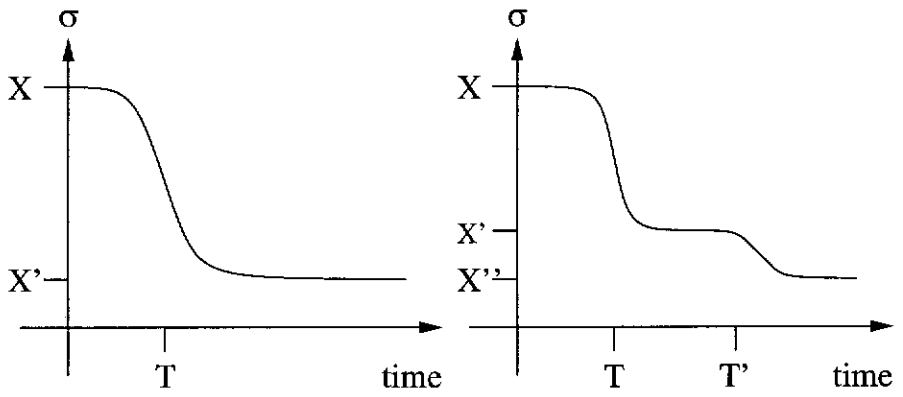
If a final neighbourhood size of 7 were to be chosen, the SOM would have insufficient resolving power to discern between the vectors in the full data set. However, a neighbourhood size of 2 is insufficient to represent the sparse syllabus training set. A small neighbourhood size such as this would result in the loss of the global topology learned during the initial training period with a large neighbourhood size.

This situation necessitates the introduction of an intermediate step of neighbourhood decay. This intermediate step can be configured to represent the expected pattern density of the syllabus training set; once this topology has been learned, the neighbourhood size can again be reduced, this time to a final value. This decay scheme is shown graphically in Figure 7.1. Under this scheme, the syllabus is used to train from time 0 to time T', and the neighbourhood size is decayed from X to X'. X' is chosen to represent the expected data resolution of the syllabus data set. The full data set is used from time T' onwards, and involves a decay to a neighbourhood size of X'', chosen to represent the expected data resolution of the full data set. In this way, it is possible to develop a well spaced topology of the syllabus data, and then refine this topology to represent the entire training set.

Although the examples presented here deal with a single syllabus, the syllabus approach could be applied iteratively, over many subsets of the training data. After an initial syllabus consisting of a 2D subset, a secondary 4D subset could be presented, followed by a tertiary 6D subset and so on, until all dimensions of the input data are represented. This could be of great use with extremely high dimensionality training sets with a complex underlying topology.

7.4 The $\mathcal{Q}$ metric and Percept Masking

In order to test the capabilities of Percept Masking, two metrics are required. Firstly, a method is required for testing topological preservation over the entire data set. Sec-



(a) Butterworth Decay (b) Two step Butterworth Decay

Figure 7.1: Normal Butterworth decay, and the two step Butterworth decay scheme, used to ensure adequate representation of a sparse syllabus.

ondly, a method is required for testing the topological preservation of specific subsets of data; for example, the subsets used for syllabus training. The $\mathcal{Q}$ metric method, presented in Section 3.4.2, is well suited to the first task; this suitability was demonstrated empirically in Chapter 5. The $\mathcal{Q}$ metric can also be used for the second task if a small modification is applied.

The $\mathcal{Q}$ metric method given in Section 3.4.2 measures the preservation of topology across an entire data set, by performing a comparison between the distance between two vectors in the data space, and the distance on the SOM between the winning neurons activated by those two vectors. In this calculation, the distance between vectors in the data space is calculated as the Euclidean distance over all dimensions of the training vectors.

However, the distance between vectors in the data space need not be a distance over all dimensions. By measuring the distance over a small subset of dimensions, it is possible to evaluate the topological preservation of those dimensions, instead of the entire data set. This modified distance calculation is also a Euclidean distance measure, but one that ignores specific dimensions of the training vectors. This method is equivalent to measuring the distance between vectors in an orthogonal projection of the data set, rather than in the full data space.

For example, consider a 4D training set, containing the vectors [1.0 0.5 0.7 0.2] and [0.0 0.4 0.6 0.9]. These vectors are a distance of 1.2288 apart in the full data set. This distance is calculated using all four dimensions of the data set; it ignores the

obvious similarity in elements 2 and 3 of the training vectors. If inputs 2 and 3 are used as a syllabus, the 2D projection of these training vectors yields [0.0 0.5 0.7 0.0] and [0.0 0.4 0.6 0.0], which are a distance of 0.1414 apart. Similarly, the 'anti-syllabus' yields vectors [1.0 0.0 0.0 0.2] and [0.0 0.0 0.0 0.9] are a distance of 1.2206 apart. A map which places these two training vectors close together will therefore demonstrate a good topological preservation of the syllabus, and a poor representation of the anti-syllabus.

These reduced distance measures can then be used to evaluate $\mathcal{Q}$ metric values for their respective data subsets by comparing with map distances, and averaging over all training vectors. Using a selection of distance measures, representing a range of data subsets (syllabi), it is possible to develop quantifiable $\mathcal{Q}$ metric descriptions of the topological representations of specific subsets of data, in addition to the full data set.

7.5 Testing Syllabus Presentation

In the following section, percept masking is used to establish specific topologies as the global map topologies in preference to secondary topologies. These experiments are performed using the Grid-in-a-Grid data set, the Questionnaire data set, and the Scotch Whisky data set.

In each experiment, the full data set can be decomposed into two subsets of the overall dimensionality. Without syllabus presentation, the SOM algorithm will attempt to produce a good overall topology which represents each subset proportional the frequency of that subset in the full training set. The subsets of the Grid-in-a-Grid are equally represented in the data set. Therefore, the $\mathcal{Q}$ metric value for each subset should be equal. Similar behaviour should be observed for the Questionnaire data set. The Scotch Whisky data set, however, is not evenly distributed over all dimensions. Therefore, the natural representation of this data set will exhibit preferential representation of some subsets.

However, *a priori* knowledge may suggest that one of the two subsets is of greater significance. If this is the case, it would be desirable for the SOM to give preferential representation to the significant subset. The purpose of syllabus training is to realise this preferential representation. If successful, the map should develop a lower $\mathcal{Q}$ metric value for the significant subset than was observed without syllabus presentation. Similar, a higher $\mathcal{Q}$ metric value for the less significant subset should be observed.

The effect on the overall topology will depend upon the composition of the data set. If the dimensional subsets are equally represented in the data set (as is the case with

the Grid-in-a-Grid and Questionnaire data sets), the overall $\mathcal{Q}$ metric value should be unaffected. Topological improvements in one subset will be evenly compensated for by topological losses in the other subset. However, in the case of a data set which does not have equal representation of subsets (such as the Scotch Whisky data set), improvements in one subset will not be evenly balanced by losses in other subsets; as a result, a slight worsening in the overall $\mathcal{Q}$ metric representation may occur.

7.5.1 Grid-in-a-Grid

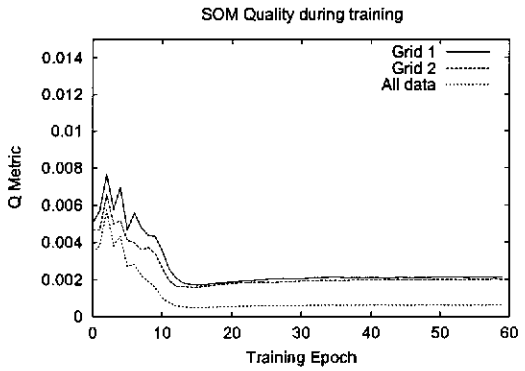
In the first experiment, a Grid-in-a-Grid data set (see Section A.5) was used. This data set consisted of a 3×3 major grid and a 3×3 minor grid. This data set consists of four obvious dimensions, as each grid possesses two physical dimensions. Each training vector consists of 18 inputs, composed by concatenating two 9 input patterns, each of which represents a 2D grid. The training syllabi were created by masking the respective grid components out of the training vector.

Three trials were performed. In the first trial, Grid 1 (inputs 1-9) was used as a syllabus, while Grid 2 (inputs 10-18) was masked. In the second trial, Grid 2 was used as a syllabus while Grid 1 was masked. For the purposes of comparison, a third trial was conducted, in which no masking was performed. Each trial was performed 10 times. $\mathcal{Q}$ metric performance was averaged over these runs. A test set of 500 vectors was used for each data subset.

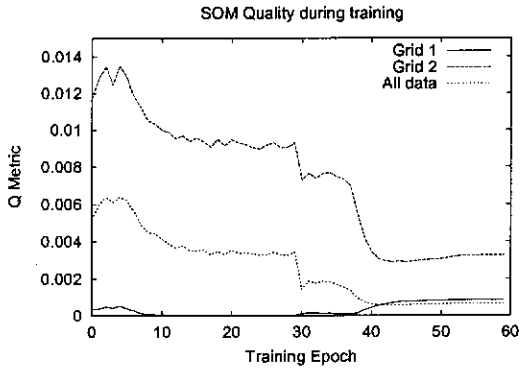
The Grid-in-a-Grid data set was trained on a 25×25 neuron map. Training was divided into two periods: a first period of 30 epochs, during which the syllabus data set was used; and a second period of 30 epochs, during which the full data set was used.

In the first two trials, Butterworth step decay was used to decay neighbourhood size. During the first period, neighbourhood size was decayed from 13 to 8, with a decay period of 10 epochs. During the second period, neighbourhood size was decayed from 8 to 2 with a decay period of 10 epochs. In the third trial, Butterworth step decay was also used. However, no intermediate step was used. Neighbourhood size was decayed in a single step from 13 to 2, over a decay period of 10 epochs. Gain was kept at a constant value of 0.2 during all experiments.

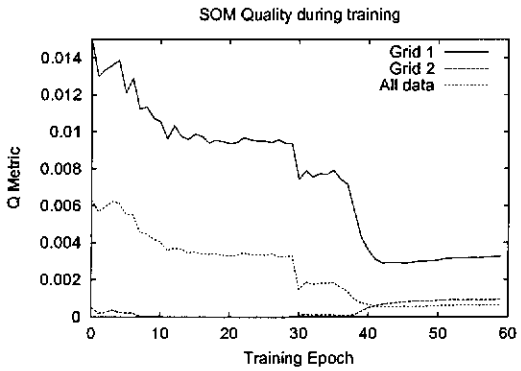
The results of this experiment can be seen in Figures 7.2 and 7.3. Figure 7.2 shows the average $\mathcal{Q}$ metric performance of each data subset during each of the three trials. Each plot shows three $\mathcal{Q}$ metric traces: the $\mathcal{Q}$ metric for the full data set, and the $\mathcal{Q}$ metric for each of the two data subsets used as a syllabus. Without the benefit of a



(a) No syllabus



(b) Syllabus consisting of grid 1 only

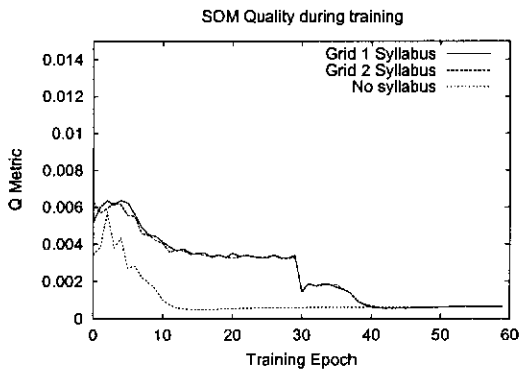


(c) Syllabus consisting of grid 2 only

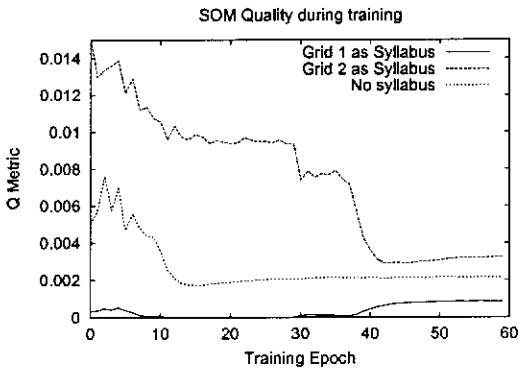
Figure 7.2: Three experiments in percept masking, using a Grid-in-a-Grid. Each plot shows the results of training using the specified syllabus; each line shows the $\mathcal{Q}$ metric for that subset of the data set.

syllabus (Figure 7.2(a)), the grid-in-a-grid data set quickly converges to a $\mathcal{Q}$ metric value of 0.0006. The two data subsets which compose the full data set converge a $\mathcal{Q}$ metric value of 0.002. This demonstrates that, in the absence of any sophisticated training mechanism, both subsets of the full data set are treated equally.

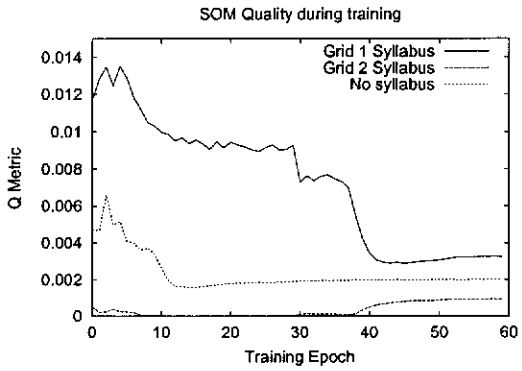
However, this behaviour can be altered by the introduction of a syllabus. Figure 7.2(b) shows the results of training using data from Grid 1 as a syllabus. During the first phase of training, the $\mathcal{Q}$ metric value for the Grid 1 data subset converges to almost 0, indicating a near perfect mapping. Conversely, the representation of Grid 2 improves very little from the $\mathcal{Q}$ metric resulting from initially random weights. At the end of the first phase of training, the representation of Grid 2 reaches an asymptotic value of 0.009. The combination of an extremely good representation of Grid 1 and an



(a) Full data set



(b) Grid 1



(c) Grid 2

Figure 7.3: Comparative performance of percept masking on a Grid-in-a-Grid. Each plot shows the comparative $\mathcal{Q}$ metric for the specified data subsets in each of the experiments from Figure 7.2.

extremely poor representation of Grid 2 results in an overall $\mathcal{Q}$ metricvalue of 0.003.

During the second phase of training, the data from Grid 2 is introduced to the training set. Not surprisingly, the representation of Grid 2 improves significantly, reaching an asymptotic $\mathcal{Q}$ metric value of 0.003. This improvement in representation of Grid 2 comes at the detriment of the representation for Grid 1. As a result, the $\mathcal{Q}$ metric value for Grid 1 increases to 0.0008. The overall representation is also observed to improve, as the overall $\mathcal{Q}$ metric value improves to approximately 0.0006. Analogous results are obtained if the syllabus consisting of Grid 2 is used (Figure 7.2(c)).

Figure 7.3 compares the performance of the masking technique on the three data sets. In each of the three trials, the full data set (Figure 7.3(a)) converges to a $\mathcal{Q}$ metric value of approximately 0.0006 — that is, the overall performance of the training

system is unaffected by the syllabus training. However, the representations of specific data subsets are strongly affected by syllabus training (Figures 7.3(b) and 7.3(c)). In the absence of a syllabus, a $\mathcal{Q}$ metric value of 0.002 can be achieved. If the syllabus favours the data subset, an improved $\mathcal{Q}$ metric value of 0.0008 is observed. On the other hand, if the syllabus favours the other data subset, representation of the subset decays to a $\mathcal{Q}$ metric value of 0.003. These results are identical for both data subsets.

7.5.2 Questionnaire

In the second experiment, a data set consisting of a 4 question Questionnaire (see Section A.4) was used. This data set consists of four obvious dimensions; each question represents a dimension in the data set.

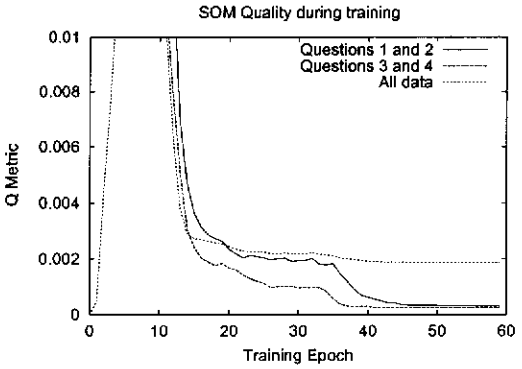
Three trials were performed. In the first trial, questions 1 and 2 were used as a syllabus, while questions 3 and 4 were masked. In the second trial, questions 3 and 4 were used as a syllabus, while questions 1 and 2 were masked. Since the data set is rotationally invariant, these results could be extrapolated to any two dimensional subset of the 4 question questionnaire. For the purposes of comparison, a third trial was conducted, in which no masking was performed. Each trial was performed 10 times. $\mathcal{Q}$ metric performance was averaged over these 10 runs.

The Questionnaire data set was trained on a 13×13 neuron map. Training was divided into two periods: a first period of 30 epochs, during which the syllabus data set was used; and a second period of 30 epochs, during which the full data set was used.

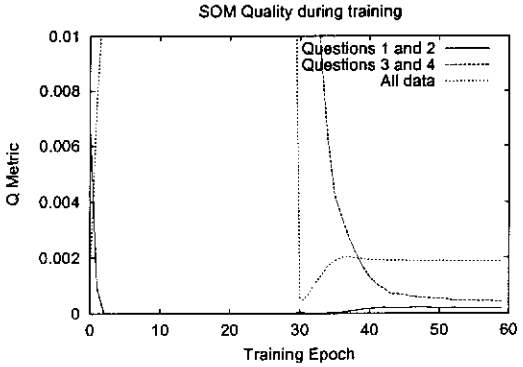
In the first two trials, Butterworth step decay was used to decay neighbourhood size. During the first period, neighbourhood size was decayed from 9 to 4, with a decay period of 10 epochs. During the second period, neighbourhood size was decayed from 4 to 2 with a decay period of 10 epochs. In the third trial, Butterworth step decay was also used. However, no intermediate step was used. Neighbourhood size was decayed in a single step from 9 to 2, over a decay period of 10 epochs. Gain was kept at a constant value of 0.2 during all experiments.

The results of this experiment can be seen in Figures 7.4 and 7.5. These plots have been trimmed in the vertical axis to preserve resolution in the final asymptotic values.

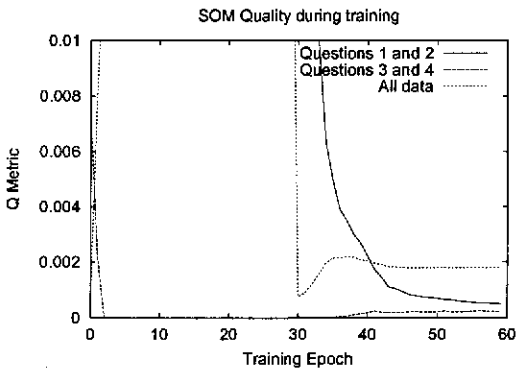
Figure 7.4 shows the average $\mathcal{Q}$ metric performance of each data subset during each of the three trials. Each plot shows three $\mathcal{Q}$ metric traces: the $\mathcal{Q}$ metric for the full data set, and the $\mathcal{Q}$ metric for each of the two data subsets used as a syllabus. Without the benefit of a syllabus (Figure 7.4(a)), the questionnaire data set quickly converges



(a) No syllabus



(b) Syllabus consisting of questions 1 and 2

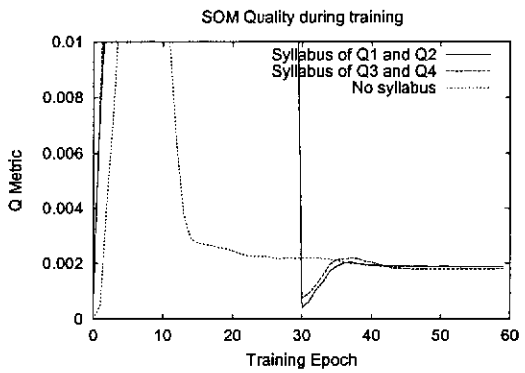


(c) Syllabus consisting of questions 3 and 4

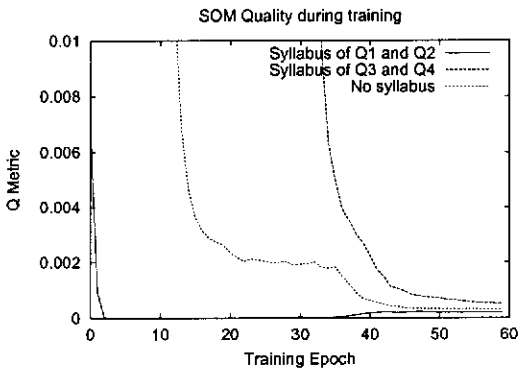
Figure 7.4: Three experiments in percept masking, using a 4 question Questionnaire. Each plot shows the results of training using the specified syllabus; each line shows the $\mathcal{Q}$ metric for that subset of the data set.

to a $\mathcal{Q}$ metric value of 0.0018. The two data subsets which compose the full data set converge to a $\mathcal{Q}$ metric value of approximately 0.0003. This demonstrates that, in the absence of any sophisticated training mechanism, all components of a data set are treated equally.

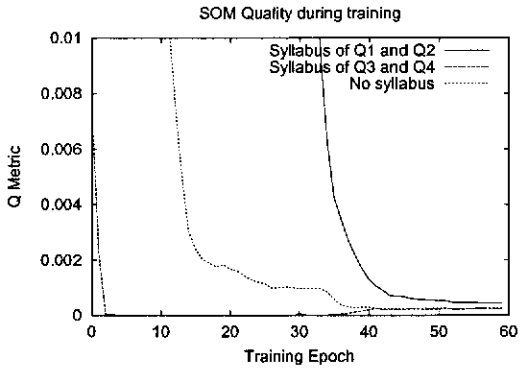
However, this behaviour can be altered by the introduction of a syllabus. Figure 7.4(b) shows the results of training using Questions 1 and 2 as a syllabus. During the first phase of training, the $\mathcal{Q}$ metric value for the first pair of questions converges to almost 0, indicating a near perfect mapping. Conversely, the representation of Questions 3 and 4 improves very little from the initial $\mathcal{Q}$ metric value, which is the result of the initially random weights. At the end of the first phase of training, the representation of the second pair of questions reaches an asymptotic value of 0.03. The



(a) Full data set



(b) Data set containing Questions 1 and 2



(c) Data set containing Questions 3 and 4

Figure 7.5: Comparative performance of percept masking on a 4 question Questionnaire. Each plot shows the comparative $\mathcal{Q}$ metric for the specified data subsets in each of the experiments from Figures 7.4.

combination of an extremely good representation of the first pair of questions and an extremely poor representation of the second pair results in an overall $\mathcal{Q}$ metric value of 0.017.

During the second phase of training, the second pair of questions is introduced to the training set. Not surprisingly, the representation of these questions improves significantly, reaching an asymptotic $\mathcal{Q}$ metric value of 0.0004. This improvement in representation comes at the detriment of the representation for the first pair of questions. As a result, the $\mathcal{Q}$ metric value for the first pair increases to 0.0002. The overall representation is also observed to improve, with the overall $\mathcal{Q}$ metric value improving to 0.0018. Analogous results are obtained if the second two questions are used as a syllabus (Figure 7.4(c)).

Figure 7.5 compares the performance of the masking technique on the three data sets. In each of the three trials, the full data set (Figure 7.5(a)) converges to a $\mathcal{Q}$ metric value of approximately 0.0018 — that is, the overall performance of the training system is unaffected by the syllabus training. However, the representations of specific data subsets are strongly affected by syllabus training (Figures 7.5(b) and 7.5(c)). In the absence of a syllabus, a $\mathcal{Q}$ metric value of 0.0003 can be achieved. If the syllabus favours the data subset, an improved $\mathcal{Q}$ metric value of 0.0002 is observed. On the other hand, if the syllabus favours the other data subset, representation of the subset decays to a $\mathcal{Q}$ metric value of 0.0004. These results are identical for each data subset.

One characteristic common to these plots are the extremely low values for the $\mathcal{Q}$ metric at the start of training with the syllabus, and after the introduction of the full training set. This is observed as a spike at epoch 30 in the $\mathcal{Q}$ metric value for the full data set in Figures 7.4(c), 7.4(c), and 7.5(a). These low values are an artifact of the size of the test set. As discussed in Section 3.4.2, small test sets are prone to developing misleadingly low $\mathcal{Q}$ metric values at the start of training. The full questionnaire test set consists of 16 training vectors, and the syllabus contains only 4 vectors. Therefore, the low $\mathcal{Q}$ metric values should not be misinterpreted as a good topology which has been overlooked by the SOM learning process; they are short lived artifacts resulting from the small testing set.

7.5.3 Scotch Whisky

In the final experiment, the syllabus training technique was applied to the Scotch Whisky data set (see Section A.6). Each vector in this data set consists of 97 components; these components can be split into 10 categories (Colour, Nose, Body, Palette, Finish, Age, Distillery Score, Whisky Score, Alcohol Content, District). The first five of these categories are tasting characteristics; the second five are manufacturing characteristics. Although the underlying dimensionality of this data set is unclear, these 10 categories could be considered to define 10 dimensions of the training data. Each category is itself multidimensional, and so could be easily split into subcomponents. However, the 10 basic categories provide a convenient partitioning of the input space.

Four training syllabi were prepared:

Syllabus A containing Colour and Nose data;

Syllabus B containing Body, Palette and Age data;

Syllabus C containing Palette and Finish data; and

Syllabus D containing Distillery score, Whisky score, Alcohol content and District data.

These syllabi were chosen at random, not as a result of any formal selection process or prior knowledge about the data.

Five trials were performed. In the first four trials, a syllabus was used to train the data, before the introduction of the complete data set. For the purposes of comparison, a fifth trial was conducted, in which no masking was performed. Each trial was performed 10 times. $\mathcal{Q}$ metric performance was averaged over these 10 runs.

The Scotch Whisky data set was trained on a 25×25 neuron map. Training was divided into two periods; a first period of 30 epochs, during which a syllabus data set was used, and a second period of 30 epochs, during which the full data set was used. Butterworth step decay was used to decay neighbourhood size. However, no intermediate step was used. Neighbourhood size was decayed in a single step from 18 to 2, over a decay period of 10 epochs. No intermediate step was required due to the large number of unique training examples in the syllabus training data. Gain was kept at a constant value of 0.2 during all experiments.

The results of this experiment can be seen in Figures 7.6 and 7.7. Figure 7.6 shows the average $\mathcal{Q}$ metric performance for each data subset during each of the five trials. As with previous experiments, components generate approximately equivalent representations if there is no training syllabus (7.6(e)). The only exception to this behaviour is the representation of Syllabus D; the reasons for this will be discussed shortly.

Training with Syllabi A, B and C (Figures 7.6(a)– 7.6(c)) also yield results similar to those observed in previous experiments. During these trials, the data subset used as a syllabus achieves a $\mathcal{Q}$ metric value better than that observed for other data subsets (and the full data set). After the introduction of the full data set at epoch 30, a minor performance decrease is observed in the representation of the data subset used as a syllabus, while minor performance improvements are observed in the representations of the other data subsets and the full data set.

One interesting feature of these results is the fact that the degree of improvement is not especially significant. In the Grid-in-a-Grid and Questionnaire experiments, there was a substantial difference between the quality of representations when using a syllabus. For example, using Grid 1 as a syllabus substantially impacted upon the

representation of Grid 2. However, in these experiments, the use of a syllabus does not give a significant representational advantage to the data represented in that syllabus. That is, using Syllabus A does not have a major effect upon the representation of data in Syllabus B.

This reveals an interesting characteristic of the Scotch Whisky data set; each of the tasting characteristics are closely related. Although the tasting characteristics are designed to represent independent characteristics of a whisky, this analysis suggests that there are close relationships between the tasting characteristics of a whisky. Although 'Yellow' colour and 'Warm' finish are intended to be independent binary characteristics, there is an underlying correlation between these (and other) features. This implies that the dimensionality of the tasting characteristic data space is much less than the 68 dimensions used to define the space, and that the underlying basis is not orthogonal to these tasting characteristics.

Furthermore, the manufacturing characteristics would appear to be somewhat dependent upon the tasting characteristics. Training using Syllabus A, B or C does not affect the representation of Syllabus D, which contains only manufacturing characteristics. This would only occur if there was some similarity between the tasting topology and the manufacturing topology. This implies that the tasting characteristics could be used to infer certain manufacturing characteristics (although the nature of this implication cannot be easily derived from these maps).

The inverse, however, is not true. Training with Syllabus D (Figure 7.6(d)) enforces a principal topology based entirely upon manufacturing characteristics. However, this fundamental topology does not allow good representations of the other data subsets. Syllabi A, B, C, and the full data set all develop poor representations before the introduction of the full data set. Elements of the topology of manufacturing characteristics support the topology of tasting characteristics, but the dominant topology of manufacturing characteristics does not support the tasting topology. This suggests that although tasting characteristics can be used to imply manufacturing characteristics, manufacturing characteristics cannot be used to imply tasting characteristics. This represents an interesting asymmetry in the data set.

This asymmetry also explains the high Q metric value observed for Syllabus D when no syllabus is used (Figure 7.6(e)). The topology underlying the manufacturing data is evidently weak, and not supported by the tasting data. When no syllabus is used, and the tasting data is equally represented on the map, the manufacturing data develops a

poor representation on the SOM.

Figure 7.7 compares the performance of the masking technique on the three data sets. The results observed in this experiment are similar to those observed with the grid-in-a-grid and questionnaire experiments. Syllabi A, B, C and D (Figures 7.7(a), 7.7(b), 7.7(c), and 7.7(d)) all perform as expected. Performance with a supporting syllabus is better than performance without a syllabus, which is better still than performance with a non-supporting syllabus. However, the improvements are not quite as extreme.

The most significant difference between the Scotch Whisky experiment and the previous experiment is the comparative performance of the whole data set. In previous experiments, no difference in overall performance was observed; the asymptotic value of the $\mathcal{Q}$ metric was unaffected by the choice of syllabus, and was consistent with performance achieved without the use of a syllabus. In this experiment, the same $\mathcal{Q}$ metric value is achieved using each syllabus. However, the performance achieved without the use of a syllabus is slightly better than the with-syllabus value. This implies that the syllabi used in this experiment are not representative of the underlying topology of the data set.

It is interesting to note that Syllabus B (Figures 7.7(b)), which is comprised of two tasting characteristics (Body and Palette) and one manufacturing characteristic (Age), gives the most consistent results. This suggests that the strongest underlying topology of the Scotch Whisky data set is comprised of both manufacturing and tasting characteristics. This supports the hypothesis that the syllabi used in this experiment are not representative of the data space; another hypothesis on the structure of the data should be made.

7.5.4 Discussion

The results presented in this section demonstrate that the syllabus presentation of a training set generated by percept masking can be used to control the development of topologies on a Kohonen-style SOM. If the chosen syllabus is a good representation of a key component of the underlying topology, specific topologies can be preferentially trained onto a SOM, improving the quality of representation of specific components of the training set at the detriment of other components. If the syllabus is well selected, this does not affect the overall quality of the representation, only the representation of individual components.

The syllabus presentation and percept masking techniques can be used in one of two

ways. Firstly, if the user has strong prior knowledge, the user can use this knowledge to enforce a specific topology on the data set. By selecting a syllabus which represents the known underlying topology, it is possible to encourage this topology to form the fundamental topology of the SOM. After the reduction in neighbourhood size, this fundamental topology will be difficult to override, as training with a small neighbourhood size has limited capacity to alter global organisation. The global topology will be preserved without the need to present supporting training examples. This ensures that the prior knowledge is retained by the SOM.

Secondly, the syllabus presentation technique can be used as a mechanism for topology discovery. The first two experiments in this chapter demonstrated that syllabus presentation can be a useful technique if good domain knowledge is available. However, this will not always be the case. In many real-world problems (such as the Scotch Whisky data set), it is difficult to acquire *a-priori* knowledge about a data domain. If there is an absence of good ground knowledge, syllabus presentation can be used to test various hypotheses regarding the topological independence of input components. If two input components are independent, syllabus training with the first component will generate a poor representation of the second component, and vice versa. This was demonstrated with the Scotch data. The $\mathcal{Q}$ metric results from this experiment (Figure 7.6) revealed that the tasting characteristics are not completely independent.

The use of syllabus presentation does require a loss of flexibility. By enforcing a specific topology, one loses the ability to represent an arbitrary topology — possibly a topology which would be a better representation of the overall problem space. The syllabus presentation technique should only be used if there is strong background knowledge and good reason to suggest that this background knowledge should form the dominant topology of a map.

However, syllabus presentation can be used to overcome one of the fundamental inflexibilities associated with SOMs. The output representation of a SOM is a 2D map. When training an n-dimensional data space, $n > 2$, approximately n-2 dimensions must be lost. Traditional training provides no method for specifying which of these dimensions will be lost — the fundamental topology is determined at random through the random weight instantiations. However, by using syllabus training, it is possible to force specific pairs of dimensions to form the fundamental topology. This deterministic approach to topology selection allows the user to generate of a family of SOMs, each of which visualises a specific 2D projection of the data space. An n-dimensional data space

would require a family of ${}^nC_2$ maps, each representing a unique pair of dimensions[1]. This overcomes one of the major representational issues associated with SOMs: the representation of high dimensionality data spaces.

The experiments presented in this chapter have shown that percept masking can be used to generate a useful training syllabus. However, this technique does make one significant assumption: that the underlying topology of a data set is orthogonal to the input components. The Scotch Whisky experiment demonstrated that this need not be the case. Complex data sets will exhibit complex topological relationships which are not orthogonal to the input components. In these problems, a simple binary masking technique will not produce the strongest possible fundamental topology. To represent these topologies, a more complex mask would be required.

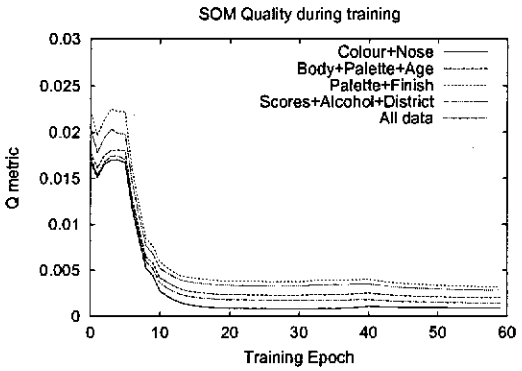
One possible mechanism for generating such a mask would be the use of principal components or Hotelling transform to filter complex topological relationships out of mixed data. Whereas simple percept masking represents a projection of a full training vector onto a subspace orthogonal to the input space, a Hotelling transform would allow projection onto a non-orthogonal subspace, allowing the representation of topologies across inputs. The development of such a technique is left as future work.

7.6 Conclusion

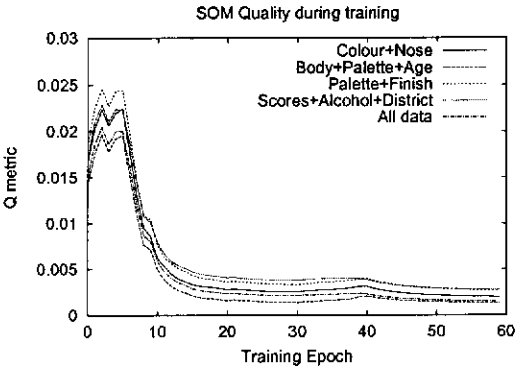
In this chapter, the concept of syllabus presentation was introduced. This technique uses a simplified training set to ensure the development of key knowledge representations. A technique was also presented for developing a simplified training set suitable for use in syllabus presentation. This technique, called percept masking, uses an orthogonal projection of the training vector onto a subspace of the full input space to simplify an initial training set. Using these techniques, given prior knowledge about a training set and careful selection of masked percepts, it is possible to encourage a specific SOM topology to develop. These techniques can also be used for exploration of a complex topology if little is known about.

The syllabus presentation and percept masking techniques allow core knowledge to be established in the fundamental topology of a SOM. This limits the extent to which these core concepts can be forgotten. However, this does not address the forgetfulness of concepts introduced in later training. This issue will be addressed in the next chapter.

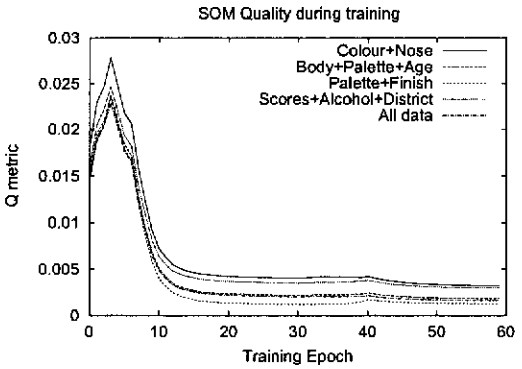
[1] This approach is similar to observing 2D projections of an n-dimensional hypermap.



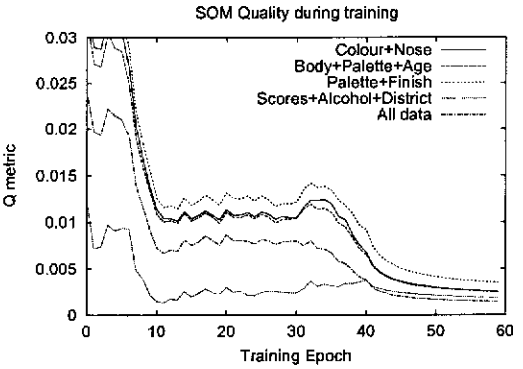
(a) Syllabus A



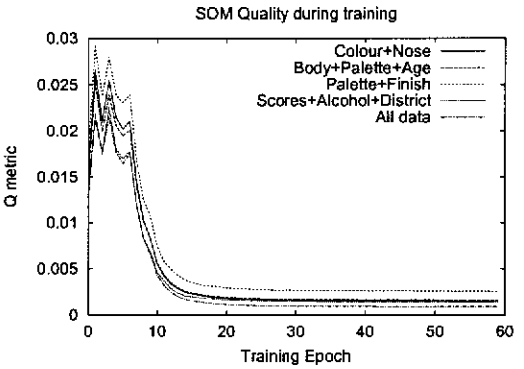
(b) Syllabus B



(c) Syllabus C

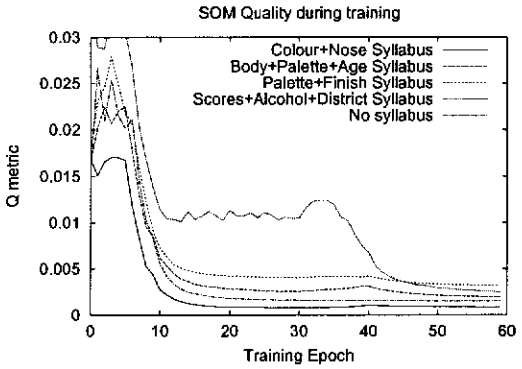


(d) Syllabus D

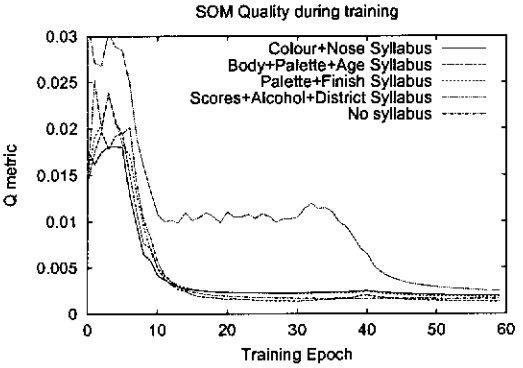


(e) No syllabus

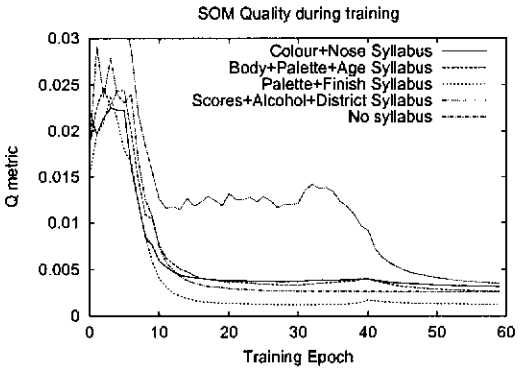
Figure 7.6: Three experiments in percept masking, using the Scotch Whisky data set. Each plot shows the results of training using the specified syllabus; each line shows the Q metric for that subset of the data set.



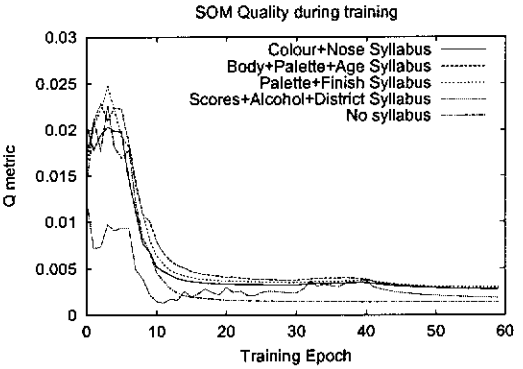
(a) Data set containing Syllabus A



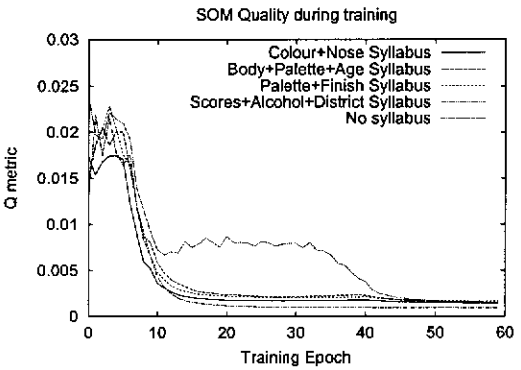
(b) Data set containing Syllabus B



(c) Data set containing Syllabus C



(d) Data set containing Syllabus D



(e) Full data set

Figure 7.7: Comparative performance of percept masking on the Scotch Whisky data set. Each plot shows the comparative $\mathcal{Q}$ metric for the specified data subsets in each of the experiments from Figure 7.6.

Chapter 8

Arbor Pruning

> Murmurs of alarm came from the crowd. The management consultant waved them down.
>
> 'So in order to obviate this problem,' he continued, 'and effectively revalue the leaf, we are about to embark on a massive defoliation campaign, and ... er, burn down all the forests. I think you'll agree that's a sensible move under the circumstances.'
>
> — *The Restaurant at the End of the Universe* (Adams, 1980)

8.1 Introduction

Using Butterworth Step Decay, neighbourhood size does not decay to 0 at the end of training, resulting in a SOM which retains the ability to update. As a result, the SOM remains capable of responding to changes in input data. If the training set presented to the SOM changes over time, the SOM becomes a dynamic model of the input data set at any given time — that is, the SOM can be made to learn continuously.

The continuously learning SOM algorithm does suffer from one major drawback: the problem of forgetfulness. The SOM algorithm effectively forms a probability density map of the input set. The area given over to a concept on a trained SOM is directly related to the frequency of the training patterns representing that concept in the training set. As the SOM adapts to a dynamic data set, the probability density map provided by the SOM will, at any given time, represent only the most recent training examples, not the range of data presented over the entire history of training. Training examples which are not repeated will, over time, lose their representation in the map, regardless of the significance of these training examples. That is, the continuously learning SOM has no long-term memory.

In this chapter, we propose one method for realising a reduction in forgetfulness in continuously learning SOMs. This technique is known as *Arbor Pruning*. This tech-

nique involves introducing a key restriction to the update stage of the SOM learning algorithm, thereby restricting the ability of the SOM to adapt to represent novel input patterns. These restrictions are introduced after an initial topology has been learned, enabling the preservation of an existing topology while remaining plastic to the introduction of new training data. The arbor pruning technique is introduced in Section 8.2.1, and is tested empirically in Section 8.3 using the Simple Grid, Questionnaire, and Scotch Whisky data sets.

8.2 Overcoming Forgetfulness

The problem of forgetfulness was demonstrated empirically in Chapter 6. The experiments presented in that chapter clearly demonstrate the two extremes of the stability-plasticity dilemma. Using linear neighbourhood decay, a SOM is completely static at the completion of training. No new knowledge can be integrated into the SOM once the neighbourhood size has decayed to 0. Using Butterworth Step Decay, the SOM is extremely plastic, and therefore remains responsive to new training examples. This training could potentially continue indefinitely. However, using Butterworth Step Decay, the long term retention of trained knowledge cannot be guaranteed without reinforcement. If training patterns are not reinforced, they eventually lose their representation on the map.

This represents a major limitation to the usefulness of a SOM for continuous learning purposes. While a continuous learning mechanism capable of short-term, forgetful memory may be useful for some applications, long term memory also has significant uses. Applications which need to learn from experience cannot afford to lose the benefit of significant training examples simply because they occurred early in training and have not been repeated recently. To be of general use, a method must be found to overcome this forgetfulness problem.

One method for overcoming forgetfulness in SOMs is to exploit the weight relationships on a map. Kohonen's SOM algorithm uses a fully arborised network of neurons. Each and every input is connected to each and every SOM neuron by a single weight. This complete arborisation strategy gives Kohonen's SOM algorithm remarkable flexibility in representation, as every input has an equivalent potential to affect each and every neuron on the map. By restricting the input arborisation, it is possible to control the ability of the SOM to represent certain topologies. If certain inputs are prevented from influencing a map neuron, that map neuron will not undergo Hebbian reinforce-

ment to support that input.

However, this leaves the important question of which weights to restrict, and by how much. In addition, we wish the outcome of restricting map update to yield a specific goal: that of realising a reduction in forgetfulness as new training patterns are presented to the map. In order to achieve this specific goal through weight restriction, an informed strategy for weight restriction is required.

One possible strategy can be obtained by examining the significance of weight values in a trained maximal SOM. In maximal SOMs, significant concepts are awarded strong weights. Weak concepts, however, have small weights and therefore contribute little to concept representation on the map. If a weight contributes little to the concept representation on the map, it makes little difference whether the weight is present or not. This fact can be exploited for the purposes of reducing forgetfulness. By controlling these weak weights, it should be possible to prevent the loss of existing patterns, and restrict the development of strongly competing patterns.

It is important to note that this technique is not possible on minimal SOMs (the more common form of Kohonen-style SOM, described in Section 2.5.1), as there is no direct analogue between weight strength and importance. Minimal SOMs attempt to minimize the distance between weight and input pattern. Therefore, the significance of a given weight cannot be determined without the use of a reference training pattern. Maximal SOMs, by contrast, do not require such a reference. A large weight indicates significance, regardless of any specific training instance.

8.2.1 Arbor Pruning

The arbor pruning technique presented in this chapter exploits this relationship between weight strength and the significance of that weight to concept representation. The arbor pruning algorithm is fundamentally simple. At the time pruning is to take place, a threshold value ϕ is chosen. At every subsequent Hebbian update of weights, the update of weights in the map which do not exceed this threshold is prevented. All other aspects of map operation are unchanged. The changes to the update portion of the algorithm are presented in full in Algorithm 8.

If the training set remains static, nothing is affected. Strong weights become stronger, and the weak weights (which contributed little to the selection of winners in the first place) are no longer reinforced. However, if the training set changes, the removal of weak weights from the map restrict the ability of the map to alter to rep-

Algorithm 8 The update portion of the arbor pruning algorithm. This procedure replaces the update function of Algorithm 4. All other aspects of the SOM algorithm are the same as the original maximal SOM algorithm.

Update_withPruning ($\mathbf{W}$,$\vec{x}$,α,σ,ϕ)
for i = 1 to **I do**
  $\Theta_i = \vec{w}_i \cdot \vec{x}$
end for
$\omega = \max_i \Theta_i$
for i = 1 to **I do**
  for j = 1 to **J do**
    if $w_{ij} > \phi$ **then**
      $w_{ij} = w_{ij} + [\alpha h_{i,\omega}(\vec{x} - \Theta_i \vec{w}_i)]_j$
    end if
  end for
end for

resent these changes. As a result, the map will resist (or completely forbid) the loss of known concepts which previously held significance in the training set.

The arbor pruning technique is applied to a map at the end of initial training. A map is fully trained using Butterworth Step Decay. This initial training period can be thought of as juvenile development, where the parent (or in this case, user) of the map carefully trains the juvenile, exposing it to appropriate and extensive stimuli. At the end of this initial training, weights which do not exceed a threshold ϕ are removed from the trained map. Using the neighbourhood size and learning gain from the end of the juvenile training period, the map can then learn continuously without significant forgetting.

8.3 Testing Arbor Pruning

In this section, an experiment is presented which demonstrates the way in which the Butterworth Step Decay mechanism can be used to implement a scheme of continuous learning. This experiment is repeated three times, using three different training sets: the Simple Grid data set (see Section A.3); the Questionnaire data set (see Section A.4); and the Scotch Whisky data set (see Section A.6). This experiment is identical in spirit to that performed in Chapter 6. However, in order to prevent the forgetfulness observed in that chapter, the arbor pruning algorithm will be implemented.

In order to demonstrate continuous learning without forgetting, each data set is split into two overlapping subsets. The first subset is presented to the SOM and learned completely. At the completion of this initial phase of learning, the second

subset is presented to the SOM. This new data set contains some familiar training examples and some new examples from the same domain. A continuously learning system should incorporate the new training examples into the topology established in the initial training phase.

However, the incorporation of these new training examples cannot be at the cost of already learned training examples. To prevent the loss of representation of training examples presented in the initial phase of learning, the arbor pruning scheme is implemented during the second phase of learning. A range of arbor pruning threshold values are tested, to demonstrate the way in which the level of forgetfulness can be controlled by controlling the threshold parameter.

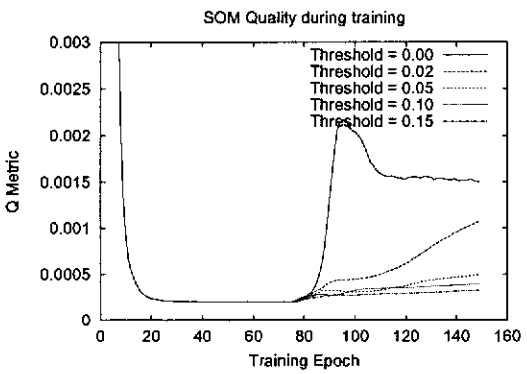
8.3.1 Simple Grid

In the first experiment, a Simple Grid data set (see Section A.3) is used. Change in the source data is achieved using the same technique described in Section 6.3.1. Using this technique, a 7×11 rectangular simple grid data set was generated, in order to simulate a data set shift of 4 units. The two 7×7 subsets were drawn from this rectangular data set as training subsets. Using these two subsets of the original data set it is possible to test the response to new data in the data source (the columns of data in subset 2 which are not in subset 1) and the response to data disappearing from the data source (the columns of data in subset 1 which are not in subset 2). This change in data source represents a change in domain along established dimensions in the data source.

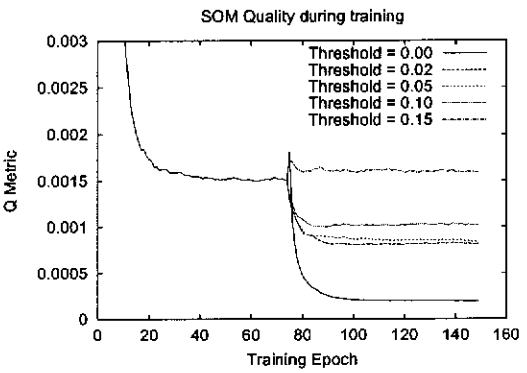
These data subsets were trained on a 13×13 neuron map. The first period of training lasted for 75 epochs, using data subset 1. After this initial period of training, data subset 2 was used for a further 75 epochs of training. Neighbourhood size is decayed from 9 to 2, using the Butterworth Step method, with the step occurring after 10 epochs. Learning gain is kept at a constant value of 0.2 in all experiments.

The arbor pruning algorithm was brought into effect at epoch 75 of each trial. To demonstrate the effect of the arbor pruning algorithm, trials utilising pruning thresholds of 0.0, 0.02, 0.05, 0.1, and 0.15 were performed. This range of pruning thresholds represents a range of pruning levels from no pruning, through to almost complete pruning.

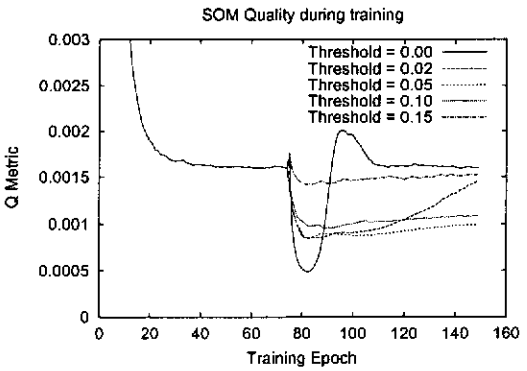
To eliminate statistical variation, each experiment was performed 10 times, and the $\mathcal{Q}$ metric for each trial averaged. The value of each $\mathcal{Q}$ metric was based upon a randomly selected set of 500 test points from within the domain of the relevant data



(a) Data Subset 1



(b) Data Subset 2



(c) Full Data Set

Figure 8.1: $\mathcal{Q}$ metric during continuous learning with arbor pruning of a Simple Grid data set, using a range of pruning thresholds. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 2 is formed by shifting subset 1 by 4 units. The full data set is the union of these two subsets.

subsets.

The results of these trials are shown in Figure 8.1. Without the benefit of pruning (i.e., $\phi = 0.0$), the behaviour observed is identical to that observed in Chapter 6. Initial training with the data subset 1 results in a SOM which has a $\mathcal{Q}$ metric for this subset of 0.0002. The representation second subset asymptotes at the larger $\mathcal{Q}$ metric value of 0.0015. This data set is able to form a partial representation, as data subset 1 partially overlaps data subset 2. However, the quality of this representation is limited. The $\mathcal{Q}$ metric value of 0.0016 is observed for the combined training set.

During the second training period, training is performed using data subset 2. Without pruning, the representation of data subset 2 improves to a $\mathcal{Q}$ metric value of 0.0002. This improvement comes at the detriment of the representation of data subset 1, which

increases to a $\mathcal{Q}$ metric value of 0.0015. This represents a complete reversal of $\mathcal{Q}$ metric values for the two data subsets. The overall $\mathcal{Q}$ metric value is not affected, except during a short period after the change of training set. During this period (epochs 75–100), the entire data set is briefly represented on the SOM; this dual representation is lost as data set 2 is reinforced.

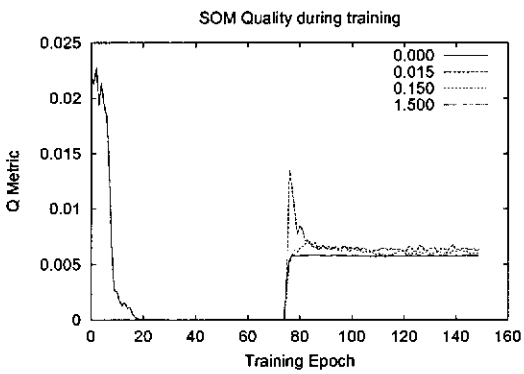
However, by slowly increasing the pruning threshold ϕ, it is possible to limit the loss of representation of data subset 1. Figure 8.1(a) shows that as ϕ is increased, a decrease in the extent of representational loss (i.e., a smaller increase in $\mathcal{Q}$ metric value) is observed, with a pruning threshold of 0.15 sufficient to prevent almost any loss. The opposite occurs to data subset 2. As ϕ is increased, the improvement in asymptotic $\mathcal{Q}$ metric value decreases. A pruning threshold of 0.15 is sufficient to prevent any improvement in representation of data subset 2. In fact, $\mathcal{Q}$ metric value for data subset 2 is observed to increase slightly.

Altering the pruning threshold has an interesting effect on the overall $\mathcal{Q}$ metric value. If the pruning threshold is set high (e.g., $\phi = 0.15$), no update to the topology can take place. Consequently, no change in overall $\mathcal{Q}$ metric value is observed. If the pruning threshold is set to 0, there is also no effect on the overall $\mathcal{Q}$ metric value. The first data subset is completely replaced by the second, with no overall gain. However, if the pruning threshold is set at a small value ($0.05 < \phi < 0.10$), it is possible to improve the overall $\mathcal{Q}$ metric value for the combined data set. In this case, the use of a pruning threshold prevents the complete loss of existing training cases, but allows for the introduction of new data; as a result, overall performance improves.

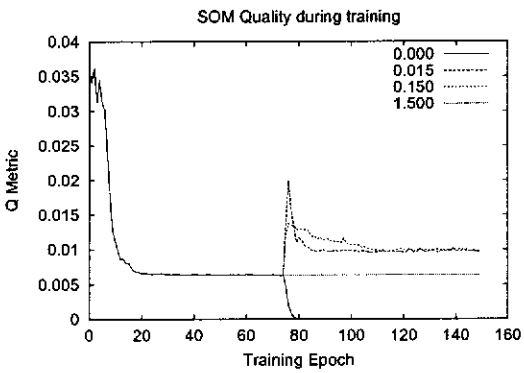
8.3.2 Questionnaire

In the second experiment, a Questionnaire data set (see Section A.4) is used. Changes in the source data are achieved in a similar manner to that used for the Simple Grid data set. The complete data set consists of 8 questions, split into two overlapping subsets of 5 questions each. This represents a shift of 3 questions between the two data subsets. The changing data source can be thought of as a questionnaire that changes over time. New questions are added, and old questions are removed. This represents a second type of shift in the data set; the introduction of new dimensions to the data source.

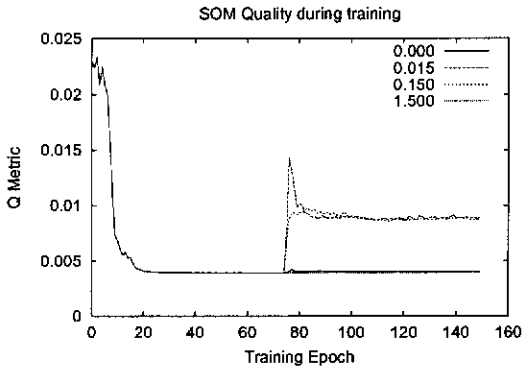
These data subsets were trained on a 15×15 neuron map. The first period of training lasted for 75 epochs, using data subset 1. After this initial period of training,



(a) Data Subset 1



(b) Data Subset 2



(c) Full Data Set

Figure 8.2: $\mathcal{Q}$ metric during continuous learning with arbor pruning of a Questionnaire data set, using a range of pruning thresholds. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. The full questionnaire contains 8 questions. Training subset 1 contains questions 1-5; training subset 2 contains questions 4-8.

data subset 2 was used for a further 75 epochs of training. Neighbourhood size was decayed from 11 to 2 using the Butterworth Step method, with the step occurring at 10 epochs. A constant learning gain of 0.2 was used in all experiments.

To demonstrate the effect of the arbor pruning algorithm, pruning thresholds of 0.0, 0.015, 0.15, and 1.5 were used. This range of pruning thresholds represents a range of pruning levels from no pruning, through to almost complete pruning. To eliminate statistical variation, each experiment was performed 10 times, and the $\mathcal{Q}$ metric for each trial was averaged.

The results of these trials can be found in Figure 8.2. The results which utilise a pruning threshold of 0.0 are identical to the results observed in Chapter 6. Initial

training with the data subset 1 results in a SOM which has a $\mathcal{Q}$ metric for this subset of 0.00001. The representation second subset asymptotes at the larger $\mathcal{Q}$ metric value of 0.006. This data set is able to form a partial representation, as data subset 1 partially overlaps data subset 2. However, the quality of this representation is limited. A $\mathcal{Q}$ metric value of 0.004 is observed for the combined training set.

During the second training period, training is performed using data subset 2. Without pruning, the representation of data subset 2 improves to a $\mathcal{Q}$ metric value of 0.00004. This improvement comes at the detriment of the representation of data subset 1, which increases to a $\mathcal{Q}$ metric value of 0.006. This represents a complete reversal of $\mathcal{Q}$ metric values for the two data subsets. The overall $\mathcal{Q}$ metric value is not affected.

Unfortunately, the introduction of pruning to this data set does not have the desired effect. The application of arbor pruning has very little effect on the representation of data subset 1 as the final asymptotic $\mathcal{Q}$ metric values are similar to those observed without arbor pruning. However, the results for data subset 2 experience a significant drop in performance. The application of arbor pruning at almost any threshold results in a significant drop in asymptotic $\mathcal{Q}$ metric value. The asymptotic value reached appears to be independent of the threshold value chosen. Similar result are obtained for the full data set. A significant and consistent loss in $\mathcal{Q}$ metric value is observed for almost all arborisation thresholds. This indicates that the application of arbor pruning has a significant detrimental performance on overall SOM performance.

The only exception to this behaviour is observed with extremely large pruning values ($\phi = 1.5$). These values correspond to a complete pruning of neuron arborisation. In this situation, no weights in the SOM are updated. As a result, the representation of data subset 1 is completely preserved and the acquisition of a representation of data subset 2 is completely prevented.

The reason for this failure is easily established. At the completion of the first phase of training, the weights connected to known questions (i.e., inputs which have seen non-zero values) have been updated. However, weights connected to unseen questions will have received no update. Given that the SOM is instantiated with small random values, these weights which will be pruned when arbor pruning is applied. Therefore, the weights which are connected to the novel inputs cannot be updated, preventing the SOM from learning any topologies represented by these inputs.

This does not render the arbor pruning algorithm useless. However, it does restrict the type of forgetting that can be prevented. Care must be taken to ensure that

all relevant inputs are represented in the initial training set before the arbor pruning technique is to be used.

8.3.3 Scotch Whisky

Finally, the use of the Butterworth Decay for continuous learning was tested on a real-world data set — an tasting analysis of Scotch Whisky (see Section A.6). This data set is drawn from expert tasting analysis of single malt Scotch Whisky. This data set does have a topology; however, it is not a regular or evenly distributed topology.

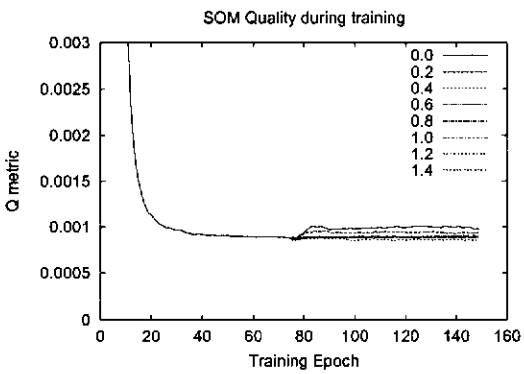
The subset method is again used to generate change in the source data. The construction of subsets of the Scotch Whisky data set is described in depth in Section 6.3.3. Chapter 7 established that the principal dimensions of the Scotch Whisky data set are not orthogonal to the input space; therefore, the problems encountered with the Questionnaire data set should not be encountered.

These data subsets were trained on a 25×25 neuron map. The first period of training lasted for 75 epochs, using the first data subset. After this initial period of training, the second data subset was used for a further 75 epochs of training. When using Butterworth neighbourhood decay, neighbourhood size is decayed from 18 to 2, with the step occurring at 10 epochs. Given the density of the training data, no intermediate step was required. In both Butterworth and linear decay experiments, a constant learning gain of 0.2 is used.

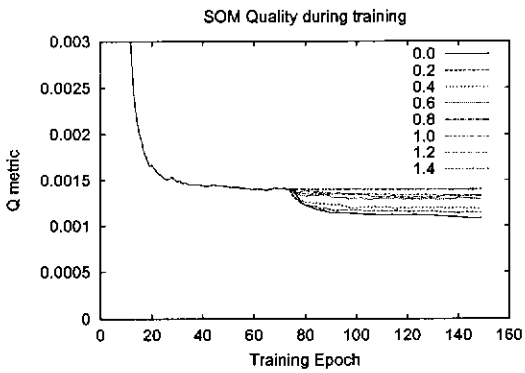
To demonstrate the effect of the arbor pruning algorithm, pruning thresholds from 0.0 to 1.4 in increments of 0.2 were used. This range of pruning thresholds represents a range of pruning levels from no pruning, through to almost complete pruning. To eliminate statistical variation, each experiment was performed 10 times and the $\mathcal{Q}$ metric for each trial averaged.

The results of these experiments can be found in Figures 8.3-8.6. Figure 8.3 is trained with a data set initially consisting of no Lowland whiskies; secondary training contains Lowland whiskies, but no Islay whiskies. Figure 8.5 is trained with a data set initially consisting of no Islay whiskies; secondary training contains Islay whiskies, but no Highland whiskies. Figure 8.6 is trained with a data set initially consisting of no Lowland whiskies; secondary training contains Lowland whiskies, but no Highland whiskies.

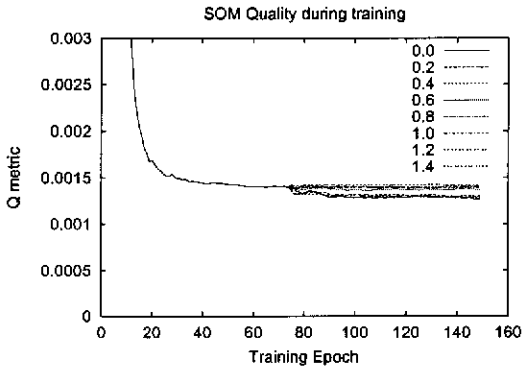
The results obtained in this experiment are comparable to those obtained when applying arbor pruning to the Simple Grid data set. During initial training, the repre-



(a) Data subset 1: No Lowland Whiskies



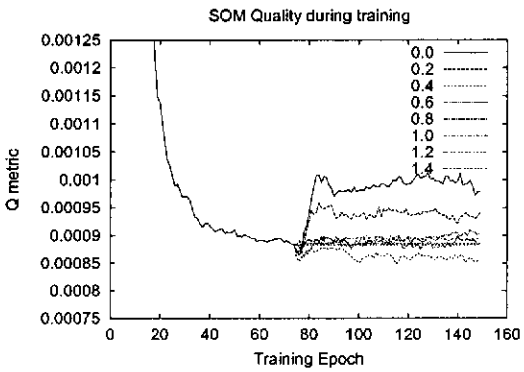
(b) Data subset 2: No Islay Whiskies



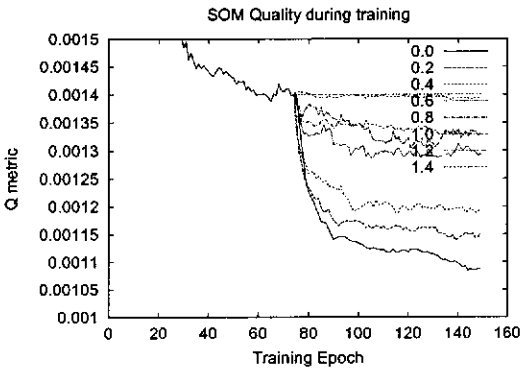
(c) Full data set: All Whiskies

Figure 8.3: $\mathcal{Q}$ metric during continuous learning with arbor pruning of the Scotch Whisky data set, using a range of pruning thresholds. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 1 contains no Lowland whiskies; training subset 2 contains no Islay whiskies. The full data set is the union of these two subsets.

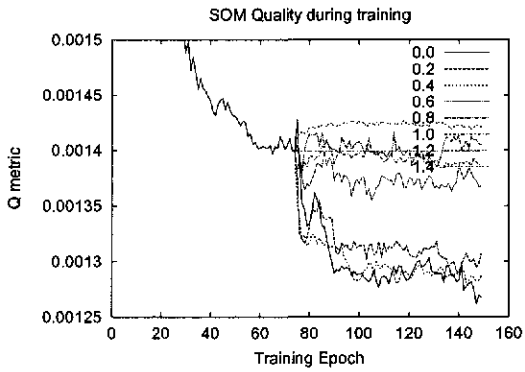
sentation for data subset 1 reaches a good representation, while data subset 2 reaches an asymptotic, but less adequate representation. During the second phase of training, data subset 2 is introduced. In the absence of pruning, the representation for data subset 1 is observed to decay, and the representation for data subset 2 is observed to improve. The extent of any changes in $\mathcal{Q}$ metric value is controlled by the size of the overlap between the training subsets. In the first experiment, there is significant overlap in the two data sets; consequently, there is very little improvement in the $\mathcal{Q}$ metric values after the introduction of data subset 2. The second two experiments exhibit very little overlap in training data, resulting in a dramatic increase in $\mathcal{Q}$ metric value after the introduction of data subset 2.



(a) Data subset 1: No Lowland Whiskies



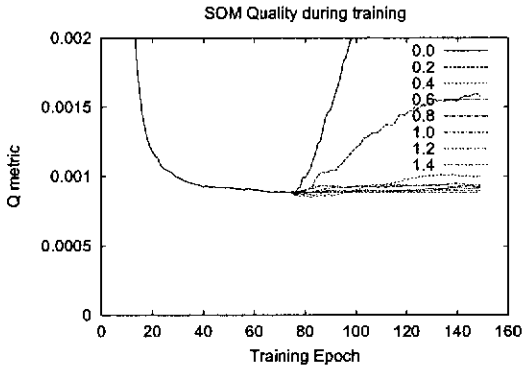
(b) Data subset 2: No Islay Whiskies



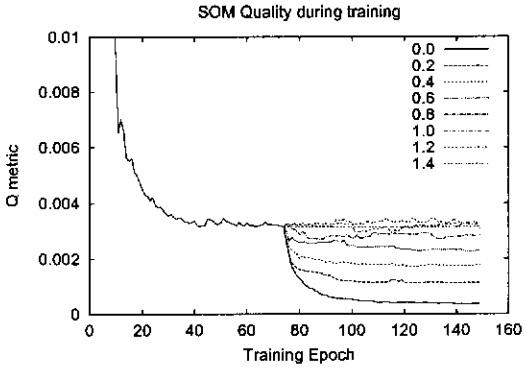
(c) Full data set: All Whiskies

Figure 8.4: Detail of Figure 8.3. These plot shows that although the difference between data subsets 1 and 2 is small, asymptotic values for the $\mathcal{Q}$ metric are still ordered by the value of the pruning threshold ϕ.

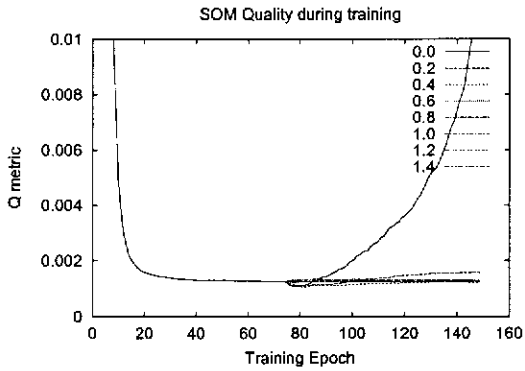
Introduction of pruning prevents this change. As the pruning threshold is increased, the extent of changes in representation are observed to decrease, as the representation of data set 2 improves less and the representation of data set 1 decays less. At a pruning threshold of approximately 1.2, the representation of the data subsets is virtually static, a result of a complete pruning of the weights. The arbor pruning algorithm is therefore able to control forgetfulness in this data set. This results can be clearly observed in Figures 8.5 and 8.6 as an ordered decrease in the asymptotic $\mathcal{Q}$ metric value for data subset 1, and corresponding decrease in the asymptotic $\mathcal{Q}$ metric values for the data subset 2. This behaviour is also observed in the first experiment (Figure 8.3); however, the small difference between data subsets makes the ordering difficult to observe. Figure 8.4 shows a detail of the asymptotic values, demonstrating that although the difference



(a) Data subset 1: No Islay Whiskies



(b) Data subset 2: No Highland Whiskies

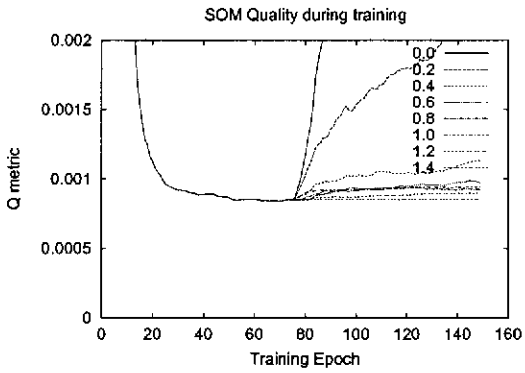


(c) Full data set: All Whiskies

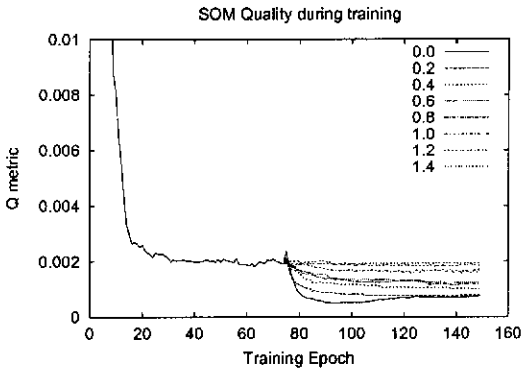
Figure 8.5: Q metric during continuous learning with arbor pruning of the Scotch Whisky data set, using a range of pruning thresholds. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 1 contains no Islay whiskies; training subset 2 contains no Highland whiskies. The full data set is the union of these two subsets.

is small, pruning can be used to control forgetfulness within this range.

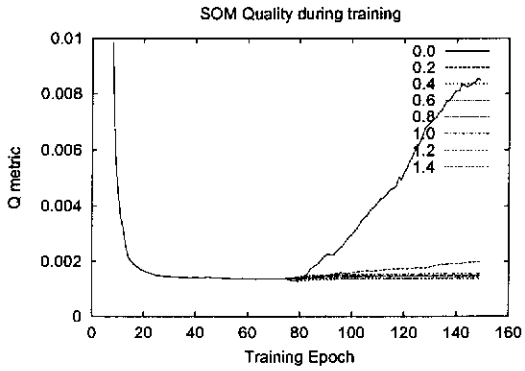
The representation of the full data set demonstrates some interesting results. As was discussed in the Scotch Whisky experiment of Chapter 8, the Q metric value of the full data set is affected by the change in training subset. This is the result of the lack of symmetry in size and density of the training subsets. In the first experiment (Figure 8.3, with detail in Figure 8.4), the two data subsets (No Lowland and No Islay) are approximately the same size. Without pruning, a small improvement in Q metric value is observed when data subset 2 is presented. Through the introduction of pruning, it is possible to control the extent of this overall improvement. As the threshold is increased, the Q metric value for the full data set increases until it reaches a level indicative of



(a) Data subset 1: No Lowland Whiskies



(b) Data subset 2: No Highland Whiskies



(c) Full data set: All Whiskies

Figure 8.6: $\mathcal{Q}$ metric during continuous learning with arbor pruning of the Scotch Whisky data set, using a range of pruning thresholds. Initial training is performed using data subset 1; after 75 epochs, training subset 2 is used. Training subset 1 contains no Lowland whiskies; training subset 2 contains no Highland whiskies. The full data set is the union of these two subsets.

no change from the state of the map at the end of the first phase of training.

Although it seems unusual to practice a technique which is detrimental to the overall representation of a data set, it is important to note that the improvement in overall representation comes at the cost of a decay in the representation of data subset 1. If an application requires a strong representation of data subset 1, a minor drop in overall representation would probably be an acceptable loss.

The overall results for the experiments 2 and 3 are significantly different. Unlike the first experiment, these experiments experience a significant *decrease* in the $\mathcal{Q}$ metric value of the full data set upon the introduction of data subset 2. This is caused by the composition of the data set. In these experiments, data subset 2 (No Highland

Whiskies) is significantly smaller than data subset 1. As a result, the overall $\mathcal{Q}$ metric value is skewed to the representation of data subset 1, and the loss of representation of data subset 1 impacts heavily upon the representation of the full data set. As the pruning threshold is increased, the loss of representation of data subset 1 is reduced. Given the strong dependence of the full data set on this subset, a similar reduction in overall representational loss is observed. When pruning is increased to a maximum ($\phi = 0.4$), virtually no representational loss is observed. The $\mathcal{Q}$ metric value for the representation of the full data set is virtually unchanged upon the introduction of data subset 2.

An interesting feature of these results for experiments 2 and 3 is the magnitude of arbor pruning threshold required to prevent a loss of representation of the overall data set. In these experiments, an arbor pruning threshold of $\phi = 0.2$ is sufficient to almost completely prevent any loss of representation in the full data set. The use of a low pruning threshold also allows for improvements in the representation of data subset 2. This shows that the use of a pruning threshold can significantly improve overall performance, without completely restricting the ability of the SOM to acquire new representations of new training data.

The results for these Scotch experiments appear contradictory. In experiment 1, pruning has the effect of increasing the $\mathcal{Q}$ metric value of the full data set, while in the second two experiments, pruning has the effect of increasing the $\mathcal{Q}$ metric value of the full data set. While these results appear to conflict, the contradiction is easily explained. In all experiments, the application of pruning limits the extent to which the full $\mathcal{Q}$ metric value deviates from the level observed at the end of the first phase of training. In all cases, complete arbor pruning results in a static $\mathcal{Q}$ metric value for the full data set upon the change in data set. The increase or decrease observed in the quality of representation of the overall data set is a function of the data sets themselves, not of the pruning function. The pruning function only serves to control the extent of any changes which occur to this representation as a result of the introduction of new training examples, or the removal of old examples.

8.4 Discussion

The results of the previous section repeatedly demonstrate that by restricting arborisation, it is possible to limit the loss of representation caused by missing or underrepresented training data. As the pruning threshold is increased, the SOM is increasingly

restricted in its ability to adapt to new training data, limiting the extent to which topologies learned using old training examples are forgotten.

The ability to restrict forgetfulness is not, however, universal. Care must be taken to ensure that all relevant inputs are represented in the initial training set if the arbor pruning technique is to be used. If new input dimensions are introduced during later training, arbor pruning at any level would prevent these input dimensions from being expressed.

Ensuring the presentation of all training inputs would be an appropriate task for syllabus learning (Chapter 7). By using a syllabus of known examples, it would be possible to ensure that all appropriate dimensions are represented in training before arbor pruning takes place. This initial training would force some weights above the pruning threshold; these weights would then retain the ability to be modified after arbor pruning has taken place.

Another possibility would be to implement a pruning scheme which selectively ignores specific inputs. If it is known that given subset of inputs cannot or will not be presented during initial training, these inputs could be exempted from the pruning process. This would preserve the ability to update weights connected to the novel inputs. The implementation of such a technique would require highly specific knowledge about the data set — for example, the knowledge that questions may be added to the questionnaire in the future. The testing of such a technique is left as future work.

The choice of an appropriate threshold value is also left as an open question. The smooth and continuous relationship between the arborisation threshold ϕ potentially allows the user to finely tune the level of arborisation to suit the desired level of concept drift and forgetfulness. This fine tuning is application specific; therefore, no general results are presented here.

However, while performing this fine tuning, it is worth remembering that the difference between limiting forgetfulness by using a large arborisation threshold, and not using continuous learning at all, is extremely limited. If the appropriate level of arborisation seems to require a large value of ϕ, one must question whether is it worth implementing arbor pruning at all. If the user is not willing to tolerate any concept drift, the choice to use a continuous learning mechanism must be questioned.

8.5 Conclusion

The continuously learning SOM algorithm suffers from one major drawback: the problem of forgetfulness. The SOM algorithm effectively forms a probability density map of the input set; the area given over to a concept on a trained SOM is directly related to the frequency of the training patterns representing that concept in the training set. This represents a significant problem for continuous learning problems. In such a problem, the training set changes constantly. Consequently, so does the probability distribution underlying the training set.

In this chapter, the technique of arbor pruning was introduced to limit forgetfulness. The arbor pruning technique utilises the correlation between weight magnitude and significance that is observed in maximal SOMs. By removing insignificant weights from the update process, it is possible to encourage the preservation of topologies without repeated presentation of data. This technique can be used if the data domain varies within a known set of dimensions. However, it cannot be used to control forgetfulness if dimensions are added or removed from the data domain.

In the next chapter, the techniques of step neighbourhood decay, syllabus presentation, and arbor pruning will be combined into a single training regime to realise a continuously learning SOM.

Chapter 9

A Lifetime Model Implemented

> "Well, what we called a computer in 1977 was really a kind of electric abacus [...] There really wasn't a lot this machine could do that you couldn't do yourself in half the time with a lot less trouble," said Richard, "but it was, on the other hand, very good at being a slow and dim-witted pupil."
>
> — *Dirk Gently's Holistic Detective Agency* (Adams, 1987)

9.1 Introduction

In Chapter 4, the importance of neighbourhood interactions in the self-organising process was established. Using this knowledge, four algorithmic extensions of Kohonen's maximal SOM algorithm were developed and presented. These extensions are:

Butterworth Step Decay (Chapter 5), a neighbourhood decay scheme which utilises a large neighbourhood size for 10 epochs, followed by prolonged training with a small neighbourhood size;

Continuous Learning (Chapter 6), a side effect of the use of Butterworth step decay which allows the learning process to extend into the period of use, rather than halting at the end of the traditional training period;

Syllabus Data Presentation (Chapter 7), a method of encouraging specific topologies to emerge as the principal topology when training data consists of more than two dimensions; and

Arbor Pruning (Chapter 8), a mechanism for preventing forgetfulness, so trained useful topologies are not lost over time due to a lack of repeated presentation.

As demonstrated in the previous chapters, these extensions provide useful functionality when used individually, but when used in concert, they implement the lifetime

model of learning and development proposed in Chapter 3. During the initial period of infancy, the cognitive process of an organism are extremely impressionable. The presence or absence of key training examples can drastically affect the later development of the individual. This corresponds to the period of training with a large neighbourhood size, using the fundamental patterns of the training syllabus. If the syllabus does not contain key dimensions of the expected real world data, these dimensions will not develop on the global map topology. During this phase, the use of a large neighbourhood size restricts the resolving power of the map.

After establishing the fundamental cognitive processes in the period of infancy, the individual progresses into a juvenile period. During this phase, the neighbourhood size decays to its smaller value, with training data continuing to come from the core training syllabus. This period of training establishes local topologies within the previously established global topology. As a result of the reduction of neighbourhood size, the resolving power of the map falls to a useful level. However, the map has not been exposed to sufficient data to be a useful tool.

During adolescence, directed learning on specific topics can take place. The syllabus is expanded to include more detail through the inclusion of less significant topological dimensions in the data set. Neighbourhood size remains small, providing good resolving power over this new training data. At the conclusion of this adolescence period, the map is can be used as a data analysis tool — it has been exposed to sufficient training data, and has the resolving power to differentiate these data.

Finally, the organism experiences a prolonged period of adulthood. This period of adulthood is prolonged and the space of input values can change during this time. Additional knowledge acquisition is limited to the fine tuning of well established cognitive patterns. This is achieved by pruning inactive weights to limit the acquisition of new concepts by the map.

It should be noted that just as is observed in nature, the change between stages is not a clear and obvious state change. There is no single point in time at which an individual changes from being a juvenile and becomes an adolescent (or any other sequential stages). Rather, a gradual shift from state to state occurs. This is achieved through gradual adjustment of parameters between extreme values, such as the Butterworth approximation of the step between large and small neighbourhood sizes.

In this chapter, the algorithmic extensions presented in the previous chapters will be used in concert, demonstrating that the lifetime model of learning and development

can be used to provide reduced training time, and increased map functionality in a data environment which is constantly changing.

9.2 Testing the Lifetime Model

The syllabus presentation and arbor pruning algorithms have been demonstrated extensively in previous chapters. However, in each demonstration only one algorithmic extension was utilised. The ability of the extensions to perform in concert has not been tested. The purpose of this experiment is to demonstrate that these extensions can be implemented in a single SOM, resulting in a continuously learning SOM.

The experiment takes the form of a thought experiment concerning the behaviour of a hypothetical agent which wanders Scotland seeking out Whisky distilleries and tasting their produce. The creator of the agent has a preconceived belief that Palette and Finish are the most defining characteristics of a whisky. Therefore, there is a preference that these characteristics dominate in the topology learned by the SOM.

Initially, the agent is land locked, and therefore experiences only Highland and Lowland whiskies. However, after a prolonged period of training, the agent gains access to a boat, discovers the Islay region, and adds Islay distilleries to its tasting tour. However, due to the time required for travel to this region, the agent ceases to travel into the Scottish Highlands and therefore ceases to have exposure to Highland whiskies.

This problem domain is continuous: the agent must be able to incorporate the experience of tasting Islay whiskies without forgetting the experience of Highland whiskies. Furthermore, there is significant background knowledge which should not be forgotten. This baseline knowledge (the importance of palette and finish) is provided by an experienced teacher.

9.2.1 Experimental Results

For the purposes of this experiment, the Scotch Whisky data set (see Section A.6) was divided into two subsets; Data Subset A consisting of Highland and Lowland whiskies, and Data Subset B consisting of Lowland and Islay whiskies. In addition, a training syllabus is constructed from Subset A by masking all inputs other than those corresponding to Palette and Finish characteristics. For the purposes of comparison, a syllabus consisting of manufacturing characteristics (Whisky Score, Distillery Score,

Alcohol Content and District) is also created; however, no training is performed using this syllabus.

Three trials were performed:

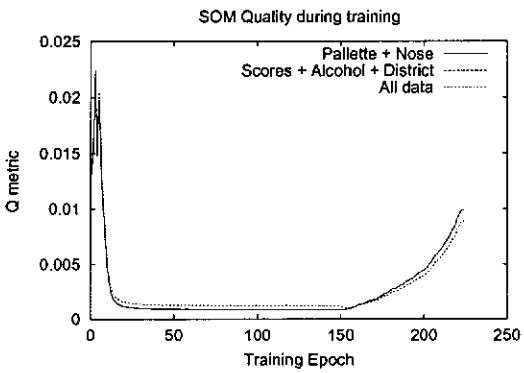
- No Syllabus, No arbor pruning;
- Syllabus, No arbor pruning; and
- Syllabus, Arbor pruning at $\phi = 0.2$.

Each trial was performed 10 times. $\mathcal{Q}$ metric performance was averaged over these 10 runs. In each trial, training was performed on a 25×25 neuron map. Training was divided into three periods of 75 epochs each. During the first two periods, data subset A was used. During the second two training periods, data subset B was used. In the second two trials, the syllabus was applied for the first period only.

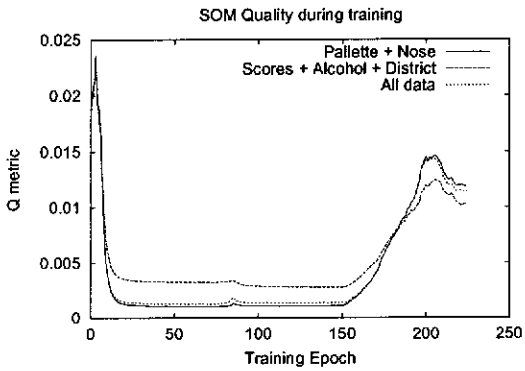
Butterworth step decay was used to decay neighbourhood size. However, no intermediate step was used. Neighbourhood size was decayed in a single step from 18 to 2, over a decay period of 10 epochs. No intermediate step was required due to the large number of unique training examples in the syllabus training data. Gain was kept at a constant value of 0.2 during all experiments.

The results of this experiment can be seen in Figures 9.1 and 9.2. These results demonstrate the familiar properties shown in the previous chapters. Without a syllabus (Figure 9.1(a)), the Scotch data set is able to develop a good topology, but there is no preferential representation of the Palette and Finish data. Topological components are equally represented in the output topology. By introducing a syllabus (Figures 9.1(b) and 9.1(c)), the representation of Palette and Finish data is made stronger than the representation of manufacturing characteristics. However, this comes at the cost of a slightly worse overall topological representation.

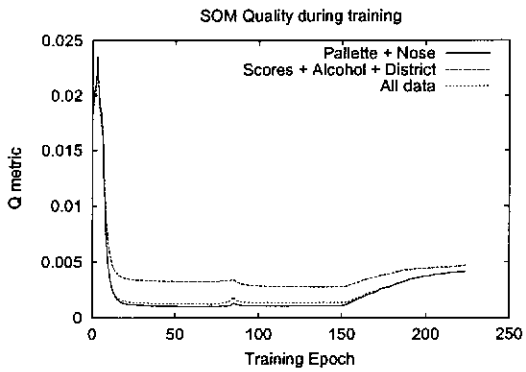
At the end of phase 2 of training (epoch 150), a stable representation of Data Subset A has been achieved in all 3 trials. Trials 2 and 3 both implement a syllabus. As a result, these trials are identical up to epoch 150. At this point, the second data subset is introduced, and arbor pruning takes effect for trial 3. In Trial 1 and 2, the representation of the full data set and Data Subset A decays significantly as a result of the change of training focus. These representational losses are limited by pruning in Trial 3. Similarly, the improvement gained in the representation of Data Subset B is restrained by pruning.



(a) No Syllabus, No Pruning



(b) Syllabus, No Pruning



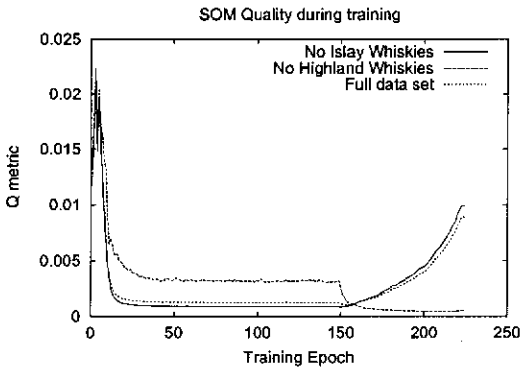
(c) Syllabus and Pruning

Figure 9.1: Representation of the training syllabi in a continuously learning SOM. The each line shows the $\mathcal{Q}$ metric for that syllabus of the data set.

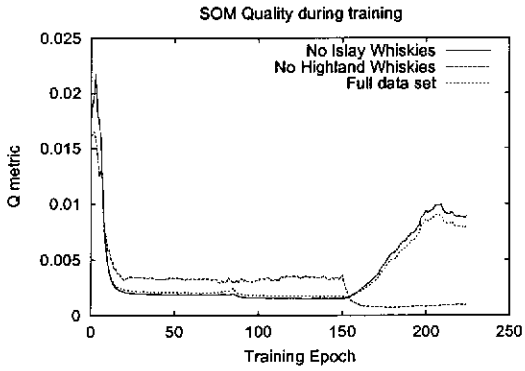
The success of this experiment should not be surprising. Each algorithmic technique was demonstrated successfully in previous chapters, and the periods of influence of these techniques do not overlap. Syllabus training should not extend beyond the juvenile period of training, while arbor pruning should not come into effect until adulthood. The experiment presented here, while demonstrative, does not reveal any genuinely novel results beyond the assurance that the techniques can be used in concert.

9.3 Conclusion

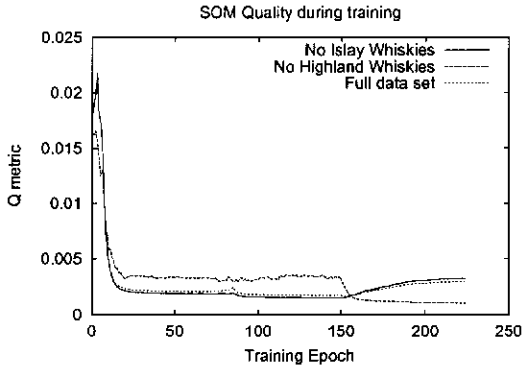
In this chapter, the algorithmic extensions presented in the previous chapters — Butterworth Step Decay, continuous SOM learning, syllabus presentation, and arbor pruning — were combined into a single process, resulting in a number of stages in the develop-



(a) No Syllabus, No Pruning



(b) Syllabus, No Pruning



(c) Syllabus and Pruning

Figure 9.2: Representation of data subsets in a continuously learning SOM. The each line shows the $\mathcal{Q}$ metric for that subset of the data set.

ment of a SOM topology. These steps can be seen to mirror the stages of development observed in biological organisms: a period of infancy during which both experience and resolving power is limited; a juvenile period during which experience is limited but resolving power is significantly improved; an adolescent period during which experience is expanded; and adult experience, during which experience is further expanded.

Each of the algorithmic extensions were demonstrated in previous chapters. The results presented in this chapter clearly demonstrate that the extensions can be used in concert to produce a continuously learning SOM, sufficiently static to retain significant initial knowledge, but sufficiently plastic to incorporate new knowledge.

Chapter 10

Conclusions

> "Well, I mean, *yes* idealism, *yes* the dignity of pure research, *yes* the pursuit of truth in all its forms, but there comes a point I'm afraid where you begin to suspect that if there's any *real* truth, it's that the entire multi-dimensional infinity of the universe is almost certainly being run by a bunch of maniacs. And if it comes to a choice between spending yet another ten million years finding that out, and other the other hand just taking the money and running then I for one could do with the exercise," said Frankie.
>
> — *The Hitch-Hikers Guide to the Galaxy* (Adams, 1979)

10.1 Summary

A common problem with traditional learning architectures is their failure to take into account the nature of real-world data. Traditional learning architectures generally restrict learning to a short training period prior to a prolonged period of use. This constrains the class of problem to which these techniques can be applied. Real world problems consist of data which can change over the lifetime of the system. Restricting learning to a short initial period prevents an architecture from adapting to changes in the data over the lifetime of the system.

The aim of this thesis was develop a model of the learning and development process as it occurs in biological systems, and to apply this model to Kohonen's Self-Organising Map (SOM) algorithm. This was achieved by breaking the lifetime of a single learning system into a number of stages, analogous to the infant, juvenile, adolescent and adult stages of development in a biological system.

In Chapter 3, the theoretical and empirical framework for this thesis was established. It was proposed that the use of a biologically inspired model of learning and development would enable the development of improved representations of a data domain. In addition, a metric was proposed for evaluating the quality of SOM data

representations. This metric quantifies the observable output of a SOM, rather than attempting to manipulate the SOM weight space.

In Chapter 4, a theoretical analysis of neighbourhood relationships in self-organising processes was presented. This analysis established the role played by lateral connections in organisation. It was then shown that in the Laplacian lateral connections common to many SOM architectures, the size of the lateral interaction kernel can have a significant effect on the speed of convergence, and the precision of representations formed.

The lessons learned in Chapter 4 were applied to Kohonen's SOM algorithm in Chapter 5. This resulted in the development of a new scheme for parameter decay in Kohonen's SOM algorithm. This scheme, known as *Butterworth Step Decay*, involves a smooth step between a large and small neighbourhood size. This decay scheme provides training times comparable to the best training times possible using traditional linear decay, but precludes the need for *a priori* knowledge of likely training times. This chapter also demonstrated that sophisticated schemes for gain decay are not necessary; the Butterworth Step Decay scheme is able to learn with constant gain.

One advantage of the Butterworth Step decay scheme is the fact that neighbourhood size never decays to 0. As a result, the Kohonen SOM algorithm never stops integrating training data into its weight representation. This capacity for continuous learning was tested in Chapter 6.

The establishment of a good fundamental data representation for a problem domain is of critical importance in almost all learning systems. In Chapter 7, a method for developing a strong fundamental topology in a SOM was presented. This technique is known as *Syllabus Presentation*. This technique involves using a selected training syllabus to reinforce knowledge known to be significant. A method for developing a training syllabus, known as *Percept Masking*, was also presented.

The continuous learning scheme presented in Chapter 6 suffers from one major limitation: forgetfulness. Training patterns which are not repeated eventually lose their representation on the SOM. In Chapter 8, a method was presented for realising a reduction in forgetfulness in a continuously learning SOM. This technique, known as *Arbor Pruning*, involves restricting weight update process to prevent the loss of significant representations. This technique can be used if the data domain varies within a known set of dimensions; however, it cannot be used to control forgetfulness if dimensions are added to or removed from the data domain.

Finally, in Chapter 9, the lessons learned in Chapters 5-8 are tied together in a

single experiment. This demonstrates that the techniques presented in the previous chapters are mutually compatible.

10.2 Future work

The material presented in this thesis suggests a number of avenues for future investigation.

10.2.1 Improvement of Percept Masking

The experiments presented in Chapter 7 showed that percept masking can be used to generate a useful training syllabus. However, the percept masking technique does make one significant assumption; that the underlying topology of a data set is orthogonal to the input components. The Scotch Whisky experiment demonstrated that this need not be the case. Complex data sets will exhibit complex topological relationships which are not orthogonal to the input components. In these problems, a simple binary masking technique will not produce the strongest possible fundamental topology. To represent these topologies, a more complex mask would be required.

One possible mechanism for generating such a mask would be the use of principal components or a Hotelling transform to filter complex topological relationships out of mixed data. Whereas simple percept masking represents a projection of a full training vector onto a subspace orthogonal to the input space, a Hotelling transform would allow projection onto a non-orthogonal subspace, allowing the representation of topologies across inputs. The development of such a technique is left as future work.

10.2.2 Improvement of Arbor Pruning

The experiments presented in Chapter 8 showed that by restricting input arborisation, it is possible to limit forgetfulness in a continuously learning SOM. However, the ability to restrict forgetfulness is not universal. Care must be taken to ensure that all relevant inputs are represented in the initial training set if the arbor pruning technique is to be used. If new input dimensions are introduced during later training, arbor pruning at any level would prevent these input dimensions from being expressed.

One possible technique for rectifying this problem would be to implement a pruning scheme which selectively ignores specific inputs. If it is known that given subset of inputs cannot or will not be presented during initial training, these inputs could be

exempted from the pruning process. This would preserve the ability to update weights connected to the novel inputs. The implementation of such a technique would require highly specific knowledge about the data set — for example, the knowledge that questions may be added to the questionnaire in the future. This knowledge is analogous to the knowledge required to form a training syllabus using the percept masking technique. The development and testing of an improved pruning technique is left as future work.

In addition, a method of selecting an appropriate pruning level is required. The experiments presented in Chapter 8 suggest that the pruning threshold is somewhat data set dependent. The development of a general technique for selecting an arbor pruning threshold is another avenue for future work.

10.2.3 Remainder of Life Cycle

In Chapter 3, a lifetime model was presented. This lifetime model encompasses the life-cycle of an individual organism, and the interaction between an individual, its progenitors and its offspring.

The work presented in this thesis represents a significant subset of this model. In this thesis, the lifetime of a single individual was investigated. The role played by genotypic expansion, genetic variation, and fitness selection in the development of the individual was largely ignored. These areas could be explored in future work.

The subject of parental interaction was briefly addressed in Chapter 7, in the discussion of Syllabus Presentation. The percept masking technique requires that the syllabus be generated based upon *a priori* knowledge, provided by an informed trainer. In a complete model of learning and development, this syllabus would be automatically generated by the parent. An automated mechanism for the development of training syllabi is another possible source of future research.

10.2.4 Alternate Neighbourhood Kernels

The maps resulting from traditional self-organising architectures are topological ones; that is, the distance between the mapped positions of two training vectors is directly related to the Euclidean distance between those training vectors. While this is a useful feature to extract, the self-organising technique could be used to generate other mappings with potential uses for pattern recognition tasks.

The technique presented in Chapter 4 allows the experimenter to evaluate the plau-

sibility of alternate lateral connection strategies. Using this technique, it is possible to establish the form taken by stable output states of a neural system as a result of a specific lateral connection strategy. It is also possible to rule out lateral connection strategies that will not produce stable output states, or produce saturated output states.

It should be noted that this technique does not tell the experimenter the type of mapping that will result from using a specific connection strategy — this remains an empirical problem. The role played by this technique is to eliminate from further investigation lateral connection strategies which will not yield a stable output. The development of alternate lateral connection strategies, and determining their significance, is another avenue for future work.

10.2.5 Conversion from Maximal Maps to Minimal Maps

One of the major limitations of this research is the requirement that a maximal SOM be used. This conflicts with common practice: minimal SOMs are the overwhelmingly dominant SOM architecture in the literature. This is due to their improved convergence speed, and enhanced representational capabilities.

Most of the techniques developed in this thesis can be applied to minimal SOMs without alteration. The Butterworth Step Decay, continuous learning and syllabus presentation techniques are not dependent upon the internal representation of a SOM for their application. Exploratory testing suggests that these techniques would be of equal utility on a minimal map as they have proven to be on a maximal map.

However, the arbor pruning technique cannot be applied to a minimal SOM. Arbor pruning requires a correlation between weight strength and significance to be effective. This correlation does not exist in minimal style maps.

One possible approach to rectifying this situation would be to use a method similar to that used by DeSieno (1998) in the 'SOM with a Conscience' algorithm. In this algorithm, each neuron is given a bias parameter, which is adjusted each time the neuron is selected as a winner. This parameter gives an indication of the significance of a given neuron. This parameter could then be used as a basis for pruning. The tuning of the conscience parameter, and the development of an appropriate pruning strategy is left as future work.

10.2.6 Applying the Lifetime Model to Other Architectures

The lifetime learning model presented in Chapter 3 is architecture agnostic. While the SOM algorithm is a powerful mechanism for data visualisation, and this thesis has applied the lifetime model to the SOM algorithm, the lifetime model is not restricted to use with SOMs. Other machine learning algorithms could benefit from a similar approach to learning. This represents a significant body of potential future research.

Appendix A

Data Sets

> "But look, you found the notice, didn't you?"
>
> "Yes," said Arthur, "yes I did. It was on display in the bottom of a locked filing cabinet in a disused lavatory with a sign on the door saying 'Beware of the Leopard'."
>
> — *The Hitch-Hiker's Guide to the Galaxy* (Adams, 1979)

A.1 Introduction

In the course of this thesis, a number of data sets are used to demonstrate key characteristics of Kohonen's algorithm and its variants. The details of these data sets are presented here so that they may be reproduced accurately by other researchers.

It should be noted that the data sets presented in this Appendix are somewhat specific to the style of map used. As was described in Section 2.5.1, this thesis utilises maximal SOMs throughout. This places certain restrictions on the types of data set which can be learned. While this does not preclude the use of these data sets in minimal SOMs, there are almost certainly more efficient representations.

A.2 Simple Line

The simple line data set is an attempt to simulate one dimensional input of the type likely to be experienced by a tonotopic map in the ear. Each vector in the data set is the representation of the ear 'hearing' a single frequency tone and the peripheral tonal stimulation resulting from 'hearing' the tone.

A.2.1 Training Set

Construction of Training Vectors

Each training pattern in the simple line data set consists of a vector of n inputs. One of these inputs is selected as the stimulated input, and is given a value of 1.0. The two immediate neighbours of the stimulated input are given a value of 0.5. This neighbourhood relationship wraps around the end of the vector (i.e., the first vector element is a neighbour of the last). All other values in the vector are set to zero.

Construction of Training Set

The training set consists of n training vectors, one example of stimulation of each input. These vectors are presented in random order during the training process.

A.2.2 Testing Set

The simple line data set is used solely to demonstrate that a stable output can be generated by a laterally connected self-organising neural system. Therefore, it does not require an independent testing set.

A.3 Simple Grid

The simple grid data set is a two dimensional extrapolation of the simple line data set, representing input of the type likely to be experienced by retinotopic or somatosensory maps in the brain. Each training vector represents the stimulation of a single point on a two dimensional surface, plus the peripheral excitation resulting from that point stimulation.

A.3.1 Training Set

Construction of Training Vectors

Each training pattern is formed from an $n \times m$ matrix. One of the elements of the matrix is selected as the stimulated input, and is considered to be the centre of a Gaussian bell with standard deviation of 1. The value of all other inputs are taken by sampling this Gaussian bell as a function of the distance from the stimulated input (for example, if element $(2,3)$ is the stimulated input, element $(3,5)$ has a distance of $\sqrt{1^2+2^2} \approx 2.23$ from the centre of activation, yielding an activation level of $exp(-2.23^2/2) \approx 0.08$.

There are no wraparound neighbourhood relationships in this matrix; if the stimulated input falls near the edge of the matrix, a partial Gaussian bell is observed.

This matrix of sampled values is then converted into a vector by concatenating rows of the matrix, yielding a vector with nm elements.

Construction of Training Set

The training set consists of nm training patterns; one example of stimulation of each input. These vectors are presented in random order during the training process.

A.3.2 Testing Set

Construction of Testing Vectors

The construction of testing vectors is almost identical to the construction of training vectors, except that stimulation points are not limited to integer values. Instead, any value in $\Re^2$ in the range $[1:n, 1:m]$ can be selected. The Gaussian bell is fitted about this point, and the testing values are sampled at integer intervals.

Construction of Testing Set

Testing sets of arbitrary size can be constructed by randomly selecting points from $\Re^2$ as centres of activation and constructing matrices and test vectors accordingly.

A.4 Questionnaire

The questionnaire data set simulates the responses to a questionnaire consisting of n questions. Each question can be answered either 'Yes' or 'No'.

A.4.1 Training Set

Construction of Training Vectors

Each training pattern consists of the responses to the questionnaire of a single individual. The answer to each question is encoded as an input pair: $\{1,0\}$ for 'Yes', and $\{0,1\}$ for 'No'. Each training vector therefore consists of $2 \times n$ inputs.

The paired representation of boolean training values is required as a result of the use of a maximal SOM. Maximal SOMs maintain a direct connection between signal strength and signal significance. That is, the magnitude of an input represents the significance of that input to the training pattern. When using boolean training values,

a value of 0 is just as significant as a value of 1. However, using a maximal SOM, it is not possible to discern a difference between a 'No' signal, and an absent signal (a signal with no signal strength) — using single inputs, both would be represented by the value 0. By encoding as pairs, the 'No' signal is given a unique representation as a significant negative; an absent signal is represented as the pair of insignificant signals $\{0, 0\}$.

Construction of Training Set

The training set consists of a spanning set of questionnaire responses. A single example of each and every possible combination of responses is present in the data set. A questionnaire with n questions therefore yields a training set of 2^n patterns. These patterns are presented in random order during the training process.

A.4.2 Testing Set

No independent test set exists for this data set; the training set spans all possible values for input. The testing set is therefore identical to the training set.

A.5 Grid-in-a-Grid

The Grid-in-a-Grid data set is a synthetic example intended to demonstrate the process of learning a data set known to possess two independent complex topologies. This data set can be conceptualised as a large grid which has a smaller grid embedded at each grid point of the larger grid.

A.5.1 Training Set

Construction of Training Vectors

Two simple grid training vectors are generated using the technique described in Section A.3. These two grids need not be of the same size (i.e., if the large grid is of size $n \times m$, and the small grid is of size $p \times q$, there are no equivalence requirements between n, m, p, and q). These two vectors are then concatenated, resulting in a training vector with $nm + pq$ elements describing two points in two separate topologies.

It should be noted that although the two sets are referred to as the large grid and the small grid, there is no requirement that $n, m > p, q$. The adjectives 'large' and

'small' are merely used to describe their position in the principal topology of the data set.

Construction of Training Set

The training set consists of the concatenation of all possible combinations of grid points from the two grid training sets.

A large grid of size $n \times m$ and a small grid of size $p \times q$ will therefore yield a training set consisting of $nmpq$ training vectors. These patterns are presented in random order during the training process.

A.5.2 Testing Set

Construction of testing vectors

The construction of testing vectors is performed by combining the methods for creating a Simple Grid testing vector and the Grid-in-a-Grid training vector.

Two Simple Grid testing vectors are generated and concatenated by randomly selecting points in $\Re^2$ and building an activation matrix and simple grid test vector. These two test vectors are then concatenated to yield a Grid-in-a-Grid test vector.

Construction of testing set

As with the training vectors, testing vectors for the Grid-in-a-Grid data set are constructed by generating a Simple Grid test vectors for each of the two sub-grids, and concatenating these test vectors. Testing sets of arbitrary size can be generated using this technique.

A.6 Scotch Whisky

Whisky, from the Gaelic *uisgebeatha*, meaning 'water of life', is defined as alcoholic liquor obtained by the distillation of a fermented starchy compound, usually a grain. Single-malt Scotch Whisky, the pinnacle of the form, is produced by 109 distilleries in Scotland (as well as one distillery in Ireland, and three in Japan).

To qualify as a single malt Scotch Whisky, a liquor must be produced from fermented (malted) barley, produced in a pot still in a single distillery, aged in oak for at least three years. Although this is a highly prescribed process, each distillery produces a whisky

with distinctive characteristics, which are savoured at great length by connoisseurs and amateurs alike.

In 'The Malt Whisky Companion' (Jackson, 1989), an expert analysis of each of all available single malt scotch whiskies is presented, yielding a set of classification terms based upon color (14 terms), nose (12 terms), body (8 terms), palette (15 terms) and finish (19 terms). These characteristics are binary variables. In addition to these binary characteristics, this analysis provides the age, alcohol content, region, district, a score rating the quality of the distillery, and a score rating the quality of each whisky.

This data set was selected as a real world example because:

- it is a large and complex data set;
- it can be formulated in a manner compatible with maximal SOMs;
- it has been described by Lapointe and Legendre (1994) providing a statistical benchmark for comparison;
- it has been analysed with a SOM (Deboeck and Kohonen, 1998), providing a benchmark for comparison of self-organising performance;
- it is in the public domain, and downloadable from:

  `http://www.fas.umontreal.ca/biol/casgrain/en/labo/scotch.html`

- it is an example that is able to sustain the interest of readers from almost any discipline.

A.6.1 Training Set

There are 109 distilleries in Scotland which produce single malt Scotch Whisky. One whisky from each distillery was selected and characterised, yielding 109 training vectors in total. The names of these distilleries, and the abbreviated names used on graphical representations of self-organising maps, are given in Tables A.1 and A.2.

The full set of potential values for the descriptors in each training vector can be seen in Table A.3. The binary classification terms (colour, nose, body, palette, and finish) are encoded one input per term, yielding a total of 68 inputs. The district descriptor is also encoded as a set of 3 binary inputs (100 $\rightarrow$ Highland, 010 $\rightarrow$ Lowland, 001 $\rightarrow$ Islay).

Abbreviated Name	Full Name	Abbreviated Name	Full Name
Abef	Aberfeldy	Abel	Aberlour
Ardb	Ardberg	Ardm	Ardmore
Auch	Auchentoshan	Ault	Aultmore
Balb	Balblair	Balm	Balmenach
Balv	Balvenie	Banf	Banff
Benn	Ben Nevis	Benr	Benromach
Bera	Benriach	Bern	Benrinnes
Blad	Bladnoch	Blai	Blair Athol
Bowm	Bowmore	Brac	Brackla
Brui	Bruichladdich	Bunn	Bunnahabhain
Caol	Caol Ila	Cape	Caperdonich
Card	Cardhu	Clyn	Clynelish
Cole	Coleburn	Conv	Convalmore
Crag	Cragganmore	Crai	Craigellachie
Dail	Dailuaine	Dall	Dallas Dhu
Dalm	Dalmore	Dalw	Dalwhinnie
Dean	Deanston	Duff	Dufftown
Edra	Edradour	Fett	Fettercairn
Galb	Glen Albyn	Gall	Glenallachie
Gbur	Glenburgie	Gcad	Glencadam
Gdev	Glen Deveron	Gdro	Glendronach
Gdul	Glendullan	Gelg	Glen Elgin
Gesk	Glenesk	Gfar	Glenfarclas
Gfid	Glenfiddich	Ggar	Glen Garioch
Ggla	Glenglasshaugh	Ggoy	Glengoyne
Ggra	Glen Grant	Gkei	Glen Keith
Gkin	Glenkinchie	Gliv	Glenlivet
Gloc	Glenlochy	Glos	Glenlossie
Gmho	Glen Mhor	Gmon	Glenmorangie
Gmoy	Glen Moray	Gord	Glenordie (Ord)
Grot	Glenrothes	Gsco	Glen Scotia
Gspe	Glen Spey	Gtau	Glentauchers
Gtur	Glenturret	Gugi	Glenugie
Gury	Glenury	High	Highland Park
Impe	Imperial	Incg	Inchgower
Incm	Inchmurrin	Inve	Inverleven
Jura	Jura	Kinc	Kinclaith
Knoc	Knockando	Knod	Knockdhu
Lady	Ladyburn	Laga	Lagavulin
Laph	Laphroaig	Link	Linkwood
Litt	Littlemill	Locn	Lochnagar
Locs	Lochside	Long	Longmorn
Maca	Macallan	Mill	Millburn
Milt	Miltonduff	Mort	Mortlach

Table A.1: Names of Scotch Whisky Distilleries, with abbreviations used. Part I, A-M

Abbreviated Name	Full Name	Abbreviated Name	Full Name
Nort	North Port	Oban	Oban
Port	Port Ellen	Pult	Pulteney
Rose	Rosebank	Sain	Saint Magdalene
Scap	Scapa	Sing	Singleton (Auchroisk)
Spey	Speyburn	Spri	Springbank
Splo	Springbank-Longrow	Stra	Strathisla
Tali	Talisker	Tamd	Tamdhu
Tamn	Tamnavulin	Tean	Teaninich
Tobe	Tobermoray	Toma	Tomatin
Tomi	Tomintoul	Torm	Tormore
Tull	Tullibardine		

Table A.2: Names of Scotch Whisky Distilleries, with abbreviations used. Part II, N-Z

The remaining descriptors (age, alcohol content, distillery score, whisky score, and district) are continuous in nature, and thus require special encoding in a maximal SOM in order to preserve continuity relationships. To achieve this, a single continuous variable is encoded using an ordered series of inputs. Each of these inputs is used to represent a key value in the expected range of the continuous variable. These key values are evenly spaced along the expected range. The activation levels assumed by the inputs take their values from the sampled value of a Gaussian curve with a mean equivalent to the value of the continuous variable, and a standard deviation equivalent to the spacing used between key values.

Figure A.1 gives an example of this process. In this figure, the age of an 11 year old scotch is represented. Scotch age is known to vary between 8 and 20 years; a set of 7 inputs, separated by 2 year intervals is used to represent this range. A Gaussian bell with standard deviation of 1 input (2 years) is centered at age 11 (between inputs 2 and 3). This yields a training vector of 0.25 0.86 0.86 0.25 0.02 0.00 0.00].

As a result of this style of encoding, age is encoded as 7 inputs (with peaks at 8, 10, 12, 14, 16, 18 and 20), distillery score is encoded as 5 inputs (with peaks at 1, 2, 3, 4 and 5), whisky score is encoded as 6 inputs (with peaks at 50, 60, 70, 80, 90 and 100), and alcohol content is encoded as 8 inputs (with peaks at 40, 42.5, 45, 47.5, 50, 52.5, 55 and 57.5). This represents an additional 26 inputs. The binary descriptors, district and continuous descriptors are concatenated to produce a complete training vector with a total of 97 inputs per vector.

An example training vector is presented in Figure A.2. In this figure, whisky from

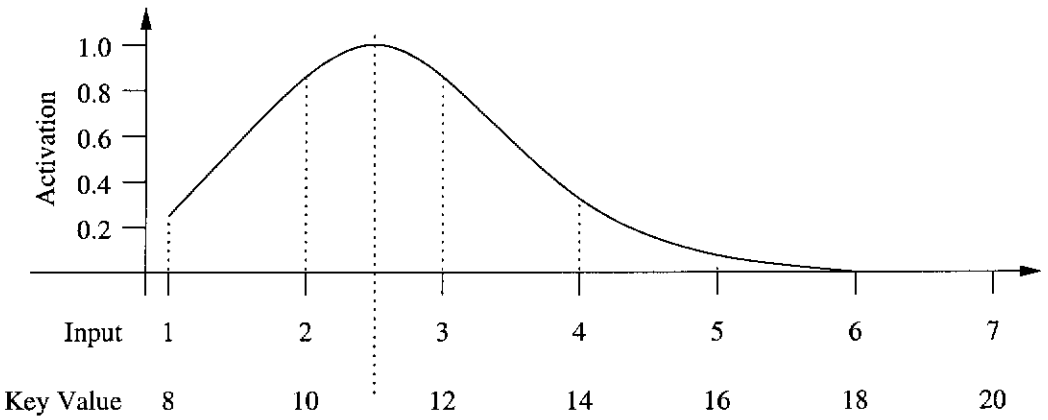


Figure A.1: Creating Maximum style input from a continuous parameter: Representation of a scotch aged 11 years.

Component	Training Vector
Color	0 1 0 0 0 0 0 0 0 0 0 0 0 0
Nose	1 1 0 0 0 0 1 0 0 0 0 0
Body	1 1 0 0 0 0 0 0
Palette	0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
Finish	1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
Age	0.86 0.86 0.25 0.02 0 0 0
Distillery Score	0.54 1 0.54 0.08 0
Whisky Score	0.11 0.61 0.99 0.47 0.07 0
Alcohol content	1 0.54 0.08 0 0 0 0 0
District	1 0 0

Figure A.2: Example training vector: Aberfeldy (Abef) distillery. The training vector is formed from the concatenation of these individual components.

the Aberfeldy distillery is described. This distillery is in the Highlands district, and has a distillery score of 2. The whisky produced at Aberfeldy is 9 years old, yellow in colour, has an aromatic, peaty and fruity nose, a soft medium body, is oily and sherry-like on the palette and has a full, dry, spicy and fruity finish. This whisky has an alcohol content of 40%, and scores 69%.

A.6.2 Testing Set

No independent test set exists for this data set; the training set spans all known values for input. The testing set is therefore identical to the training set.

Characteristic	Data Type	Valid values
Colour	One of	White wine, yellow, very pale, pale gold, gold, old gold, full gold, bronze, pale amber, amber, full amber, red, fino sherry.
Nose	Any of	Aromatic, peaty, sweet, light, fresh, dry, fruity, grassy, salty, sherry, spicy, rich.
Body	Any of	Soft, medium, full, round, smooth, light, firm, oily.
Palette	Any of	Full, dry, sherry, big, light, smooth, clean, fruity, grassy, smoky, sweet, spicy, oily, salty, aromatic.
Finish	Any of	Full, dry, warm, big, light, smooth, clean, fruity, grassy, smoky, sweet, spicy, oily, salty, aromatic, quick, long, very long, lingering.
Age	Integer	> 3 (All data in range $8 - 20$)
Distillery Score	Integer	in range $0 - 5$ (All data in range $1 - 5$).
Whisky Score	Integer	in range $0 - 100$ (All data in range $55 - 90$).
Alcohol content	Integer	in range $0 - 100$ (All data in range $40 - 57.1$)
District	One of	Highland, Lowland, Islay.

Table A.3: Descriptive Characteristics of Single Malt Scotch Whisky.

Bibliography

Adams, D. (1979). *The Hitch-Hikers Guide to the Galaxy.* Pan Books.

Adams, D. (1980). *The Restaurant at the End of the Universe.* Pan Books.

Adams, D. (1987). *Dirk Gently's Holistic Detective Agency.* Heinemann.

Adams, D. (1988). *The Long Dark Tea-Time of the Soul.* Heinemann.

Adams, D. (1992). *Mostly Harmless.* Heinemann.

Ahn, W. and Brewer, W. F. (1993). Psychological studies of explanation-based learning. In *Investigating Explanation-Based Learning.* Kluwer Academic Publishers, Boston, Dordrecht and London.

Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, **27**, 77–87.

Amari, S. (1980). Topographic organisation of nerve fields. *Bulletin of Mathematical Biology*, **42**, 339–364.

Amari, S. (1990). Mathematical foundations of neurocomputing. *Proceedings of the IEEE*, **78**(9), 1443–1464.

Apter, J. T. (1945). Projection of the retina on superior colluculus of cats. *Journal of Neurophysiology*, **8**, 123–134.

Baraldi, A. and Blonda, P. (1998). A survey of fuzzy clustering algorithms for pattern recognition II. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, **29**, 786–801.

Baraldi, A. and Parmiggiani, F. (1997). Novel neural network model combining radial basis function, competitive Hebbian learning rule, and fuzzy simplified Adaptive Resonance Theory. In *Proceedings of SPIE Optical Science, Engineering and Instrumentation 1997: Applications of Fuzzy Logic Technology IV*, volume 3165, pages 98–112, San Diego.

Bauer, H. and Pawelzik, K. R. (1992). Quantifying the neighborhood preservation of Self-Organizing Feature Maps. *IEEE Transactions on Neural Networks*, **3**(4), 570–579.

Bauer, H. and Villmann, T. (1997). Growing a hypercubical output space in a Self-Organizing Feature map. *IEEE Transactions on Neural Networks*, **8**(2), 218–226.

Beymer, D. and Poggio, T. (1995). Face recognition from one model view. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 500–507.

Bishop, C. M., Svensén, M., and Williams, C. K. I. (1997). GTM: A principled alternative to the Self-Organizing Map. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 354–360. The MIT Press, Cambridge, MA.

Bishop, C. M., Svensén, M., and Williams, C. K. I. (1998). GTM: The generative topographic mapping. *Neural Computation*, **10**, 215–234.

Blackmore, J. and Miikkulainen, R. (1993). Incremental Grid Growing: Encoding high-dimensional structure into a two dimensional feature map. In *Proceedings of ICNN93: International Conference on Neural Networks*, volume I, pages 450–455, Piscataway, N.J. IEEE Service Center.

Blackmore, J. and Miikkulainen, R. (1995). Visualizing high-dimensional structure with the Incremental Grid Growing neural network. In A. Prieditis and S. Russell, editors, *Machine Learning. Proceedings of the Twelfth International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann Publishers, San Francisco.

Blakemore, C. and Cooper, G. (1970). Development of the brain depends on the visual environment. *Nature*, **228**, 477–478.

Carpenter, G. and Grossberg, S. (1987a). ART2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, **26**, 4919–4930.

Carpenter, G. and Grossberg, S. (1987b). ART2A: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, **4**, 493–504.

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

Carpenter, G. and Grossberg, S. (1990). ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, **3**, 129–152.

Carpenter, G. and Grossberg, S. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, **4**, 759–771.

Carpenter, G., Grossberg, S., and Reynolds, J. (1991). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, **4**, 565–588.

Carpenter, G., Grossberg, S., Markuzon, N., Reynolds, J., and Rosen, D. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, **3**(5), 698–713.

Cooper, S., Daniel, P., and Whitteridge, D. (1953). Nerve impulses in the brain stem and cortex of the goat. *Journal of Physiology*, **120**, 514–527.

Cottrell, M., Fort, J., and Pagés, G. (1998). Theoretical aspects of the SOM algorithm. *Neurocomputing*, **21**, 119–137.

Deboeck, G. and Kohonen, T., editors (1998). *Visual Explorations in Finance with Self-Organizing Maps*. Springer.

DeSieno, D. (1998). Adding a conscience to competitive learning. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 117–124.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, **48**, 71–99.

Erwin, E., Obermayer, K., and Schulten, K. (1992a). Self-Organizing Maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, **67**(1), 47–55.

Erwin, E., Obermayer, K., and Schulten, K. (1992b). Self-Organizing Maps: Stationary states, metastability and convergence rate. *Biological Cybernetics*, **67**(1), 35–45.

Fahlman, S. E. (1991). The Recurrent Cascade-Correlation architecture. Technical Report CMU-CS-91-100, Carnegie Mellon University, School of Computer Science.

Fahlmann, S. E. and Lebiere, C. (1990). The Cascade-Correlation learning architecture. In D. S. Touretsky, editor, *Advances in Neural Information Processing Systems*. Morgan Kauffman.

Fisher, D. H. (1990). Knowledge aquisition via incremental conceptual clustering. In J. W. Shavlik and T. G. Diettrich, editors, *Readings in Machine Learning*, pages 267–284. Morgan Kauffman.

Flanagan, J. A. (1998). Sufficient conditions for self-organization in the one-dimensional SOM with a reduced width neighbourhood. *Neurocomputing*, **21**, 51–60.

Frean, M. (1990). The Upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, **2**, 198–209.

Fritzke, B. (1992). Growing cell structures—a self-organizing network in k dimensions. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks, 2*, volume II, pages 1051–1056. North-Holland.

Fritzke, B. (1994). Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, **7**(9), 1441–1460.

Fritzke, B. (1995). Growing grid — a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, **2**(5), 9–13.

Fritzke, B. (1996). Growing self-organizing networks — why? In *ESANN'96: European Symposium on Artificial Neural Networks*, pages 61–72.

Gaze, R. M. (1958). The representation of the retina on the optic lobe of the frog. *Quarterly Journal of Experimental Physiology*, **43**, 209–224.

Gaze, R. M., Keating, M. J., Szèkely, G., and Beazley, L. (1970). Binocular interaction in the formation of specific intertectal neuronal connections. *Proceedings of the Royal Society*, **B**(175), 107–147.

Gennari, J. H., Langley, P., and Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, **40**, 11–61.

Grossberg, S. (1986). Adaptive pattern classification and universal recoding I: Parallel development and coding of neural features. *Biological Cybernetics*, **23**, 121–134.

Haese, K. (1998). Self-Organizing Feature Maps with self-adjusting learning parameters. *IEEE Transactions on Neural Networks*, **9**(6), 1270–1278.

Haese, K. and Goodhill, G. J. (2001). Auto-SOM: Recursive parameter estimation for guidance of Self-Organizing Feature Maps. *Neural Computation*, **13**, 569–619.

Hebb, D. O. (1949). *The Organisation of Behaviour.* John Wiley, New York.

Hecht-Nielsen, R. (1990). *Neurocomputing.* Addison-Wesley, U.S.A.

Heins, L. and Taurisz, D. (1995). Adaptive Resonance Theory (ART): An introduction. Technical report, Department of Computer Science, Leiden University.

Heskes, T. (1999). Energy functions for Self-Organizing Maps. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 303–316. Elsevier, Amsterdam.

Heskes, T. and B., K. (1993). Error potential for self-organization. In *Proceedings of ICNN'93: International conference on Neural Networks*, volume 3, pages 1219–1223.

Hubel, D. H. and Wiesel, T. N. (1963). Receptive fields of cells in striate cortex of very young, visually inexperienced kittens. *Journal of Neurophysiology*, **26**, 994–1002.

Hush, D. R. and Horne, B. (1992). An introduction to the theory of neural networks. Technical Report EECE90-005, Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131.

Jackson, M. (1989). *Michael Jackson's Malt Whiskey Companion: A Connoisseur's Guide to Malt Whiskies of Scotland.* Dorling Kindersley, London.

Kang, B., Kim, J., and Cho, M. (1995). Learning rate updating schemes of unsupervised learning. In *Proceedings of IEEE Conference on Systems, Man and Cybernetics*, pages 3259–3262.

Kangas, J. and Kaski, S. (1998). 3043 works that have been based on the Self-Organizing map (som) method developed by Kohonen. Technical Report A49, Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Computer and Information Science.

Kangas, J., Kohonen, T., Laaksonen, J., Simula, O., and Ventä, O. (1989). Variants of Self-Organizing Maps. In *Proceedings of IJCNN89: International Joint Conference on Neural Networks*, volume II, pages 517–522, Piscataway, N.J. IEEE Service Center.

Kaski, S. and Kohonen, T. (1994). Winner-take-all networks for physiological models of competitive learning. *Neural Networks*, **7**, 973–984.

Kaski, S. and Lagus, K. (1996). Comparing Self-Organizing Maps. In C. Von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of ICANN96, International Conference on Artificial Neural Networks, Bochum, Germany, July 16–19*, pages 809–814, Berlin. Springer.

Keith-Magee, R. (1997). A neural architecture for modelling error in low resolution numerical predictions of mathematically intractable systems. Honours Thesis, Department of Computer Science, Curtin University of Technology.

Kiviluoto, K. (1996). Topology preservation in Self-Organizing Maps. In *Proceedings of ICNN96: International Conference on Neural Networks*, volume I, pages 294–299, Piscataway, N.J. IEEE Service Center.

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

Kohonen, T. (1988). *Self-Organization and Associative Memory.* Springer Series in Information Sciences. Springer-Verlag, 2nd edition.

Kohonen, T. (1990). The Self-Organizing map. *Proceedings of the IEEE*, **78**(9), 1464–1479.

Kohonen, T. (2001). *Self-Organizing Maps.* Springer Series in Information Sciences. Springer-Verlag, 3rd edition.

Kohonen, T. and Hari, R. (1999). Where the abstract feature maps of the brain might come from. *Trends in Neurosciences*, **22**(3), 135–139.

Kohonen, T., Kaski, S., and Lappalainen, H. (1997). Self-organized formation of various invariant-feature filters in the Adaptive-Subspace SOM. *Neural Computation*, **9**(6), 1321–1344.

Koikkalainen, P. and Oja, E. (1990). Self-organising heirarchical feature maps. In *Proceedings of IJCNN90: International Joint Conference on Neural Networks*, volume II, pages 279–285, Piscataway, N.J. IEEE Service Center.

Kraaijveld, M. A., Mao, J., and Jain, A. K. (1992). A non-linear projection method based on Kohonen's topology preserving maps. In *Proceedings of ICPR92: 11th IAPR International Conference on Pattern Recognition*, volume B, pages 41–45, Los Alamitos, CA. IEEE Comput. Soc. Press.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957, San Mateo. Morgan Kauffmann.

Kubat, M. (1989). Floating approximation in time-varying knowledge bases. *Pattern Recognition Letters*, **10**, 223–227.

Kubat, M. (1991). Conceptual inductive learning: The case of unreliable teachers. *Artificial Intelligence*, **52**, 169–182.

Kubat, M. (1993). Flexible concept learning in real time systems. *Journal of Intelligent and Robotic Systems*, **8**, 155–171.

Lando, M. and Edelman, S. (1995). Generalizing from a single view in face recognition. Technical Report CS-TR-95-02, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel.

Lapointe, F. and Legendre, P. (1994). A classification of pure malt scotch whiskies. *Applied Statistics*, **43**(1), 237–257.

Lazarescu, M., Venkatesh, S., and West, G. (1999). Incremental learning with forgetting (ILF). In *Proceedings of ICML-99 Workshop on Machine Learning in Computer Vision.*

Le Cun, Y., Denker, J., and Solla, S. (1990). Optimal Brain Damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605, San Mateo. Morgan Kaufmann.

Lebowitz, M. (1986). Concept learning in a rich input domain: Generalization based memory. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann.

Lebowitz, M. (1989). Categorizing numeric information for generalization. *Cognitive Science*, **9**, 285–309.

Liebert, W., Pawelzik, K., and Schuster, H. G. (1991). Optimal embeddings of chaotic attractors from topological considerations. *Europhysics Letters*, **14**, 521.

Lippman, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22.

Lo, Z. and Bavarian, B. (1991). Improved rate of convergence in Kohonen neural networks. In *Proc. IJCNN'91, International Joint Conference on Neural Networks*, volume II, pages 201–206, Piscataway, NJ. IEEE Service Center.

Loh, A., Robey, M., and West, G. (2001). IFOSART: A noise resistant neural network capable of incremental learning. *IEEE Transactions on Neural Networks*. Currently under review.

Luttrell, S. P. (1991). Code vector density in topographic mappings: Scalar case. *IEEE Transactions on Neural Networks*, **2**(4), 427–436.

Luttrell, S. P. (1992). Code vector density in topographic mappings. Technical Report 4669, RSRE, Malvern.

Luttrell, S. P. (1994). A Bayesian analysis of Self-Organizing Maps. *Neural Computation*, **6**(5), 767–794.

Martin-Smith, P., Pelayo, F. J., Diaz, A., Ortega, J., and Prieto, A. (1993). A learning algorithm to obtain Self-Organizing Maps using fixed neighbourhood Kohonen networks. *Lecture Notes in Computer Science*, **686**, 297–304.

Martinetz, T. (1993). Competitive hebbian learning rule forms perfectly topology preserving maps. *Biological Cybernetics*, **43**, 59–69.

Martinetz, T. and Schulten, K. (1991). A Neural Gas network learns topologies. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402. North-Holland, Amsterdam.

McCulloch, W. C. and Pitts, W. (1943). A logical calculus of the ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133.

Moses, Y., Ullman, S., and Edelman, S. (1993). Generalization across changes in illumination and viewing position in upright and inverted faces. Technical Report CS-TR-93-14, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel.

Mulier, F. and Cherkassky, V. (1994). Learning rate schedules for Self-Organizing Maps. In *Proceedings fo the 12th IAPR International Conference on Pattern Recognition*, volume 2, pages 224–8.

Plunkett, K. and Marchman, V. (1990). From rote learning to system building. Technical Report TR 9020, Center for Research in Language, University of California, San Diego.

Protzel, P., Kindermann, L., Lewandowski, A., and Tagscherer, M. (1998). NIPS '98 workshop on continuous learning: Workshop description. Available online at `http://bfws7e.informatik.uni-erlangen.de/aknn/cont-learn/`.

Quinlan, J. (1982). Semi-autonomous aquisition of pattern based knowledge. In J. Hayes, D. Michie, and Y. Pao, editors, *Machine Intelligence*, volume 10, pages 159–173. Ellis Horwood Limited.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, **1**, 81–106.

Quinlan, J. R. (1993). *C4.5: Progams for Machine Learning.* Morgan Kaufmann.

Reich, Y. (1991). Constructive induction by incremental concept formation. In Y. Feldman and A. Bruckstein, editors, *Artificial Intelligence and Computer Vision*, pages 191–204. Elsevier Science Publishers.

Ritter, H. and Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, **61**, 241–254.

Ritter, H., Martinetz, T., and Schulten, K. (1992). *Neural Computation and Self-Organizing Maps: An Introduction.* Addison-Wesley.

Rose, J. E. and Mountcastle, V. B. (1959). Touch and kinesthesis. In J. Field, H. W. Magoun, and V. E. Hall, editors, *Handbook of Physiology*, volume 1. American Physiological Society.

Rose, J. E., Galambos, R., and Hughes, J. R. (1959). Microelectrode studies of the cochlear nuclei of the cat. *Bulletin of John Hopkins Hospital*, **104**, 211–251.

Rosenblatt, F. (1958). *Principles of Neurodynamics.* Spartan Books, Washington, DC.

Schlimmer, J. and Granger, R. (1986a). Beyond incremental processing: Tracking concept drift. In *Proceedings of the AAAI'86 Conference*, pages 502–507.

Schlimmer, J. and Granger, R. (1986b). Incremental learning from noisy data. *Machine Learning*, **1**, 317–354.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly spaced data. In *Proceedings of the 23rd National Conference of the ACM*, pages 517–523.

Shih, F., J., M., and Chang, F. (1992). A new ART-based neural architecture for pattern classification and image enhancement without prior knowledge. *Pattern Recognition*, **25**(5), 533–542.

Siegelmann, H. T., Horne, B. G., and Giles, C. L. (1997). Computational capabilities of recurrent NARX neural networks. *IEEE Transactions on Systems, Man and Cybernetics B: Cybernetics*, **27**(2), 208. Available electronically at `http://www.neci.nj.nec.com/homepages/giles.html`.

Squire, D. M. (1996). *Model-Based Neural Networks For Invariant Pattern Recognition.* Ph.D. thesis, School of Computing, Curtin University of Technology.

Stanfill, C. and Waltz, D. (1986). Towards memory based reasoning. *Communications of the ACM*, **29**(12), 1213–1228.

Szepesvári, C. and Löincz, A. (1993). Topology learning solved by extended objects: A neural network model. In *Proceedings of ICANN93: International Conference on Artificial Neural Networks*, volume II, page 678, London. Springer.

Takeuchi, T. and Amari, S. (1979). Formation of topological maps and columnar microstructures. *Biological Cybernetics*, **35**, 63–72.

Talbot, S. A. and Marshall, W. H. (1941). Physiological studies on neural mechanisms of visual location and discrimination. *American Journal of Ophthalmology*, **24**, 1255–1264.

Tanaka, S. (1990). Theory of self-organisation of cortical maps: Mathematical framework. *Neural Networks*, **3**, 625–640.

Thrun, S. (1996). Is learning the n-th thing any easier than the first? In D. Touretzky and M. Mozer, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA. MIT Press.

Truong, K. (1991). Multilayer Kohonen image codebooks with a logarithmic search complexity. In *Proceedings of ICASSP91: International Conference on Acoustics, Speech and Signal Processing*, volume IV, pages 2789–2792, Piscataway, N.J. IEEE Service Center.

Vaario, J. and Ohsuga, S. (1992). An emergent construction of adaptive neural architectures. *Heuristics - Journal of Knowledge Engineering*, **5**(2), 1–12.

Villmann, T., Der, R., and Martinetz, T. (1994). A new quantitative measure of topology preservation in Kohonen's feature maps. In *Proceedings of ICNN'94: International Conference on Neural Networks*, pages 645–648, Piscataway, NJ. IEEE Service Center.

Villmann, T., Der, R., Herrmann, M., and Martinetz, T. (1997). Topology preservation in Self-Organizing Feature Maps: Exact definition and measurement. *IEEE Transactions on Neural Networks*, **8**(2), 256–266.

Von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striata cortex. *Kybernetik*, **14**, 212–225.

Wan, W. and Fraser, D. (1993). M2dSOMAP: Clustering and classification of remotely sensed imagery by combining multiple Kohonen Self-Organizing Maps and associative memory. In *Proceedings of IJCNN93: International Joint Conference on Neural Networks*, volume III, pages 2464–2467, Piscataway, N.J. IEEE Service Center.

Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 875–882. Morgan Kaufmann Publishers, Inc.

Widmer, G. and Kubat, M. (1992). Learning flexible concepts from streams of examples: FLORA 2. In *European Conference on Artificial Intelligence*, pages 463–467.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record: Part 4, Computers: Man-Machine Systems*, pages 96–104. Morgan Kaufman.

Willshaw, D. and Von der Malsburg, C. (1976). How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society*, **B**(194), 431–445.

Wilson, H. and Cowan, J. (1973). A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik*, **13**, 55–80.

Woolsey, C. N. (1952). Pattern of localization in sensory and motor areas of the cerebral cortex. In T. biology of mental health and disease, editors, *Handbook of Physiology*. Hoeber, New York.

Zell, A. (1994). *Simulation Neuronaler Netze.* Addison-Wesley, Bonn, Germany.