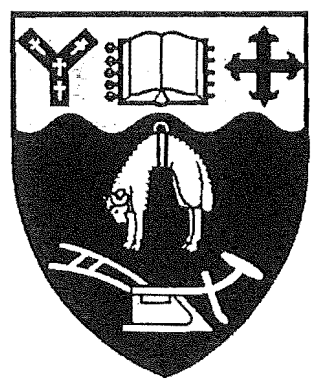
Detection of epileptiform activity in the electroencephalogram using artificial neural networks

Christopher J. James B.Elec.Eng.(Hons)



A thesis presented for the degree of
Doctor of Philosophy
in
Electrical and Electronic Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

February 1997

ABSTRACT

A system for automated detection of epileptiform activity in the electroencephalogram (EEG) has been developed and tested on prerecorded data from a range of patients. Epileptiform activity is manifest as spikes in the EEG and consequently the automated detection of spikes in the EEG is an important tool in the diagnosis of epilepsy and is a goal sought by many researchers. The system presented herein is centred around artificial neural networks (ANNs), in particular the multi-layer perceptron (MLP) and the self-organising feature map (SOFM). The MLP is used in the form of an adaptive filter to enhance the presence of epileptiform transients in the EEG while the SOFM is used to form a novel pattern classifier. A modification to the 'standard' calibration technique for the SOFM is proposed based on a method involving Bayesian probabilities. The SOFM allows a large quantity of EEG data to be used to form a pattern classifier in an unsupervised manner. Fuzzy logic is introduced in order to incorporate spatial contextual information in the spike detection process. By using fuzzy logic it has been possible to develop an approximate model of the spatial reasoning performed by an electroencephalographer (EEGer) as opposed to a precise biological model.

The human brain is overviewed in terms of its structure, organisation and function. Simplistic mathematical modelling of the neural network of the brain is discussed and ANNs are introduced. After reviewing ANNs in general the perceptron based network is introduced and discussed. The SOFM is introduced and through a number of computer simulations several suggestions are put forward regarding the choice of parameters for training the SOFM. After a review of the literature on spike detection systems, in particular ANN based systems, a multi-stage spike detection system is proposed. There are four stages to the system: spike enhancer, mimetic stage, SOFM and fuzzy logic stage. Each stage of the system is discussed at length and measures of performance are indicated at each stage. The importance of spatial and temporal contextual information is discussed and a method using fuzzy logic is proposed to model the spatial reasoning of an EEGer.

The system was trained on 35 epileptiform EEGs containing in excess of 3000 epileptiform events and was tested on a different set of 7 EEGs (6 containing epileptiform activity and 1 'normal') containing 133 epileptiform events. The EEGs consisted of standard clinical recordings with an average length of 22.9 minutes. Preliminary

results show that the system has a sensitivity of 59% and a selectivity of 31% with an average false detection rate of 61 per hour. The performance compares well with other leading systems to be found in the literature once the measures of performance obtained in each are case placed in context. Several aspects in the system have been identified for modification which should lead to considerable improvements in performance (e.g., temporal context, improved mimetic stage).

The new approach to the spike detection problem presented in this thesis shows that it is possible to form an accurate classifier in a self-organised fashion, thus eliminating the need to accurately label large quantities of data – a weak point in many spike detection systems. Furthermore, the importance of spatial contextual analysis is highlighted showing that it is possible to model the spatial reasoning of an EEGer with a fuzzy logic system, thus eliminating the need to produce accurate models of the process.

ACKNOWLEDGEMENTS

During the course of my research and thesis preparation I have received much support, encouragement, advice and assistance from many people and to this end I am deeply grateful to them all.

I would primarily like to thank my supervisors Dr Philip Bones and Dr Richard Jones for their unending guidance and encouragement in my work and for making myself and my family welcome in New Zealand.

I would also like to thank Associate Professor Martin Hagan for introducing me so well to artificial neural networks and for his advice on matters of that ilk.

I thank staff and postgraduates alike at the EEE Department for their help in various aspects of my work. In particular I thank Mike Shurety for putting up with my incessant questions on computing and for greeting me with a smile to boot.

I also thank the staff at the Departments of Neurology and Medical Physics and Bioengineering at Christchurch Hospital for accepting me as one of the team. In particular I thank Grant Carroll for his help on the EEG side of things and Michael Black for pointing me in the right direction on statistical matters.

I gratefully acknowledge receipt of the Commonwealth Scholarship and thank the New Zealand Vice-Chancellor's Committee for funding my stay here.

I thank, and am deeply indebted to, the families Flintoff, Slater and Schembri for making our stay in New Zealand so memorable and making New Zealand a "home away from home".

Last, but not least, I thank my wife Antoinette for accepting to move half way around the world with me only to spend most of the time without my company! For all her help in every step of my work and in preparing this thesis, for putting up with my ever changing moods throughout these years in New Zealand and most of all for believing in me all the time.

This work is dedicated to little Stuart, for making me realise that there is life after Ph.D.

CONTENTS

PREFACE

The work presented in this thesis is the culmination of three years as a Ph.D. student in the Department of Electrical and Electronic Engineering at the University of Canterbury. The work presented has involved collaboration with the Department of Neurology and the Department of Medical Physics and Bioengineering at Christchurch Hospital.

My first exposure to the application of engineering to medicine was in 1991 when Professor Charles Pulé of the Department of Electrical Engineering at the University of Malta introduced me to the basics of reconstruction of images as used in computed-tomography scanners. This led to a final year project for my Bachelor of Electrical Engineering degree in which I performed a simple simulation of the image reconstruction techniques employed in CT-scanners. From that point onward it has been my desire to pursue a career in the field of biomedical engineering. Due to the absence of postgraduate degree opportunities in the EE Department of the University of Malta, in 1993 I applied for, and was subsequently awarded, a Commonwealth Scholarship to New Zealand.

On my arrival in New Zealand in 1994 I enrolled for the degree of Doctor of Philosophy in Electrical and Electronic Engineering under the supervision of Dr Philip Bones and Dr Richard Jones. It was then that I was introduced to the spike detection problem which has occupied much of my time, energy, and thoughts for the past three years!

Work on automated detection of spikes in the EEG had already become well established as a major collaborative project between the Departments of Medical Physics and Bioengineering and Neurology at Christchurch Hospital and the Department of Electrical and Electronic Engineering at the University of Canterbury. As part of their Ph.D. research Dr Bruce Davey and, subsequently, Dr Alison Dingle had developed working spike-detection systems based on the use of a mimetic stage followed by an expert system; encouraging results had been reported on both systems [Davey *et al.* 1989], [Dingle *et al.* 1993]. Alison's research had led to the suggestion that investigating the use of artificial neural networks (ANNs) for the spike detection problem could be a worthy avenue of research; I was keen to pursue this approach.

During my first few months at the University of Canterbury I was fortunate to

attend a series of lectures introducing ANNs by Associate Professor Martin Hagan, a visiting lecturer from Oklahoma State University, U.S.A. This provided a very important starting point for my work.

A more general review of the spike detection problem and spike detection methods reported in the literature was followed by a closer look at methods which focused on the use of ANNs. While getting to grips with the problem, I developed a number of tools using MATLAB to allow the viewing and manipulation of digitized EEG.

My first line of research was centred around a method of adaptively enhancing the presence of epileptiform activity in the EEG and led to the development of a system based on the technique of multi-reference adaptive noise cancelling through the use of ANNs.

On researching the use of ANNs as pattern classifiers it soon became clear that many of the ANN-based solutions to the spike detection problem involved mainly supervised methods and more often than not used perceptron networks. The problem with supervised methods for spike detection is the need to 'label' large quantities of data. The large disagreement amongst EEGers about what constitutes a spike means that there is no real 'gold-standard' and, hence, there is an element of uncertainty in labelling large numbers of spikes. In addition, the perceptron networks, while quite powerful in pattern classification applications, required the choice of many parameters, the values of which could greatly influence the performance.

These two points moved my research towards unsupervised networks and finally to the self-organising feature map (SOFM). The SOFM benefits from training on large quantities of data in a self-organised manner and was particularly appealing for the spike detection problem. At this point I set about gaining a better understanding of the SOFM and its training parameters through a number of simulations. This resulted in my drawing up a number of suggestions for training the SOFM. This work was then applied to training a number of SOFMs using EEG data.

Another important point made by Alison in her thesis was that considerable use must be made of spatial and temporal contextual information in order for a spike detection system to follow the reasoning of an EEGer. With this in mind, while working on the SOFM based classifier I began to investigate the possibilities of combining spatial and temporal information in the spike detection system that was beginning to emerge. This led to the use of fuzzy logic and the derivation of an experimental fuzzy rule-base in order to combine the spatial information of the multi-channel EEG recording. Due to time constraints, the investigation of utilising temporal contextual information is limited to a proposed system which is intended to further enhance the overall spike detection system developed.

The software for the EEG manipulation tools, as well as for each stage of the system, has been developed in MATLAB. The SOFM training algorithms and the

overall spike detection system, however, were finally implemented in 'C'. Through the use of 'C', it was possible to obtain 'real-time' operation of the spike detection system – an important secondary goal for the project.

This thesis is divided into three main parts. The first part provides an introduction to the human brain and the EEG, giving a historical overview of the discovery of the EEG, recording techniques and its use in the diagnosis of epilepsy (Chapters 1 and 2).

The second part introduces ANNs. Following a brief historical overview of their development, it discusses the different architectures and learning paradigms of the most important networks in the literature (Chapter 3). Perceptron networks are discussed in greater detail in Chapter 4, along with the error-backpropagation algorithm most commonly used for their training. Methods for enhancing the performance of the system and for obtaining the optimal network architecture for the task at hand are also discussed. Chapters 5 and 6 focus on the SOFM, proposed by Kohonen in 1981 and employed here for the first time for detecting spikes in the EEG. Chapter 6 presents the results of a number of computer simulations carried out in order to obtain a better understanding of the various training parameters needed to train a SOFM for a given problem.

The third and final part deals with the spike detection problem in the EEG and the development of an automated system based mainly on the use and attributes of ANNs. Chapter 7 reviews the spike detection problem and the diverse approaches in the literature aimed at providing a solution. A multi-stage system is then proposed based around an ANN approach and implementation. The first stage is presented in Chapter 8 and describes a method to enhance the presence of epileptiform activity in the EEG based upon an ANN implementation of multi-reference adaptive noise cancelling. Chapters 9, 10 and 11 then proceed to describe various aspects of the spike detector/classifier system in detail. Chapter 9 presents the two single-channel stages — mimetic and SOFM — of the system. Chapter 10 presents the spatial-combiner stage of the system which makes use of valuable multi-channel cues (i.e., spatial context) implemented by fuzzy logic, to enhance performance. Chapter 11 assesses the performance of the overall system and compares the results obtained to systems described in the literature.

The thesis then concludes with a discussion on the relative merits of ANNs and their application to the spike detection problem. Areas which could benefit from future research are also mentioned.

During the period of doctoral study, I have presented the work herein at the following meetings:

1. James, C. J., Hagan, M. T., Jones, R. D., Bones, P.J. and Carroll, G. J., 'Neural

network based spatio-temporal filtering of the EEG via multireference adaptive noise cancelling', *Engineering and Physics in Medicine*, Queenstown, New Zealand, November 20–24, 1995.

2. James, C. J., Hagan, M. T., Jones, R. D., Bones, P.J. and Carroll, G. J., 'Spatio-temporal filtering of the EEG via neural network based multireference adaptive noise cancelling', *3rd New Zealand conference of Postgraduate Students in Engineering and Technology*, Christchurch, New Zealand, July 1–2, 1996.

3. James, C. J., Jones, R. D., Bones, P.J. and Carroll, G. J., 'The self-organising feature map in the detection of epileptiform discharges in the EEG', *3rd New Zealand conference of Postgraduate Students in Engineering and Technology*, Christchurch, New Zealand, July 1–2, 1996.

4. James, C. J., Hagan, M. T., Jones, R. D., Bones, P.J. and Carroll, G. J., 'Neural network based spatio-temporal filtering of the EEG', *Canterbury Medical Research Society*, Christchurch, New Zealand, July 11, 1996.

5. James, C. J., Hagan, M. T., Jones, R. D., Bones, P.J. and Carroll, G. J., 'Spatio-temporal filtering of the EEG via neural network based multireference adaptive noise cancelling', *Proceedings of the 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Amsterdam, 31 October – 3 November, 1996.

   (Finalist in IEEE/EMBS Whitaker Student Paper Competition representing Region 10).

6. James, C. J., Jones, R. D., Bones, P.J. and Carroll, G. J., 'The self-organising feature map in the detection of epileptiform transients in the EEG', *Proceedings of the 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Amsterdam, 31 October – 3 November, 1996.

7. James, C. J., Jones, R. D., Bones, P.J. and Carroll, G. J., 'The detection of epileptiform activity in the EEG: An artificial neural network based system', *Signal processing research group, EE Department, University of Malta*, Malta, 6 November, 1996.

The following has been accepted for publication:

1. James, C. J., Hagan, M. T., Jones, R. D., Bones, P.J. and Carroll, G. J. (1996), 'Multireference adaptive noise cancelling applied to the EEG', accepted by: *IEEE Transactions on Biomedical Engineering.*

The following is in press:

1. James, C. J., Hagan, M. T., Jones, R. D., Bones, P.J. and Carroll, G. J., 'Neural network based spatio-temporal filtering of the EEG', in press: *New Zealand Medical Journal.*

It is planned to submit papers on the following:

1. 'The Bayesian approach to calibrating the self-organising feature map.'
2. 'Spike detection in the EEG: incorporating spatial information with fuzzy logic.'
3. 'A multi-stage system to detect epileptiform activity in the EEG.'

Chapter 1

THE HUMAN BRAIN

1.1 INTRODUCTION

The human brain is an organ that enables man to survive in an ever changing environment. Signals from the environment are received and processed by the brain and appropriate actions are taken in response. It also stores information, both for the short term, and for the longer term, allowing relevant information to be extracted at will.

This highly complex organ is still relatively poorly understood, although research into the inner workings of the brain can be traced back to the time of the ancient Greeks of around 500 BC. Understanding how the brain works is now an interdisciplinary task, in which an increasingly large number of disciplines address different aspects of the same problem. For example, neuroanatomists study the various components of the brain and how they are connected together, while neurophysiologists examine the operation of these components. The higher functions of the brain (e.g., language, memory, etc.) are the field of study of psychologists and philosophers. The advent of high speed digital computers, along with new mathematical techniques, means that engineers have become more directly involved in brain research through measuring, analysing and modelling brain activity and function.

In this chapter, a brief overview of brain structure and organisation is given, along with a some brief notes on brain functions and disorders. Unless otherwise specified, the major sources of information for this chapter are Clarke and Dewhurst [1972], Coen [1985] and Dingle [1992].

1.2 THE BRAIN

The brain in humans consists of a watery mass weighing about 1400 g, divided by delicate membranes into a huge number of compartments. Each compartment contains very complicated chemical and electrical systems, which are in a perpetual state of

change, fuelled by energy derived from respiration. If these changes cease, we first lose consciousness and then quickly die.

1.2.1 Basic building blocks

The major components of the brain are the many cells, the most important being the nerve cells or *neurons*. Neurons, which only make up about 10% of the cells in the brain [Thompson 1967], are cells specialised for the task of processing and communicating information. The 3 main components of the neuron are the (a) *dendrites* through which most input signals are received, (b) the *soma* or cell body which collates the inputs from the dendrites and sums and thresholds the inputs, and finally (c) the *axon* along which the output is transmitted to other neurons (see Figure 1.1). Each neuron has a membrane potential due to the differences in ion concentrations in the intracellular and extracellular fluids (such as Na^+, K^+, Cl^-). At rest, the membrane potential is typically -70 mV, and it is certain changes in the membrane potential at the origin of the axon that determines whether an *action potential*, or electrical output pulse, is carried along the axon to other surrounding neurons. Information across the junction between neighbouring axons and dendrites is transmitted by way of uni-directional chemical interfaces called *synapses*. When an action potential reaches a synapse, a chemical substance known as a *neurotransmitter* is released, which crosses the synapse and activates *receptors*, which are special molecules attached to the membrane of receiving (or *post-synaptic*) neuron. A post-synaptic potential is thus induced in the membrane potential of the post-synaptic neuron, this *post-synaptic potential* may be either *excitatory* or *inhibitory*. The strength of synapse coupling is variable and it is this variability that facilitates the important learning and memory effects seen in the brain.

A neuron may receive tens of thousands of inputs by way of the synapses and the resultant post-synaptic potentials are integrated (both spatially and temporally) to produce a change in the membrane potential. When the membrane potential just before the axon exceeds a threshold value (typically around -20 mV), an action potential is generated and transmitted along the axon to other neurons. Once a neuron has 'fired' it needs a finite time – the *absolute refractory period* (1–2 ms) – to recover, during which the neuron is unable to produce an action potential, regardless of the strength of the inputs. This is then followed by a *relative refractory period* of 5–7 ms during which the neuron operates with an elevated threshold, before resuming normal activity [Amit 1989].

It is the *firing frequency* of a neuron that is often measured by neuroscientists and it appears that the only significant expression of information in a train of pulses in a nerve signal is the frequency of action potentials.

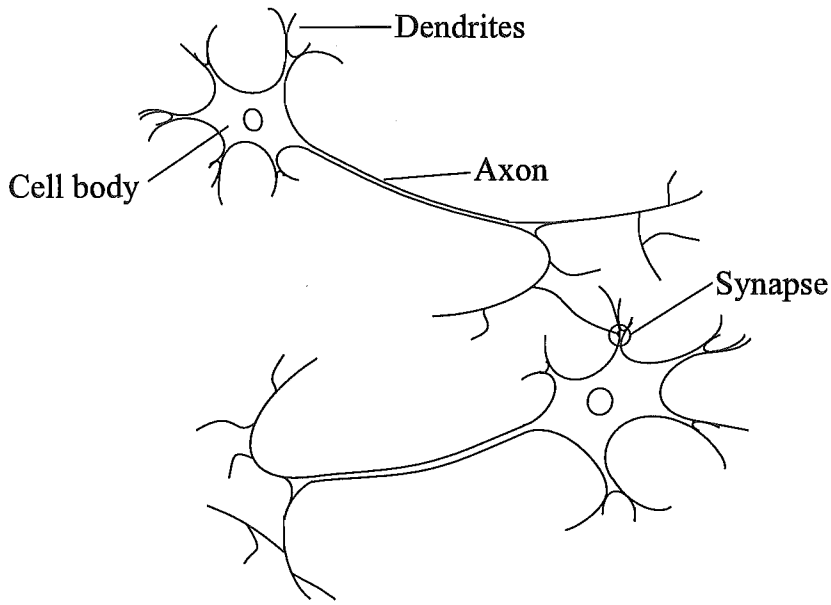


Figure 1.1 A simplified schematic of neurons.

Most of the remaining 90% of the cells in the brain are collectively termed *glial cells*. These surround, support and nourish the neurons. In particular: the *astrocytes* provide neurons with physical and nutritional support, the *microglia*, along with some astrocytes, remove neurons which have died as a result of injury or from 'old age' and the *oligodendroglia* produce the *myelin* sheath which insulates the axons and thus prevents messages from interfering with each other [Carlson 1986].

1.2.2 Protection of the brain

The brain is the most protected organ of the body, the greatest protection being provided by the skull surrounding the brain. However, protection is also provided by the *meninges* or protective sheaths around the brain. The meninges consist of three layers, as shown in Figure 1.2. The outer layer is known as the *dura mater* and is thick and tough, yet flexible. The middle layer is known as the *arachnoid membrane* and is soft and spongy. Finally, the *pia mater*, which is the innermost layer, is closely attached to the brain and follows every convolution of its surface. A gap between the arachnoid membrane and the pia mater, known as the *subarachnoid space*, is filled with *cerebrospinal fluid*. This fluid is responsible for absorbing much of the shock caused by sudden head movements.

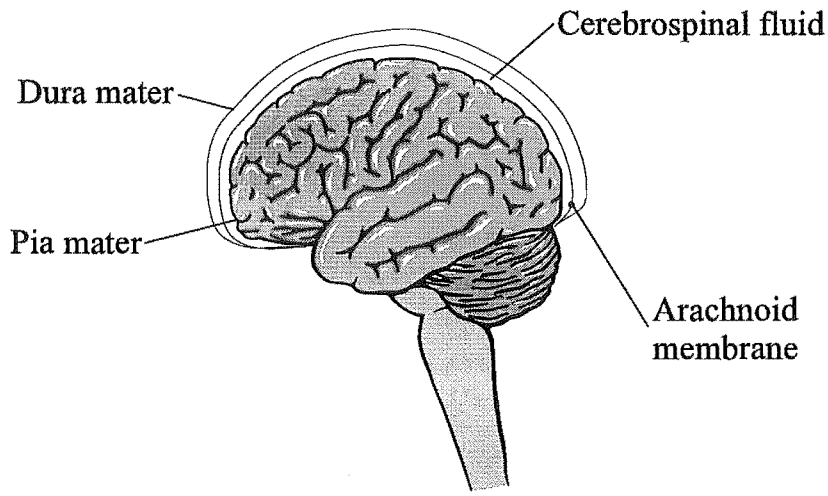


Figure 1.2 The protective layers of the brain.

1.3 THE ORGANISATION OF THE BRAIN

The 10^{11} or so neurons which make up the human brain are grouped together in an organised fashion. The brain can be split into three major regions: the *cerebrum*, the *brain stem* and the *cerebellum*. (See Figure 1.3).

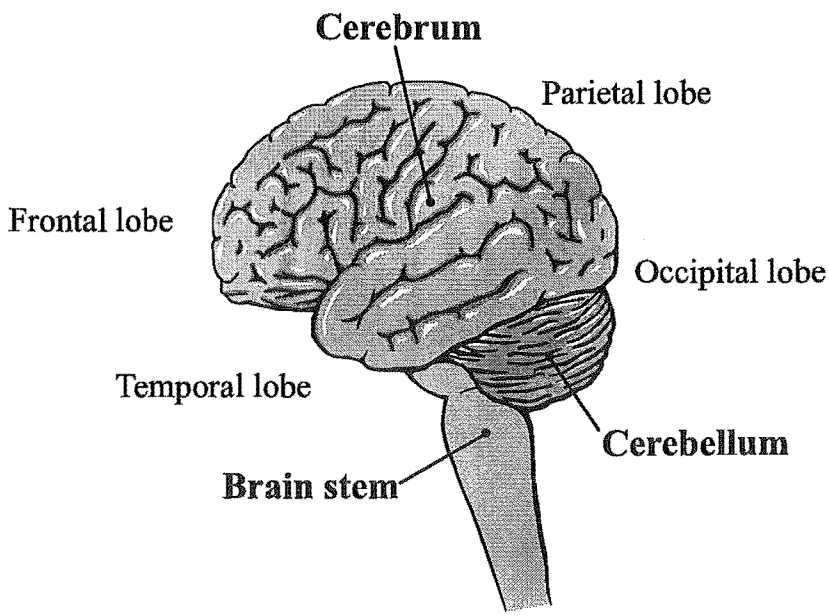


Figure 1.3 The major regions of the brain.

1.3.1 The cerebrum

The cerebrum, the largest of the three regions, is divided into two almost equal halves known as the *left* and *right cerebral hemispheres.* These two hemispheres are interconnected by several *commissures*, the largest and most important of which is the *corpus callosum.*

The outer portion of the cerebrum is 3–4 mm thick and is known as the *cerebral cortex.* The cerebral cortex consists of a large number of *sulci* (small grooves), *fissures* (larger grooves) and *gyri* (bulges between adjacent grooves). These convolutions essentially mean that the cerebral cortex has quite a large surface area. In fact, there is a general correlation between the surface area of the cerebral cortex of an animal and the complexity of its behaviour [Hilgard *et al.* 1979]. The surface of each cerebral hemisphere is divided into four lobes: the *frontal lobe*, the *temporal lobe*, the *parietal lobe* and the *occipital lobe* (see Figure 1.3).

- ***frontal lobes*** – The frontal lobes are specialised for the planning, control and execution of movements, as well as several cognitive and personality functions. In particular, the *primary motor cortex* contains neurons that participate in the control of movement. Studies have shown that the movement of particular parts of the body is controlled by particular parts of the motor cortex. In fact, the body is effectively mapped onto the motor cortex, with disproportionately large areas dedicated to movement of the muscles involved in speech and to the fingers. Damage to the frontal lobe may also cause changes in personality [Walsh 1978].

- ***parietal lobes*** – The parietal lobes are concerned with somatic sensation and perception, as well as intersensory or cross-modal association. In particular the *primary somatosensory cortex* receives information from the somatosenses (touch, pressure, pain, temperature, body position and vibration). The body is mapped onto the somatosensory cortex in a similar way as onto the motor cortex.

- ***occipital lobes*** – The occipital lobes are specialised for vision, containing the primary visual cortex, which receives sensory signals from the retina, and are involved with visual perception and visual memory.

- ***temporal lobes*** – The *temporal lobe* contains the *primary auditory cortex* which receives sensory information from the receptors of the inner ear. Auditory signals are mapped onto the auditory cortex according to their frequency (*tonotopic mapping*). The temporal lobes are also concerned with olfaction, the integration of visual experience with other forms of sensory information, and memory.

Although the hemispheres give the appearance of symmetry, each hemisphere has its own specialised talents. The somatosensory and motor regions of each hemisphere

are concerned mainly with the opposite or *contralateral* side of the body. Similarly, in the visuosensory system the right occipital cortex processes the left half of the visual field, while the left occipital cortex processes the right half of the visual field. Other, more specialised functions, are often represented asymmetrically in the brain. Linguistic abilities, for example, tend to reside mainly in the left hemisphere, while the right hemisphere appears to be important for the recognition of emotion, perception of melodies and the recognition of complex non-verbal visual patterns [Walsh 1978]. Figure 1.4 shows the mapping of the specialised functions handled by the different regions of the cerebral cortex.

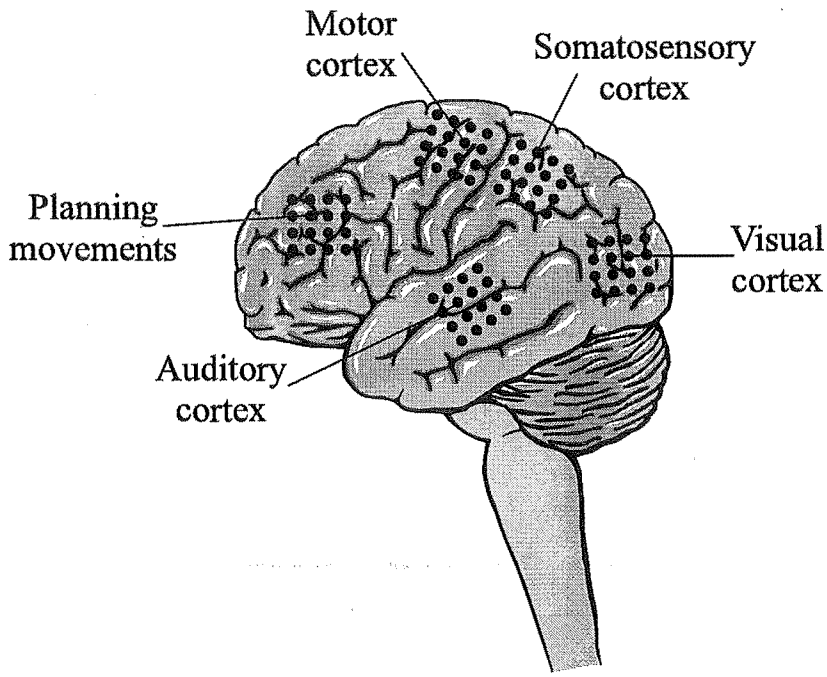


Figure 1.4 Mapping functions onto the cerebral cortex.

The cerebral cortex is often referred to as *grey matter*, due to the large concentration of glia, cell bodies, dendrites and interconnecting axons of neurons which gives the cortex a greyish brown appearance. Beneath the cerebral cortex run millions of axons which connect the neurons of the cerebral cortex with those of the subcortical structures. The large concentration of myelin gives this tissue an opaque appearance, whence comes the name *white matter*.

The subcortical structures include: the *basal ganglia*, which forms an important part of the motor system; the *thalamus*, which relays sensory information to the cerebral cortex; the *hypothalamus*, which organises behaviours related to survival (e.g., fleeing, feeding, fighting and mating) the *limbic system*, which is involved in emotional behaviour, motivation and learning; the *hippocampus* is an important structure of the limbic system which appears to play a major role in learning and memory [Carlson 1986].

1.3.2 The brain stem

The brain stem contains structures that control arousal and sleep as well as vital functions, such as respiration and the cardiovascular system.

1.3.3 The cerebellum

The cerebellum receives a large amount of sensory information which it uses to exert a coordinating and smoothing effect on movements.

1.4 FUNCTIONS OF THE BRAIN

It is the human brain that allows us our capacities to live out a conscious life. For example, at the occicipit are areas especially concerned with vision. Others in the temporal region deal with hearing and higher up are those for touch. Further areas are necessary for speech, both for production and for understanding. The Russian neurosurgeon Luria has emphasised how injury to the frontal lobes produces, among other things, defects of intention, when patients are unable to initiate action and are liable to sit for hours passively doing nothing [Luria 1973].

It is important to recognise that all living things have a purpose or aim, essentially to continue life of the individual or the species. The brain is geared to controlling or handling all the functions that make life possible, most of which occur without our conscious knowledge. These functions include breathing, a rhythm which is controlled by the neurons in the lower parts of the brain, and other long term rhythms such as that of sleep. In the higher parts of the brain more complicated tasks are catered for, such as consulting the memory when we recognise a face or find answers to questions, recognise speech, etc. It is important to note that these activities do not proceed in a random manner but are all directed or aimed by the overriding goal of the continuation of life, or homeostasis. These aims stem from the actions of relatively small central parts of the brain, still little understood. The *reticular system* serves to activate both what comes in and goes out, keeping the whole organisation going. These areas, therefore, set the *aims* of the whole individual, they establish the needs. The satisfaction of such needs provide the experience of pleasure; Olds and Milner first showed in 1954 how rats with electrodes placed in such centres will quickly learn to press a lever that gives the pleasure of this stimulus [Olds and Milner 1954].

1.5 DISORDERS OF THE BRAIN

For many people, the brain is an organ which functions quite effectively for most of their life, 60 – 70 years on average. However, defects do sometimes arise in its structure or in its electrical and chemical processes. This can be due to many causes, including accident or some hereditary defect. In order to adequately treat brain disorders, an understanding of its mode of operation is essential. An example is Parkinson's disease, which is a progressive chronic condition characterised by muscle rigidity and involuntary tremor. Research has revealed that the disease is caused by the progressive destruction of dopamine pathways in the brain; this discovery has led to an effective treatment to be developed to adequately treat the cause of the illness [Kety 1979].

Epilepsy is a condition characterised by recurrent seizures associated with a disturbance of consciousness. In many cases, the cause and mechanisms of seizure activity are not understood, yet, surprisingly, the condition can often be effectively treated with anticonvulsant drugs, such as Dilantin. If drugs fail, surgery may be performed to remove the source of seizure activity, if that can be accurately localised.

One might think that due to the present limited understanding of the functioning brain, treatment of many disorders would not be possible. However, empirical treatments are often surprisingly successful. Treatments for schizophrenia and depression, for example, were discovered quite by accident, without knowledge of their effects on the brain.

Paradoxically, it is unfortunate that it is disorders of the brain which provide a valuable source of information about the operation of the brain. The oldest and still most widely used approach to the problem of brain function is to analyse the effects of lesions to areas of the brain. However, conclusions from studies of this kind need to be made carefully, in order to avoid incorrect conclusions being drawn.

Imaging techniques, such as CT, MRI and PET, are revealing both structural and physiological differences between brains of certain patients and those of the normal population. In particular functional MRI is helping to locate areas of the brain specific to particular tasks with a high degree of accuracy [Singh *et al.* 1996].

Chapter 2

THE ELECTROENCEPHALOGRAM

2.1 INTRODUCTION

The electroencephalogram (EEG) is a recording of the electrical activity in the brain, generally measured at the scalp. Although the origin of the EEG is not completely understood, it is generally accepted that it represents the averaged activity of many millions of neurons in the brain, after being filtered by layers of fat, bone and cerebrospinal fluid. This aside, the EEG has been found to be a valuable tool in the diagnosis of numerous brain disorders. Nowadays, the EEG recording is a routine clinical procedure and is particularly useful in the investigation of epilepsy.

This chapter first gives a brief overview of the history and origin of the EEG before reviewing standard recording techniques. Then what constitutes *normal* EEG is discussed along with how that can be contaminated with artifacts. Finally the abnormal EEG and what makes an EEG abnormal, is discussed along with the application of the EEG as a diagnostic tool with particular reference to the diagnosis of epilepsy. Unless otherwise specified, the major sources of information for this chapter are Spehlmann [1981], Empson [1986] and Dyro [1989].

2.2 THE HISTORY AND ORIGIN OF THE EEG

2.2.1 History of the EEG

The development of electroencephalographic (EEG) analysis, like that of other electrophysiological techniques, was dependent on advances in the science of electromagnetism finding appropriate applications in the mid- to late nineteenth century. However, there were earlier attempts at investigating the relationship between electricity and living organisms during the eighteenth century. Louis XV 'caused an electric shock from a battery of Leyden jars to be administered to 700 Carthusian monks joined hand to hand, with prodigious effect'. More reliable experiments had to await the invention

of reliable sources of electric current. Volta, with the help of his newly invented battery cell (the Voltaic pile), gave the opportunity of electrical stimulation to those early researchers interested in the effects of electricity on living organisms.

Following Volta's invention came the development and elaboration of electromagnetic principles. Insight by Faraday as early as 1831 allowed the immediate development of sensitive electrical instruments, such as the string galvanometer, which was standard equipment for the early electroencephalographers.

Despite the crudeness of the equipment available, and the difficulties in recording the electrical changes in living tissue, Du Bois-Reymond, working in Berlin in 1848, discovered the standing voltage between the surface and the cut end of a nerve, and also the sudden variation in response to a stimulus, now known as the 'action potential'. So the apparatus and techniques now existed to record from animals' brains and also to stimulate living preparations. Two Prussian medical officers, Fritsch and Hitzig, in 1870 took advantage of the opportunity offered by the Franco-Prussian War to study the exposed brains of soldiers struck down on the battlefield. They discovered that Galvanic stimulation of some parts of the cortex caused movements in the contralateral limbs, and that these movements could be reliably elicited by stimulation at the same place.

So, it was now obvious that with sufficiently sensitive recording techniques it should be possible to map the sensory cortex. This task was taken up by Richard Caton, working in Liverpool, who in August 1875 published a report in the *British Medical Journal* which stated [Caton 1875]:

> 'In every brain hitherto examined, the galvanometer has indicated the existence of electric currents. The external surface of the grey matter is usually positive in relation to the surface of a section through it. Feeble currents of varying direction pass through the multiplier when the electrodes are on two points of the external surface ... '

Caton had thus discovered spontaneous EEG in animals and also showed that it was possible to detect electrical brain responses to stimuli and located the visual cortex in the occiput, or rear of the head. He was unable to find any location specifically responsive to sound stimulation. The observation of spontaneous electrical activity at the surface of the cortex may not have meant much at the time but it was the discovery of the electroencephalogram.

This work was duplicated by Adolf Beck, a Polish scientist, in 1890 [Beck 1890]. He also discovered spontaneous oscillations of voltage from the occipital cortex which disappeared with light stimulation but not with noise. (This seems analogous with the phenomenon of alpha blocking found in humans some 40 years later).

Since many of the techniques in use at around this time were quite crude, it is not surprising that many researchers were quite reluctant to publish any findings in this

field. Until a means was found of permanently recording the EEG, the spontaneous activity shown by the cortex could only be regarded as a curiosity.

It was Hans Berger who in 1929 first published the recorded electroencephalogram of his young son, measured four years earlier [Berger 1929]. Berger discovered the alpha rhythm, running at 10 Hz, and also discovered that it disappeared if the eyes were opened, with mental effort and with loud noises or painful stimuli. Berger's work was disregarded by physiologists at the time, party because he published in psychiatric journals, and also perhaps because of his reputation for eccentricity. Only after his work was replicated by Adrian and Matthews in Cambridge [Adrian and Matthews 1934] did he get the credit he deserved for having established human electroencephalography.

2.2.2 Origin of the EEG

There have always been doubts as to whether the EEG actually originates in the cortex. Berger, when recording from the scalps of patients with trephined holes in the skull, found that the alpha rhythm was always greater when one of the electrodes was placed over the hole, and greatest when a needle pierced the scalp above the opening [Berger 1929]. He took this as evidence that the EEG did not originate in the scalp. He made sure that subjects remained still during recording so as to eliminate eye movements and other movements as a cause.

Adrian and Matthews [1934] found that movements of the eyes did not affect the voltages recorded between the occiput and the vertex and the only recordable effects were greatest around the eyes themselves (such as those produced by blinking). Adrian and Yamigawa [1935] used a cadaver to confirm that electrodes at the scalp surface could indeed be used to pick up signals emanating from an artificial dipole generator placed inside the brain.

There have periodically been suggestions that waves such as the alpha rhythm are artifactual and not generated by the brain itself. One good reason for this is that it is difficult to conceive how such regular sinusoidal waves could be generated by nerve cells whose own electrical activity has always been regarded as being exclusively spikey. Adrian and Matthews suggested that the alpha rhythm represented a spontaneous beat in a group of cortical neurons, that the neurons depolarised almost simultaneously and that the gross EEG represents the envelope of the activity of the underlying tissue. The gross EEG, recorded from the other side of a centimeter of bone, skin, fat and cerebro-spinal fluid, was thought to represent a blurred and averaged picture of the sum of the activity of many individual cells.

However, Elul [1972] implanted microelectrodes in the brains of experimental animals and found that wave-like EEG activity could be recorded when the electrodes were only 30 μm apart. Elul continues to say that 'In fact, decreasing the inter-electrode sep-

aration had no appreciable effect on the wave activity recorded differentially between them. These results can be interpreted only as a contradiction of the Adrian-Matthews model'.

Other suggestions have been put forward by many researchers: the EEG may be due to the slow changes in neuronal synaptic potential [Purpura 1959]; or it may be due to the pyramidal neurons in the underlying cortex, kept in synchrony by generators in the thalamus [Andersen and Andersson 1968]. Put simply, neurophysiologists do not know exactly *how* the human EEG is generated. There is no doubt, however, that electrical activity originating in the human cortex can be measured from the scalp.

2.3 RECORDING THE EEG

The modern EEG machine is a far cry from the crude string galvanometers used by Berger. It consists entirely of electronic amplifiers to provide simultaneous, multichannel recordings which are recorded on paper or on some other storage device (tapes, disk drives, etc.). Multichannel recordings usually comprise 8, 16, 19 or 32 channels of EEG.

2.3.1 Electrode placement

Recording takes place across a number of electrodes, generally placed on the scalp. Scalp electrodes are cup shaped and are applied to the cleaned scalp with a conductive paste and are held in place by a rubber head-set or some kind of adhesive (e.g., collodion). Another type of electrode is the needle or *dermal* electrode, which may be inserted into the scalp. Other types of electrodes are nasopharyngeal electrodes, which are long S-shaped plastic coated wires with a silver ball at the tip inserted via the nose and advanced until they make contact with the nasopharynx, and sphenoidal electrodes which are inserted under the brain via an entry near the ear. These type of electrodes may reveal specific activity when routine scalp recordings appear normal. Finally, intracranial depth-needle electrodes are sometimes used intraoperatively to locate foci of epileptic activity.

The standard method for scalp electrode localisation is the 10–20 electrode system, so called because it is derived by 10% and 20% measurements relative to four scalp landmarks [Jasper 1958]. The four scalp landmarks are: (1) the bridge of the nose (the nasion), (2) the bump at the back of the head immediately above the neck (the inion) and (3) & (4) the left and right preauricular points (depressions above the angle of the cheekbone just in front of the ear). The system derives 19 locations on the scalp, of which right-sided electrodes are even numbered and left-sided electrodes are odd numbered. Letters preceding the numbers refer to cortical regions. Frontal is

F, prefrontal is *Fp* (or *frontopolar*), parietal is *P*, temporal is *T*, central is *C*, and occipital is *O*. Electrodes along the midline have no number, only the letter *z* (e.g., *Fz* and *Cz*, with *Cz* being the vertex). When nasopharyngeal electrodes are used, they are designated *NP* or *PG*. Figure 2.1 depicts the 10-20 system of electrode localisation and the numbering system used to label each electrode.

2.3.2 Montages

Electrodes are connected together under different *montages* (montage is the French word for "array"). In general, montages can be divided into two types: *referential* and *bipolar*. Most of the information obtained with one type of montage can be obtained with the other, but each montage has some shortcomings which can be overcome to a certain extent through the use of the other. A combination of both types is used routinely in recording the EEG for this reason. Note that it is possible to extract bipolar EEG from that originally recorded using a referential montage but not *vice versa*.

2.3.2.1 Referential Montage

Referential montages connect one input of each differential channel to electrodes placed at various parts of the head and the remaining input for each channel to a common electrode placed in an area considered to be electrically quiet; this electrode is called the 'reference electrode'. In this way, the voltages between the various points on the head and the reference electrode are recorded. Figure 2.2 shows a referential montage recording the voltage at a number of scalp electrodes. In the referential montage, the origin of this voltage on the head is recognised by the amplitude, i.e., the channel which records the voltage of highest amplitude is connected to the electrode nearest the source of the voltage. If the output of two channels is of equal amplitude, the source of the voltage is considered to be located approximately equidistant from each of the two electrodes connected to the inputs of these channels.

The advantage of a referential montage is that the recording can give an undistorted display of the shape of voltage changes and is especially useful for recording voltages which have a wide distribution.

The disadvantage of a referential montage is that it is often impossible to find a reference electrode which is entirely electrically 'quiet' or inactive. Sources located near the reference electrode may generate signals on all channels connected to that electrode.

Some standard referential montages are listed below and are described in more detail in Appendix A.

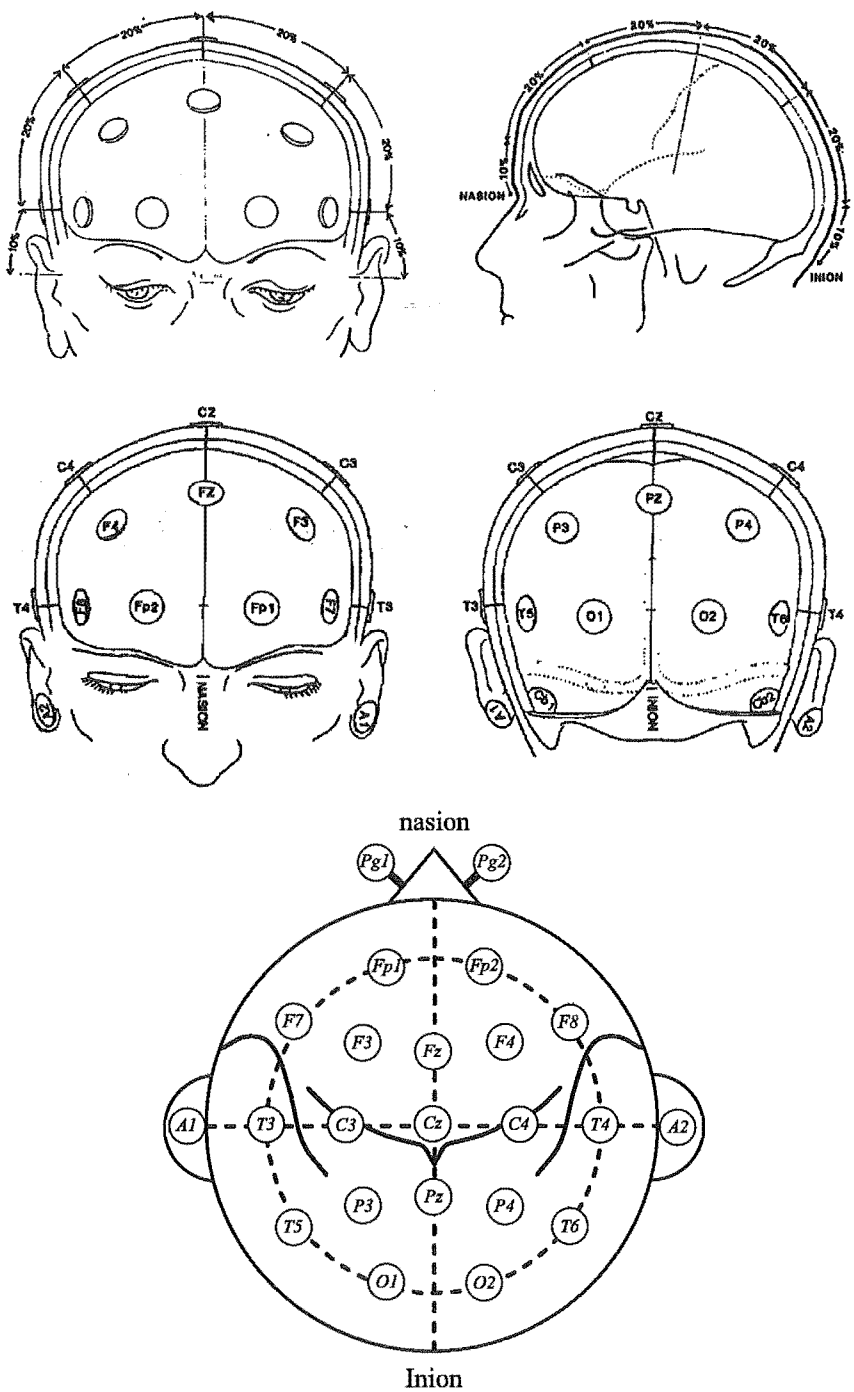


Figure 2.1 The 10–20 system of electrode localisation (from Jasper [1958]).

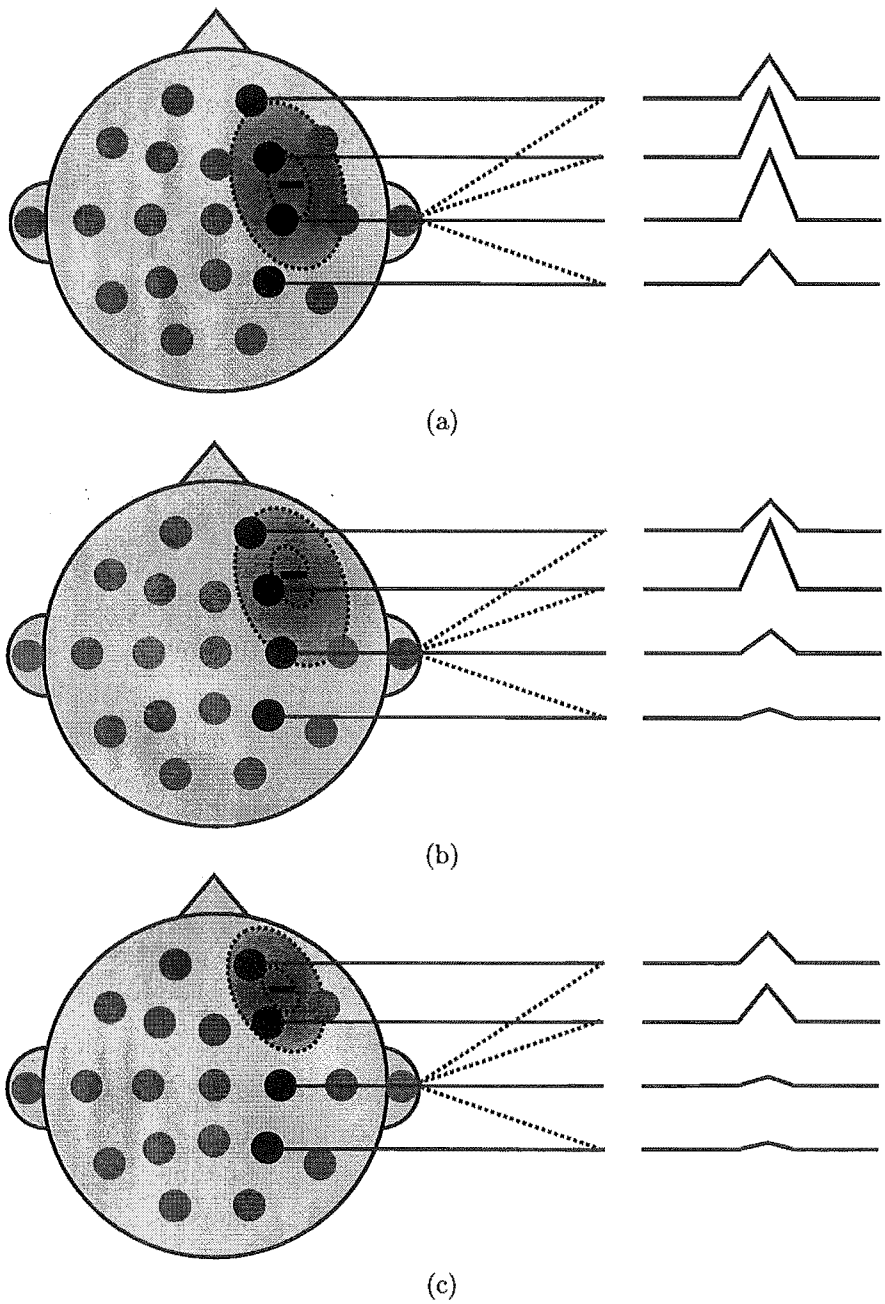


Figure 2.2 Recordings made using a referential montage. The polarity of the waveform at each channel is determined by subtracting the electrode potential from the reference potential.

Average: the reference signal is composed of the average of all of the recorded EEG signals.

Ipsilateral ear-reference: the reference electrode used is the ipsilateral ear-electrode for each group of electrodes.

Linked ear-reference: both ear electrodes are linked and serve as the reference channel.

Averaged ear-reference: the average of both ear electrode signals are used as the reference channel.

Vertex: the vertex electrode *Cz* is used as the reference channel.

2.3.2.2 Bipolar Montage

With a bipolar montage, different pairs of electrodes are connected to the differential inputs of each channel amplifier. The best bipolar montages link electrodes in chains so that an electrode which is serving as a reference for one channel becomes an input for the next channel, and so on down the chain (Figure 2.3). With this type of arrangement the location of a source is detected, more by the direction of the deflection of the recording rather than by the amplitude. A peak in the potential field located at an electrode which is common to two amplifiers in a bipolar chain causes deflections in the EEG recording of opposite direction, or what is known as *phase reversal* at the output of these amplifiers. Figure 2.3 shows bipolar montage recordings for different potential fields featuring a region of relatively negative potential. It should be noted that the phase reversal does not indicate a reversal in the polarity of the voltage but only a reversal in the direction of deflection due to the physical setup of the channel amplifiers. A region of relatively positive potential would give rise to a phase reversal in the opposite direction to the case of a negative region. If the maximum of the voltage at the scalp occurs between two of the recording electrodes (Figure 2.3a) then no deflection is recorded at that channel connected to both the electrodes. However, due to the chain nature of the connections, adjacent channels will show a phase reversal. As a rule, phase reversals due to a region of negative surface potential cause deflections in channels "towards" each other and, in the case of a region of positive surface potential, "away" from each other (based on standard in neurophysiology whereby negative-going is *upwards*).

The advantage of using bipolar montages is that voltage changes in neighbouring electrodes can be sharply distinguished and thus can localise regions of activity more precisely than can referential montages.

The disadvantage of bipolar montages is that they distort the waveshape and amplitude of voltages which are distributed widely and affect both recording electrodes

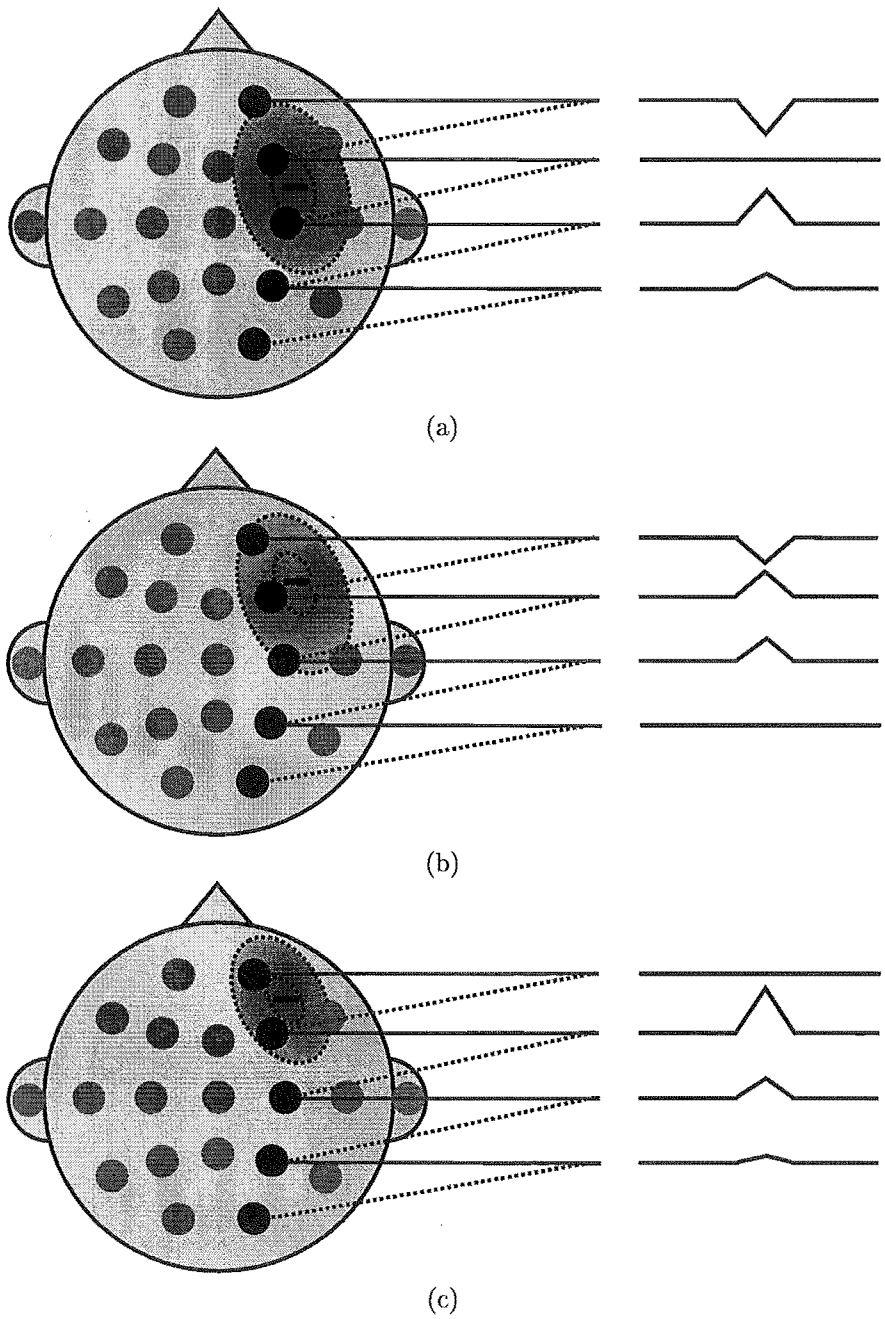


Figure 2.3 Recordings made using a bipolar montage. The polarity of the waveform at each channel is determined by subtracting the electrode potential from the potential at the next electrode in the bipolar chain.

connected to one channel. Also voltages between the last two electrodes in a chain may produce confusing recording deflections and widespread abnormal voltages may be missed.

Some of the more common bipolar montages include Longitudinal, Transverse, Longitudinal-Transverse and Circumferential. These are described in more detail in Appendix A.

Figure 2.4 shows how the same potential field recorded at the scalp using both referential and bipolar montages can be depicted in terms of phase reversal or not, as the case may be. What is immediately apparent is how the localisation of the centre of negativity can be obtained (a) by the maximum amplitude deflection of the recording in the referential montage case and (b) by observing the phase reversal in adjacent channels in the bipolar montage case.

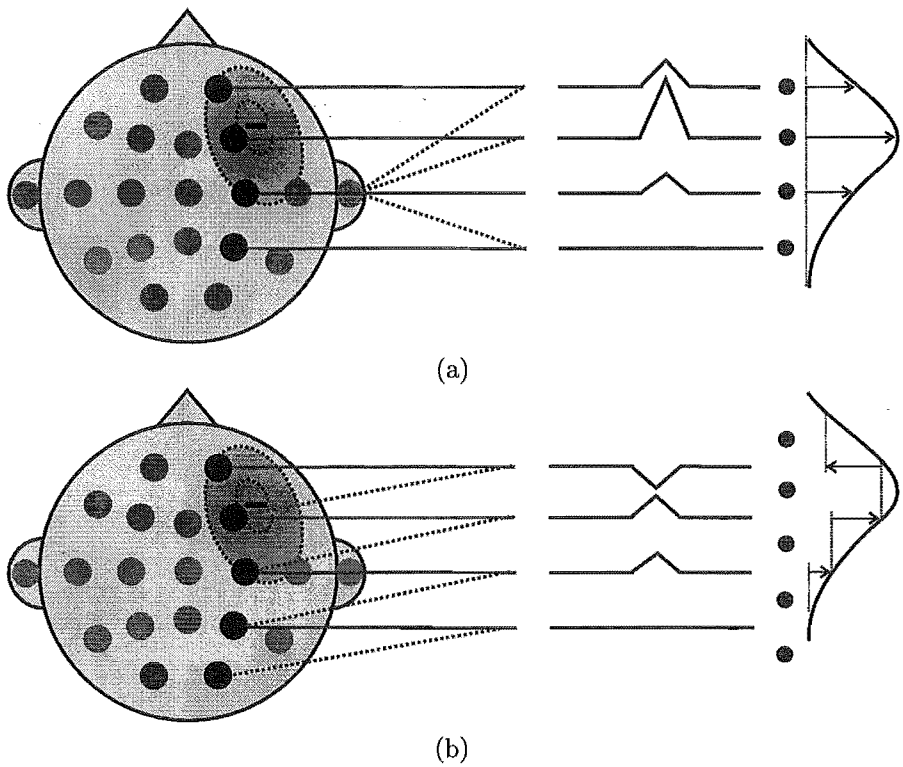


Figure 2.4 Plotting EEG recording deflection using (a) referential and (b) bipolar montages.

2.3.3 The electroencephalograph

As has already been alluded to in the previous section, each channel in the multichannel recording of the EEG makes use of a separate differential amplifier. The differential voltage recorded at the scalp is normally less than 200 μV, while common mode voltages may be much higher. The amplifiers must therefore have high differential gain,

very high common mode rejection and low noise characteristics. As the EEG signal components of interest are usually within the 1.5–70 Hz range, a band-pass filter with cut-off frequencies of about 1 and 70 Hz is usual. Older EEG systems produced a paper record of the EEG and so the amplified EEG signals are made to drive galvanometer pens to produce deflections proportional to the input voltages. More recent 'paperless' systems sample the (analog) voltages and convert them to digital equivalents for on-line storage, facilitating computer analysis of the signals and printout in a variety of formats. Different montages can be used by re-routing the connections to the scalp electrodes to give a separate recording montage. On modern digital EEG machines all signals tend to be recorded on *one* montage (e.g., ear-reference) from which others can be generated as required via "reformatting".

2.4 NORMAL EEG

A wide variety of normal EEG patterns can be seen in different persons of the same age, and an even greater variety of normal patterns can occur in different age groups; also, the waking pattern generally shows more variability between subjects than the sleep EEG. All these show that it is not practical to define the normal EEG by listing all the normal patterns and their variations. Neither can the normal EEG be defined by requiring that specific normal components be present. In fact, as there are only a few EEG components known to be definitely abnormal in each age group, the normal EEG can be defined more effectively by the absence of abnormal components than by the presence of normal patterns. Conversely, an EEG is considered abnormal if it contains abnormal components regardless of whether or not it also contains normal components.

Although severe abnormalities of the brain are likely to cause EEG abnormalities, normal EEGs may be seen in some cases of long-standing, mild and small cerebral abnormalities. The converse is also true, i.e., although most abnormal EEG patterns indicate abnormal brain function, a few specific mild EEG abnormal patterns occur occasionally in persons not showing evidence of brain disease.

In order to analyse the EEG, the recording must be broken down into its many components in order to try and make sense of the underlying brain activity. Components include individual waveshapes and how often they are repeated. The distribution of the waveforms (spatially) is also important. The frequency and amplitude are further qualities looked for when analysing the EEG.

2.4.1 Normal rhythms

The on-going EEG activity is referred to as the *background activity.* Although it is essentially a mixture of waves of multiple frequencies, under particular conditions waves

of a certain frequency often dominate. When looking at the 'normal' (or background) EEG, the frequency of EEG waves is often divided into four groups or frequency bands, given by:

1. *delta* frequency band < 4 Hz
2. *theta* frequency band ≥ 4 Hz to < 8 Hz
3. *alpha* frequency band ≥ 8 Hz to < 13 Hz
4. *beta* frequency band ≥ 13 Hz.

Although these divisions are somewhat arbitrary, they help to set apart the most normal and abnormal waves in the EEG. Waves under 8 Hz are commonly called *slow waves* and waves over 13 Hz are called *fast waves.* Table 2.1 provides a concise description of the principal characteristics of each frequency band observed in the EEG.

Delta	< 4Hz	Ubiquitous during deep (Stage IV) sleep.
Theta	≥ 4 Hz to < 8 Hz	Young children and normal adults during light sleep.
Alpha	≥ 8 Hz to < 13 Hz	Dominates occipital scalp.
Beta	≥ 13 Hz	Precentral, frontal. Rare except during sleep.
Mu	≥ 7 Hz to < 11 Hz	Central, left and right. Blocked by contralateral movement.

Table 2.1 A summary of the frequency bands in the *normal* EEG.

2.4.2 Drowsiness, sleep and drugs

Progressing from the alert resting state to the drowsy state, and finally to sleep, also affects the EEG. The 'normal' background EEG is replaced by waveforms which do not correspond to the 'usual' background EEG, but are nonetheless normal when the state of the patient is taken into account.

Drowsiness is the first stage of sleep (stage I sleep) and with it comes a 'slowing down' of the EEG, where the rhythmic background EEG drops to around 5 Hz. As the patient progresses from stage I sleep to stage II sleep, slower waves begin to appear. At this point, 14 to 15 Hz *sleep spindles* appear. As sleep progresses these waveforms increase in amplitude and become more prominent. Deep sleep (stage III sleep) is characterised by large amounts of spindle activity and a very slow but symmetrical record.

Finally, drugs, such as barbiturates and benzodiazepines, as well as other sedatives may produce excessive fast activity initially, which is followed by generalized slowing. The effects of sleep and drugs on the EEG must be taken into account when analysing

the EEG. When under the influence of sleep or drugs the EEG may not necessarily appear 'normal', but when put in context, the EEG is not necessarily abnormal either.

2.5 ARTIFACTS

Artifacts are waveforms recorded in the EEG not caused by cerebral activity. They may be due to other physiological activity originating from the patient, interference from power lines and other electrical sources affecting the recording electrodes, or waveforms generated by the recording system itself, including the recording electrodes themselves. Artifacts are not desirable in the EEG recording because they may mimic or obscure cerebral activity. Great care and attention is taken during the recording of the EEG to minimise artifacts, although it is rather difficult to obtain a completely artifact-free EEG recording.

The following are the most commonly observed artifacts in EEG recordings which can be confused with abnormal cerebral activity or even partially or totally obscure abnormal cerebral activity.

- ***Eye blinks and other eye movements*** — Since the eye is a dipole (the aqueous humor is 100 mV electropositive with respect to the retina), when the patient blinks and Bell's phenomenon occurs (i.e., the eyes roll upward), this increases the positivity of the prefrontal electrodes *Fp1* and *Fp2*. This, in turn, makes *F4* and *F3* relatively more negative and causes a downward deflection in the EEG recording for *Fp1-F3* and *Fp2-F4* in bipolar longitudinal EEG recording (see Figure 2.5a). Lateral eye movements also cause deflections of the recording, when recording with lateral or transverse bipolar electrode chains, mainly in *Fp1-F7* and *Fp2-F8*. Blinking, or eye closure causes a large downwards deflection, and eye opening a large upward deflection. Eye movement artifacts in the EEG can usually be identified by their frontal distribution, their symmetry on the two sides and their characteristic shape.

- ***Muscle activity*** — Muscle artifact is usually easily identified by its shape and repetition. This type of artifact can be generated by movement of the jaw, grinding of teeth, shivering/trembling, and generally any movement involving the scalp and/or facial muscles. Muscle artifact may not only be mistaken as an abnormal EEG pattern but, can frequently obscure abnormal EEG activity in the EEG recording (Figure 2.5b).

- ***Electrocardiogram*** — Voltage changes generated by the heart may be picked up in the EEG, mainly in recordings with wide interelectrode distances, especially in linkages across the head. Small electrocardiogram (ECG) artifacts usually reflect

the R-wave of the ECG, although larger artifacts may reflect additional components of the ECG. These artifacts are usually recognised due to their periodicity and the fact that they usually appear on all channels using a common reference (i.e., referential montages) (Figure 2.5c).

- ***Electrical interference*** — The most common artifact due to electrical interference is the 50/60 Hz noise coming from power lines and equipment. EEG recording equipment generally has filters specifically to filter out mains borne noise, but when contamination occurs it usually affects all recording electrodes.
- ***Electrode 'pop'*** — This artifact is caused by a sudden change of electrode contact causing deflections in the EEG recording which rise or fall abruptly and may mimick fast abnormal EEG activity (Figure 2.5d).

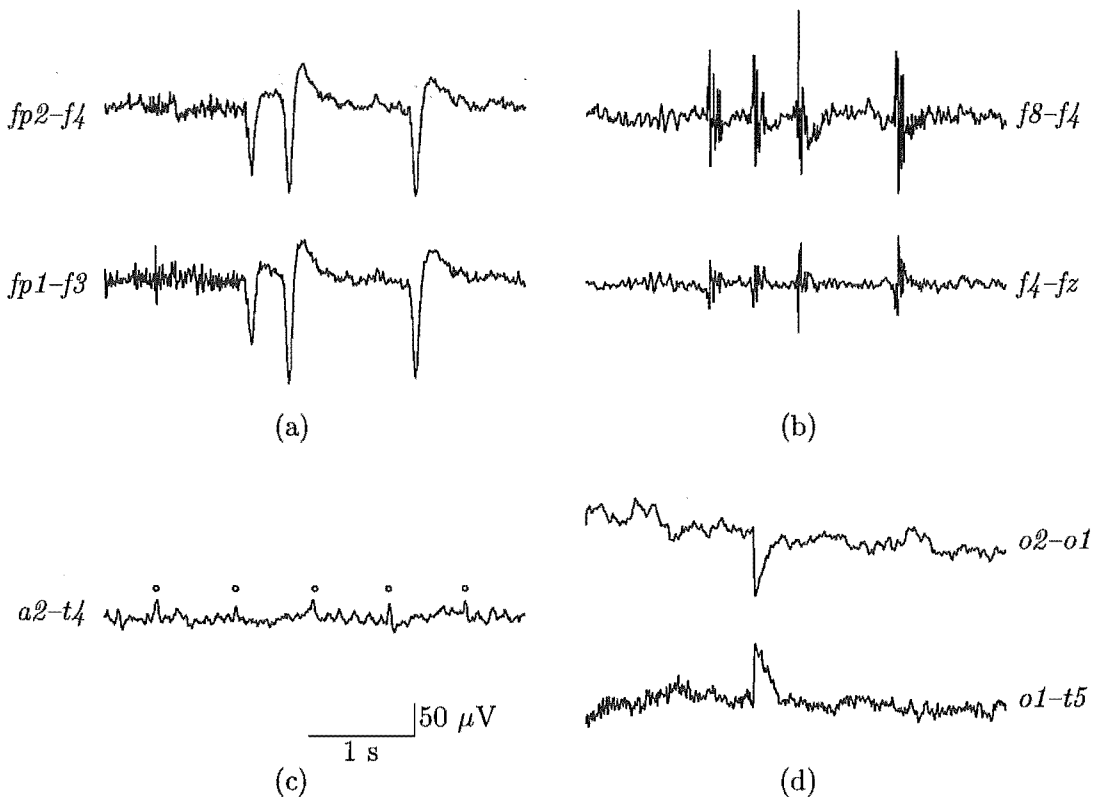


Figure 2.5 Common artifacts that appear in the EEG recording: (a) Eyeblinks, (b) Muscle artifact, (c) ECG and (d) electrode artifact.

2.6 ABNORMAL EEG

An EEG is usually called abnormal not because it completely lacks normal patterns but because it contains: (a) slow waves, (b) abnormalities of amplitude, (c) deviations

from normal patterns and/or (d) epileptiform activity. In most abnormal EEGs, the abnormal patterns do not entirely replace normal activity, but appear, sometimes intermittently, only in some channels, or only superimposed on a normal background. The four categories can be further subdivided thus:

1. Slow waves
   (a) Localized slow waves
   (b) Generalized asynchronous slow waves
   (c) Bilaterally synchronous slow waves
2. Amplitude abnormalities
   (a) Localized amplitude changes: Asymmetries
   (b) Generalized amplitude changes
3. Deviation from normal patterns
4. Epileptiform activity
   (a) Localized epileptiform activity
   (b) Generalized epileptiform activity
   (c) Special epileptiform patterns

Each of the basic abnormal EEG patterns listed above can be caused by one or a few types of cerebral abnormalities. The abnormalities are characterised by their irritative or destructive character and by their cortical, subcortical or epicortical location. There is a correlation between EEG patterns, cerebral pathology and specific diseases. However, the correlation between abnormal EEG patterns and diseases is weakened by several facts: (a) Many diseases cause more than one type of cerebral lesion and, therefore, more than one pattern; (b) Not all cases of a neurological disease cause an observable EEG abnormality – the EEG may appear normal especially if the cerebral lesion is small, chronic or located deeply in the brain. A disease may also produce EEG abnormalities which are intermittent and so rare that they do not appear during the period of a routine EEG recording; (c) The EEG may be abnormal in some people who show no other evidence for a disease.

2.6.1 The EEG and epilepsy

The term *paroxysmal activity* is used in describing EEG records with fast transient abnormalities. Seizure disorders can manifest themselves in an EEG by several types

of paroxysmal activity. Seizures, in neurological terms, are episodes of sudden disturbances of consciousness, mental functions, motor, sensory and autonomic activity, caused by a paroxysmal malfunction of cerebral nerve cells. Epilepsy is the term used to describe the condition of patients who have recurring seizures due to some lasting cerebral abnormality. Epileptic seizures are divided by their clinical manifestations into (a) partial (or focal), (b) generalized, (c) unilateral and (d) unclassified seizures. Partial seizures involve only part of the cerebral hemisphere and produce symptoms in corresponding parts of the body or in some related mental functions. Generalized seizures involve the entire brain and produce bilateral motor symptoms usually with loss of consciousness. Both types of seizures can occur at all ages. Generalized seizures can be subdivided into absence (*petit mal*) seizures and tonic-clonic (*grand mal*) seizures.

Paroxysmal activity of this type usually consists of *transients*. A transient is considered to be a waveform which clearly stands out against the background. A *sharp transient* is a wave of any duration which has a pointed peak, and a sharp transient with a duration of 70 – 200 ms is called a *sharp wave*. A sharp transient with a duration of 20 – 70 ms is called a *spike* [Chatrian *et al.* 1974]. Spikes and sharp waves may be followed by a slow wave and form a *spike-and-wave complex* and a *sharp-and-slow-wave complex* respectively. Figure 2.6 depicts a spike and a sharp wave found in an epileptiform EEG. Spikes and sharp waves are often jointly referred to as spikes, spikes-and-sharp-waves (SSWs), epileptiform transients or epileptiform discharges (EDs).

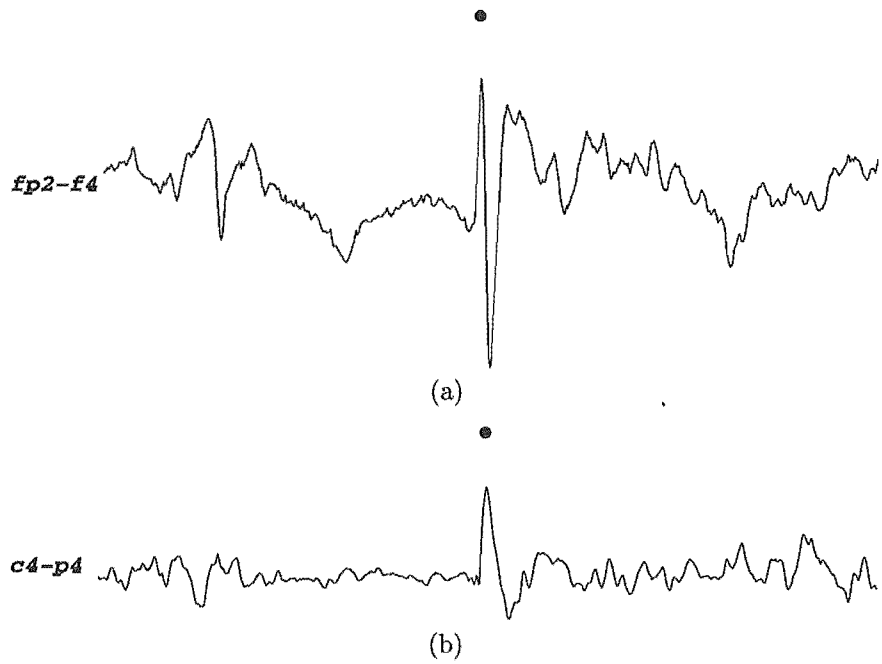


Figure 2.6 Epileptiform transients: (a) Spike and (b) Sharp wave.

Spikes and sharp waves which occur in the EEG and last for more than a few

seconds are called seizure patterns or *ictal* patterns. Although seizure patterns are often associated with clinical seizure manifestations, they may occur without such correlates and are then called *subclinical* seizure patterns. Spikes and sharp waves which last for less than a few seconds are called *interictal epileptiform activity.*

The presence of spikes and sharp waves in the EEG is often indicative of epilepsy and, in *interictal* periods (i.e., periods when there is no clinical manifestation) the detection and proper diagnosis of such abnormal patterns becomes all the more important.

As mentioned above, epileptiform activity can be either localised or generalized. Local epileptiform activity is usually due to a focal irritative lesion of the cerebral cortex. Generalized epileptiform activity is either not associated with demonstrable lesions (i.e., idiopathic epilepsy) or associated with a variety of conditions which increase the excitability of subcortical centres, of wide parts of the cerebral cortex or of both. Both types of epileptiform activity can be identified through the use of various bipolar and referential montages. As explained in Section 2.3.2, epileptiform transients (which are associated with regions of relatively negative potential) recorded using a referential montage will be characterised by upward going (in the EEG, negative is upwards by convention) spikes on a number of channels in the multichannel EEG.

Epileptiform transient refers to the paroxysmal activity seen on one channel in the EEG, however, as epileptiform transients often arise simultaneously on several EEG channels, they are collectively termed an *epileptiform event.*

Local epileptiform activity generally gives rise to *focal* epileptiform events (Figure 2.7a) where the localised centre of negativity is known as the *focus.* Generalized epileptiform activity gives rise to *non-focal* epileptiform events (Figure 2.7b), where the individual epileptiform transients do not change in polarity, amplitude and sharpness and hence do not indicate any focus.

2.7 SUMMARY

The EEG is a recording of the electrical activity in the brain, which is generally recorded at the scalp. It provides information pertinent to the diagnosis of a number of brain disorders, in particular the diagnosis of epilepsy. Whilst the presence of epilepsy can be clinically shown through the many types of clinical manifestations or seizures, the interictal EEG may contain epileptiform transients which are indicative of a diagnosis of epilepsy, without the clinical manifestations.

The EEG, which is recorded in both referential and bipolar montages, is routinely recorded for the electroencephalographer (EEGer) to visually inspect for abnormal EEG patterns. The EEGer looks for focal or non-focal epileptiform events, depending

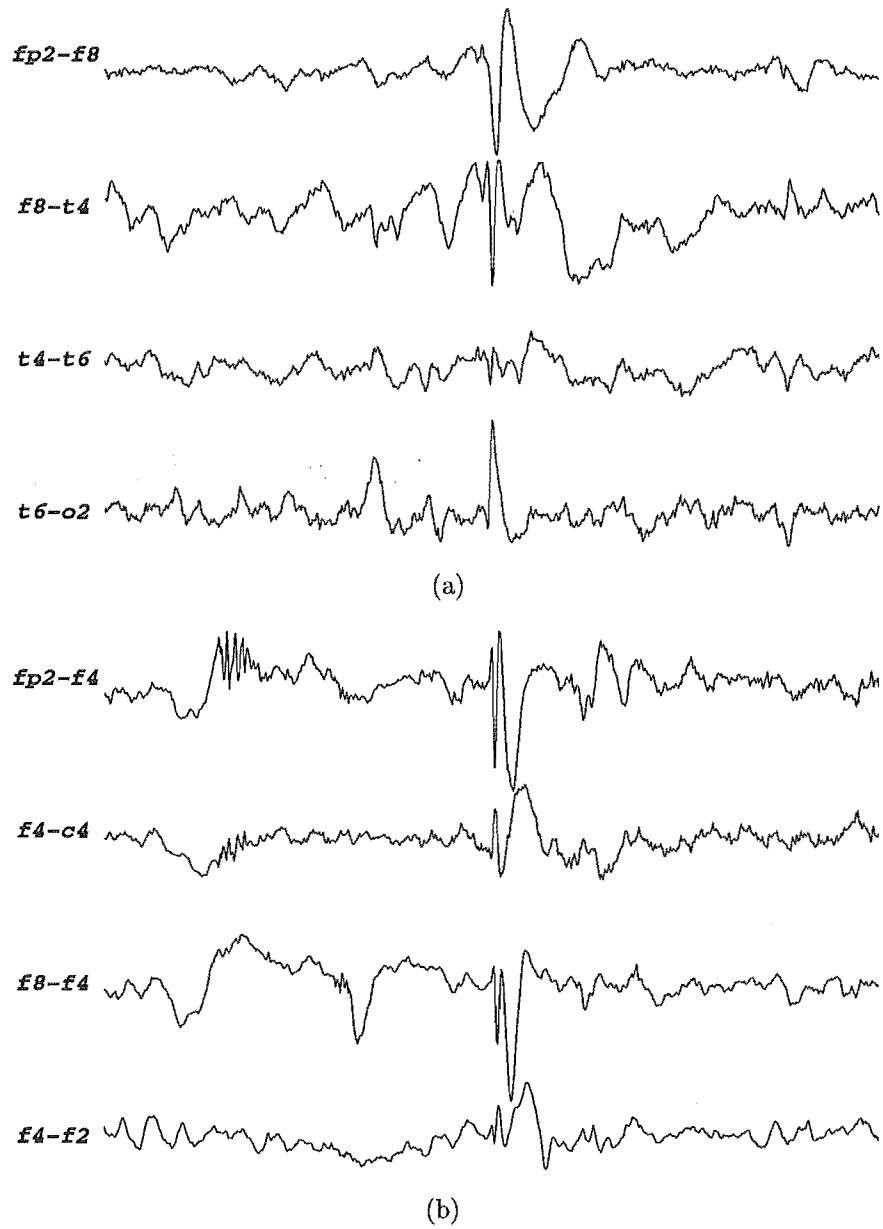


Figure 2.7 (a) A focal event on a bipolar montage showing phase reversal separated by a null channel. (b) A non-focal event.

on the spatial relationships between individual epileptiform transients on each channel which make up the epileptiform event.

The process is made more difficult by the fact that the EEG is invariably contaminated with artifacts which can mimic or obscure abnormal EEG activity, making the detection of epileptiform events in particular, more dependant on the individual EEGer's skill in observing various spatial and temporal clues.

Chapter 3

ARTIFICIAL NEURAL NETWORKS

3.1 INTRODUCTION

As we all go about our daily life, we perform many complex tasks which we all seem to take for granted, tasks such as listening to music, experiencing the awe of a sunset and enjoying the smell of fragrant flowers. We also take for granted tasks such as reading, writing, and walking, not to mention all the mundane tasks such as breathing, and controlling all our other bodily functions. All of these tasks have one thing in common, they all need the use of the complex biological neural network that we call the brain.

This biological neural network consists of an estimated 100 billion neurons [Shepherd and Koch 1990]. Although each neuron is 5 or 6 orders of magnitude slower than silicon logic gates, each has the complexity of a microprocessor. It is the complexity of each neuron coupled with their staggering numbers and their immensely high level of connectivity (it is estimated that there are some 60 trillion connections between neurons [Shepherd and Koch 1990]) which makes the human brain the marvel that it is.

Terms such as *highly complex*, *nonlinear* and *parallel computer* are all terms that are used to describe the brain, and rightly so. For example, the brain can routinely perform perceptual tasks, such as recognising a familiar face against an unfamiliar background, in somewhere around 100–200 ms, whereas much less complex tasks would take days on huge conventional computers.

Although just *how* biological neural networks operate is only just beginning to be understood, it is generally accepted that all biological neural functions, including memory, are stored in the neurons and their interconnections. Learning is viewed as the establishment of new connections between neurons or the modification of existing connections. At birth, our brain has great structure and the ability to build up its own rules through what we call *experience*. In the first two years from birth the most dramatic development of the human brain takes place in which it is estimated that 1 million *synapses* or connections are formed per second [Shepherd and Koch 1990].

Development and learning do not stop there, of course, and in fact continue throughout life.

Although we only have a rudimentary understanding of biological neural networks, it is possible to construct a small network of *artificial neurons* and train them to perform some "simple" task. Such artificial neurons are usually realized as elements in a program or as circuits made of silicon. These *artificial* neural networks are based on extremely simple abstractions of their biological counterpart and do not have a fraction of the power of the human brain. This notwithstanding, they can be trained to perform useful functions.

This chapter first gives a brief historical summary of the development of the field of artificial neural networks followed by a detailed explanation of biological neural networks and their artificial counterparts. It then covers the various ways in which artificial neurons can be interconnected to give various architectures, followed by a look at the learning process in artificial neural networks and the various ways learning can be achieved. Finally the chapter ends with a summary of the more common artificial neural networks and how they are related according to architecture and learning paradigm. Unless otherwise specified, the major sources of information for this chapter are Zurada [1992], Haykin [1994] and Hagan *et al.* [1996].

3.2 A BRIEF HISTORICAL OVERVIEW

The short historical summary which follows is by no means a complete historical account and is intended only to highlight the most important milestones in the colourful history of the development of the artificial neural network as we know it today.

The origin of the neural network field is often attributed to Warren McCulloch and Walter Pitts who in 1943 [McCulloch and Pitts 1943] showed that, in principle, networks of artificial neurons could compute any arithmetic or logical function. However, their model's implementation simply was not technologically feasible through the use of the bulky vacuum tubes of that era.

Hebb [1949] first proposed a learning scheme for updating neuron's connection strengths in 1949. He maintained that information can be stored in the connections of interconnecting neurons and further postulated the learning scheme known as *Hebbian learning* today.

A great deal of interest was generated late in the 1950's when Frank Rosenblatt and his colleagues put together a neural network using their neuron-like element called a *perceptron* [Rosenblatt 1958] and demonstrated their network's ability to perform pattern recognition. Although it was later shown that the perceptron network could only solve a limited class of problems this laid the groundwork for the basic learning algorithms still in use today in training neural networks.

At around the same time, Widrow and Hoff [1960] produced their *ADALINE* or ADAptive LINEar combiner with a new learning rule capable of performing as well as Rosenblatt's perceptron network. Their learning rule is known as the *Widrow-Hoff learning rule* and is still in use today. Early applications of the ADALINE included pattern recognition, adaptive control and weather forecasting.

Books by Nilsson [1965] and Minsky and Papert [1969] both clearly summarised the developments of that time but also formulated the inherent limitations of both the perceptron and ADALINE networks. Both Rosenblatt and Widrow knew of those limitations and proposed network architectures that would overcome them, but could not successfully alter their training algorithms to train the new, modified, networks. These limitations, coupled with the relatively modest computational resources of the time, convinced many researchers that neural networks research was a dead end and resulted in neural networks research being suspended for more than a decade.

A handful of researchers did, however, pursue research in the field of neural networks during the period from 1965 to 1984. Kohonen [1972] and Anderson [1972] continued to work in associative memory research. Stephen Grossberg and Gail Carpenter also introduced a number of neural architectures and theories [Grossberg 1976], [Carpenter and Grossberg 1991].

Neural networks were reborn in the early 1980's following the introduction of two new significant concepts. The first was by Hopfield [1982] who introduced a recurrent neural network architecture for associative memories. The second development was that of the *backpropagation algorithm* introduced to train multilayer perceptrons, a feat which had eluded most researchers for the preceding decade. Foremost in this work were Rumelhart and McClelland [1986].

After the introduction of these developments there was no turning back and the field of artificial neural networks has taken off over the last ten years with more and more applications being found. Applications range from aerospace, banking and insurance to medical, defense and entertainment.

3.3 WHAT IS A NEURAL NETWORK ?

A neuron in the brain can be broken down into three major components: the dendrites, the cell body (or soma) and the axon. Electrical signals are carried into the cell body by the dendritic tree, which forms a very fine bush of thin fibres around the cell body. The cell body effectively sums and thresholds these incoming signals. Impulses are then carried away from the cell body along the axons which are long cylindrical connections. Axons branch out at the tip and terminate in a small end-bulb which almost touch the dendrites of neighbouring neurons. The axon-dendrite contact is called a *synapse*. It is the arrangement of the neurons and the strengths of the individual synapses that

establishes the function of the neural network that is the brain. Figure 3.1 is a simplified schematic diagram showing two biological neurons.

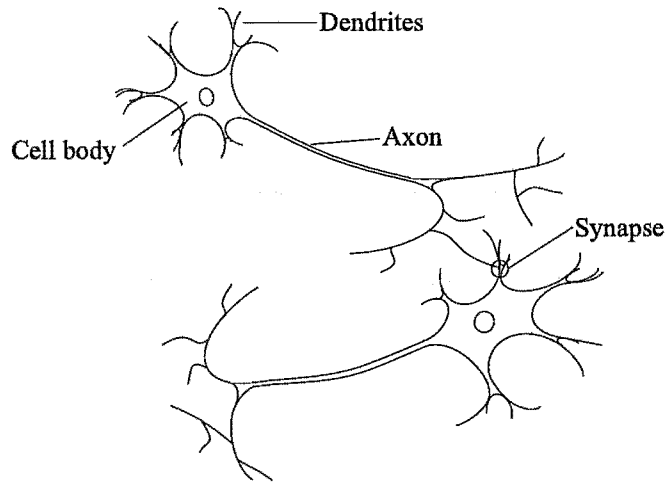


Figure 3.1 A simplified schematic of two biological neurons.

At birth some of the neural structures are already formed but most of the other parts are developed through the learning process of growing up as new connections are made and others waste away. The major developments take place in the early stages of life although the neural structures continue to change throughout life. Later changes are, however, more of strengthening or weakening synaptic functions. The process that developing neurons and interconnections go through can be termed *plasticity.*

Just as plasticity is essential to the functioning of neurons in the human brain, so is it in neural networks made up of *artificial* neurons. In its most general form an artificial neural network is a machine that is designed to roughly *model* the way in which the brain performs a particular task. Artificial neural networks are usually simulated in software on a digital computer or implemented using electronic components. Artificial neural networks are realised through the massive interconnections of simpler neural models ("neurons") which are made to perform useful computations after following a process of *learning.* The following definition of a neural network is given by Aleksander and Morton [1990]:

> "A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:
>
> 1. Knowledge is acquired by the network through a learning process.
> 2. Interneuron connection strengths known as synaptic weights are used to store the knowledge."

The learning process is controlled by a *learning algorithm* whose function it is to alter the synaptic weights in such a way as to make the output of the neural network approach the desired outcome.

3.3.1 A mathematical model of a neuron

The simple model of a neuron shown in Figure 3.2 can be seen to be of the same basic form as its simplified biological counterpart in that it has a number of inputs which are weighted, summed and thresholded at the "cell-body" which in turn produces an output. Three basic elements can be identified in the model:

1. The neuron has a set of *synaptic weights* which are multiplied with each input as they are presented to the neuron. Specifically, an input p_j at the input of synapse j connected to neuron k is multiplied by synaptic weight w_{kj}.

2. A *summing junction* sums the weighted values of the inputs, such that the *net input* for neuron k is given by

$$n_k = \sum_{j=1}^{R} w_{kj} p_j + b_k \tag{3.1}$$

   where $p_1, p_2, \ldots, p_R$ are the input signals, $w_{k1}, w_{k2}, \ldots, w_{kR}$ are the synaptic weights of neuron k, and b_k is an offset term known as the *bias*.

3. The output of the summing junction is passed through an *activation function* which is typically a nonlinear function which tends to limit the amplitude of the output of the neuron, such that

$$a_k = f(n_k) \tag{3.2}$$

   or

$$a_k = f(\sum_{j=1}^{R} w_{kj} p_j + b_k) \tag{3.3}$$

   where a_k is the output of neuron k.

The bias term b_k is sometimes included in the model as an extra synaptic weight with an input of +1, such that

$$a_k = f(\sum_{j=0}^{R} w_{kj} p_j) \tag{3.4}$$

where $w_{k0} = b_k$ and $p_0 = +1$.

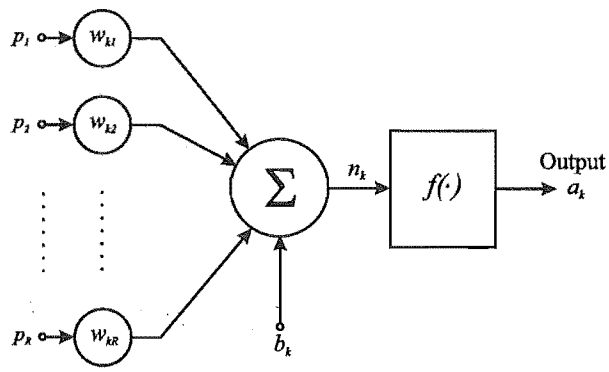


Figure 3.2 A simplified mathematical model of an artificial neuron.

3.3.2 Types of activation functions

The activation function, given by $f(\cdot)$, may be a linear or nonlinear function of n_k the net input to neuron k. An activation function is chosen to satisfy some specification of the problem that the neuron is attempting to solve. Of the many activation functions that can be implemented (see sample in Table 3.1) three of the most commonly used activation functions are described below.

1. The *hard-limiting* activation function (Figure 3.3a) sets the neuron output to 0 if the function argument is less than 0, or 1 if the argument is greater than or equal to 0 i.e.

$$f(n_k) = \begin{cases} 1 & \text{if } n_k \geq 0 \\ 0 & \text{if } n_k < 0. \end{cases} \tag{3.5}$$

Such a neuron is referred to in the literature as the *McCulloch-Pitts model* in recognition of the pioneering work done by McCulloch and Pitts.

2. The *linear* activation function described in Figure 3.3b sets the neuron output equal to the input, i.e.

$$a_k = n_k. \tag{3.6}$$

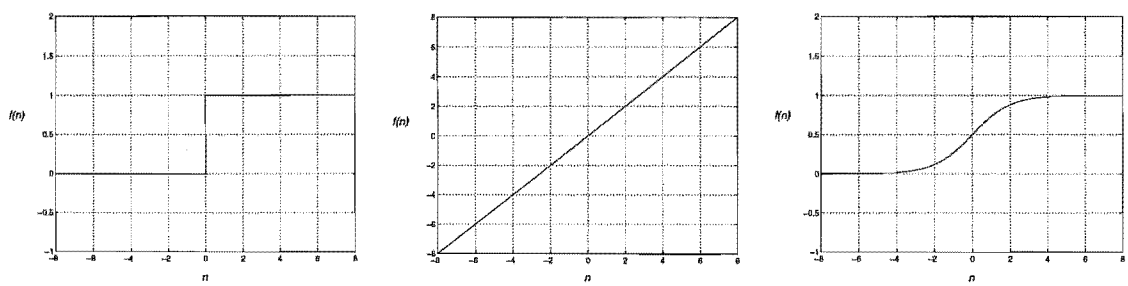
3. The *log-sigmoid* activation function described in Figure 3.3c takes the input (which may have any value which may range between plus and minus infinity) and "squashes" the output into the range 0 to 1 according to the expression

$$f(n_k) = \frac{1}{1 + \exp(-cn_k)} \tag{3.7}$$

where c is the *slope parameter* of the log-sigmoid function. By varying c we can obtain log-sigmoid functions of varying slopes. In the limit, as the slope parameter approaches infinity, the log-sigmoid function becomes simply a hard-limit activation function. The log-sigmoid activation function is by far the most common form of activation function used in artificial neural networks which is used (a) because of its continuous range of output values from 0 to 1 and (b) the fact that it is continuously differentiable (whereas the hard limit function is not).

When it is desirable to have the activation function range from -1 to $+1$, the log-sigmoid function is usually replaced by the *hyperbolic tangent* function defined by

$$f(n_k) = \tanh\left(\frac{n_k}{2}\right) = \frac{1 - \exp(-n_k)}{1 + \exp(-n_k)}. \tag{3.8}$$



(a) (b) (c)

Figure 3.3 The most commonly used activation functions. (a) hard-limiting, (b) linear and (c) log-sigmoid.

Name	Activation function
Hard limit	$f(n) = 0 \quad n < 0$ $f(n) = 1 \quad n \geq 0$
Symmetrical hard limit	$f(n) = -1 \quad n < 0$ $f(n) = +1 \quad n \geq 0$
Linear	$f(n) = n$
Saturating linear	$f(n) = 0 \quad n < 0$ $f(n) = n \quad 0 \leq n \leq 1$ $f(n) = 1 \quad n > 1$
Symmetric saturating linear	$f(n) = -1 \quad n < -1$ $f(n) = n \quad -1 \leq n \leq 1$ $f(n) = +1 \quad n > 1$
Log-sigmoid	$f(n) = \frac{1}{1+\exp^{-n}}$
Hyperbolic tangent	$f(n) = \frac{\exp^{n} - \exp^{-n}}{\exp^{n} + \exp^{-n}}$
Positive linear	$f(n) = 0 \quad n < 0$ $f(n) = n \quad n \geq 0$

Table 3.1 A summary of some of the various activation functions implemented in artificial neural networks.

3.4 NETWORK ARCHITECTURES

The manner in which neurons of a neural network are structured is often closely linked to the learning algorithms being used to train a particular neural network. Neural network architectures (or structures), in general, fall into four different classes:

- Single-layer feedforward networks.
- Multi-layer feedforward networks.
- Recurrent networks.
- Lattice structured networks.

The basic building block of most of the single-layer and multi-layer feedforward networks in the literature is the *perceptron*, developed by Rosenblatt [1958]. Both the perceptron and the networks formed out of it are discussed in more detail in the next chapter.

3.4.1 Single-layer feedforward network

Neurons are generally organised in the form of layers. The simplest is a single-layered network consisting of an output layer of neurons and a number of input nodes. Note that some sources in the literature count the 'layer' of input nodes as part of the layers making up an ANN (in this way the above mentioned network would become a 2-layered

ANN). In this text only layers consisting entirely of *neurons* are counted when describing an ANN. This is a *feedforward* type neural network because the input progresses forward through the layer to the output without feedback. Figure 3.4 depicts the case for R inputs and S output nodes (neurons). For the neural network of Figure 3.4 the output of neuron k is given by

$$a_k = f(\sum_{j=1}^{R} w_{kj}p_j + b_k) \qquad \text{for } k = 1, 2, \ldots, S. \tag{3.9}$$

In matrix notation this can be written as

$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b}) \tag{3.10}$$

where the input vector is defined as $\mathbf{p} = [p_1 p_2 \cdots p_R]^T$ and the output vector as $\mathbf{a} = [a_1 a_2 \cdots a_S]^T$. The synaptic weight matrix connecting each input node to each neuron in the output layer is given by the weight matrix

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1R} \\ w_{21} & w_{22} & \cdots & w_{2R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S1} & w_{S2} & \cdots & w_{SR} \end{bmatrix} \tag{3.11}$$

and the bias terms are given by

$$\mathbf{b} = [b_1 b_2 \cdots b_S]^T. \tag{3.12}$$

It can be seen that the row indices of the elements of matrix $\mathbf{W}$ indicate the destination neuron associated with that weight, while the column indices indicate the source of the input for that weight.

The matrix operator $\mathbf{f}$ of activation functions f is given by

$$\mathbf{f}[\cdot] = \begin{bmatrix} f(\cdot) & 0 & \cdots & 0 \\ 0 & f(\cdot) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\cdot) \end{bmatrix} \tag{3.13}$$

where each activation function $f(\cdot)$ on the diagonal of the matrix operates component-wise on the net inputs of each neuron (n_k).

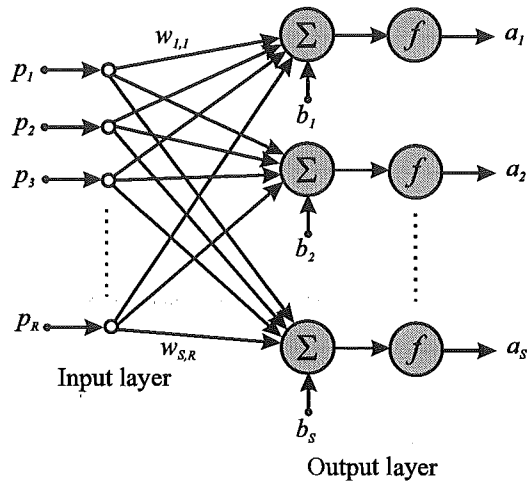


Figure 3.4 A single-layer feedforward neural network. (This can also be termed a two layer ANN if the input nodes are counted as a layer).

3.4.2 Multi-layer feedforward network

The more commonly used feedforward neural network structure is that of the multi-layer feedforward neural network. This arrangement has a number of *hidden layers* composed of *hidden neurons* or *hidden nodes.* One or more hidden layers can be added between the external input and the network output and each hidden layer can have different numbers of neurons. The presence of the hidden layer (or layers) enables the network to extract higher-order statistics.

In the multilayer feedforward network, the outputs of each layer are the inputs to the next.

- **Fully connected**

  Figure 3.5 depicts a three layer feedforward network with R input nodes, $S1$ neurons in the 1^{st} layer (1^{st} hidden layer), $S2$ neurons in the 2^{nd} layer (2^{nd} hidden layer) and $S3$ neurons in the output layer. For brevity the network in Figure 3.5 is referred to as a R-$S1$-$S2$-$S3$ network. The network depicted is said to be *fully connected* in the sense that every node in each layer of the network is connected to every node in the next.

  The output of neuron l in the output layer of the neural network depicted in Figure 3.5 is given by

$$a_l^3 = f^3\left(\sum_{k=1}^{S2} w_{lk}^3 f^2\left(\sum_{j=1}^{S1} w_{kj}^2 f^1\left(\sum_{i=1}^{R} w_{ji}^1 p_i + b_j^1\right) + b_k^2\right) + b_l^3\right) \text{ for } l = 1, 2, \ldots, S3 \tag{3.14}$$

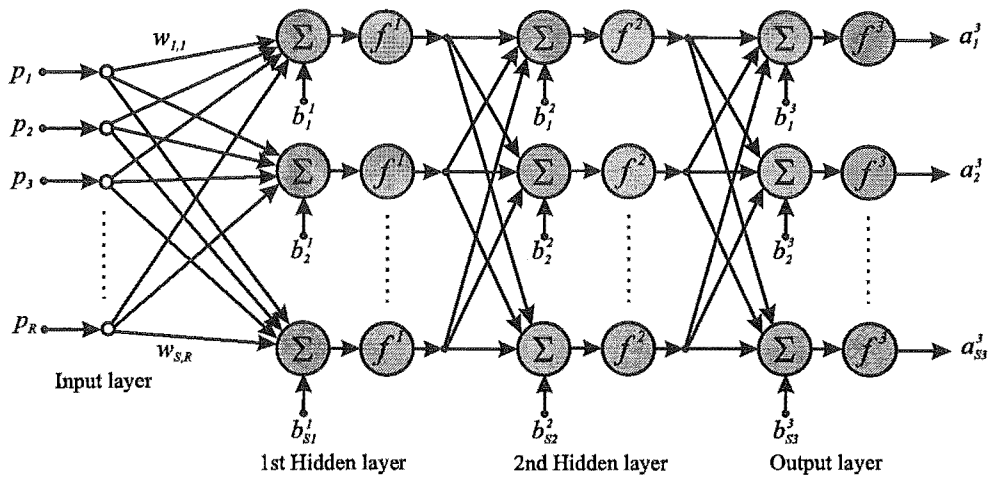


Figure 3.5 A fully connected (R-$S1$-$S2$-$S3$) feedforward neural network.

where superscripts have been used to identify the layers to which each weight matrix and bias vector belongs, such that the weight matrix for the 1st hidden layer is given by $\mathbf{W}^1$ and for the 2nd hidden layer is $\mathbf{W}^2$ and so on. In matrix notation the output of the neural network is given by

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3). \tag{3.15}$$

Note that it is not necessary to keep the same activation function through the entire neural network. The activation functions can be different for different layers as denoted by f^1, f^2 and f^3.

- **Partially connected**

  If some synaptic links are missing, then the network is said to be *partially connected*, as shown in Figure 3.6. The network depicted is said to be locally connected where each neuron in the hidden layer is connected to a local set of source nodes that lies in the immediate neighbourhood. The neurons in the output layer are likewise connected to a local set of hidden neurons. Such a specialised structure in a neural network normally arises when some *a priori* information is known about the nature of the input signal to the network.

3.4.3 Recurrent networks

A *recurrent neural network* is different to a feedforward neural network in that it possesses at least one *feedback* loop. A simple example of a recurrent neural network is given by Figure 3.7. The figure depicts a network with both hidden neurons and

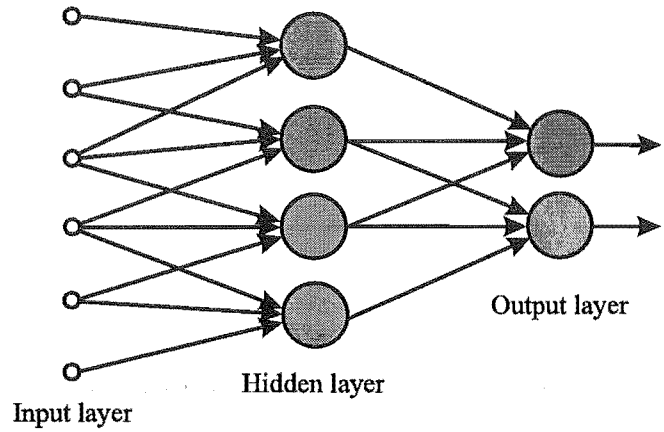


Figure 3.6 A partially connected feedforward neural network.

output neurons, both of which have feedback connections to some of the input nodes. The type of feedback depicted here is termed self-feedback as the output of a neuron is being fed back into its own input. The feedback loops involve the use of the *unit-delay* elements given by z^{-1}.

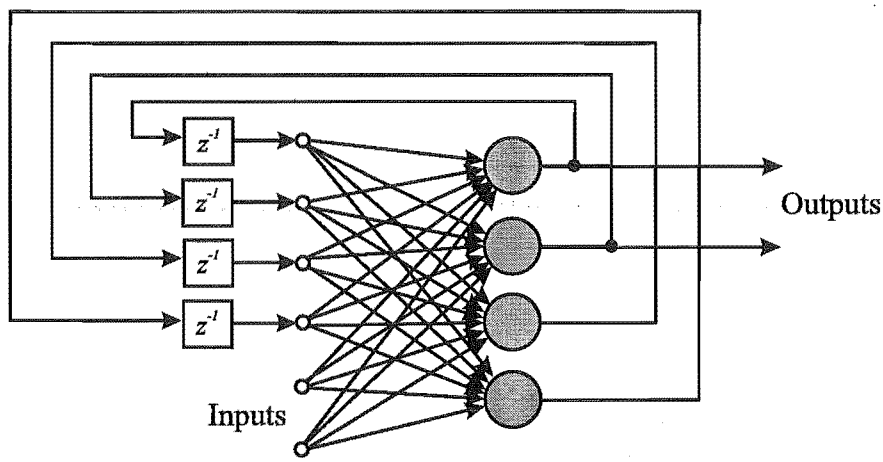


Figure 3.7 A recurrent neural network.

3.4.4 Lattice structured networks

Lattice structured networks consist of neurons arranged in one-, two- or higher-dimensional arrays. In each case the input nodes supply the input to every neuron in the array. The dimension of the array refers to the dimensions of space in which the arrangement of neurons exists. Figure 3.8a depicts a one-dimensional lattice of neurons whereas Figure 3.8b depicts a two-dimensional lattice. In each case the input vector is presented to each neuron in the lattice.

A lattice network is really a single-layer feedforward network. The only difference is that the spatial location of each neuron in the single-layer network now has a significance. The spatial organisation of neurons and their response is an important feature of certain ANNs; an example is the self-organising feature map which is discussed in more detail in Chapters 5 and 6.

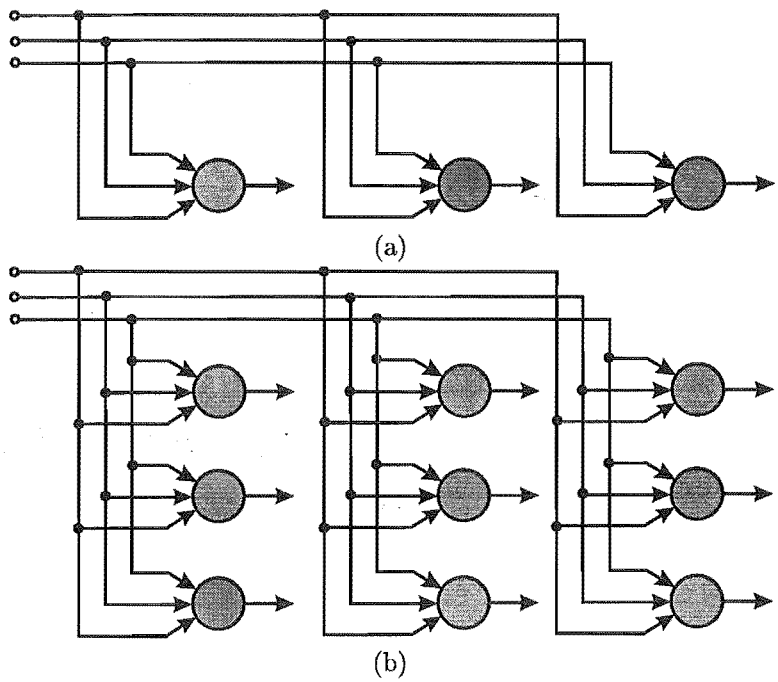


Figure 3.8 Lattice structured networks. (a) A one-dimensional lattice and (b) a two-dimensional lattice.

3.5 THE LEARNING PROCESS

Fundamental to a neural network is the ability to learn from its environment and to improve performance through the process of learning. Learning in neural networks takes place iteratively through adjustments to the synaptic weights and biases and ideally the network becomes more knowledgeable about its environment after each iteration of the learning process. The learning process can be broken down into a sequence of three steps:

1. *Stimulation.* The network is stimulated by inputs from its operating environment.

2. *Response.* The network alters its response to inputs from the operating environment due to the changes in its synaptic weights and biases.

3. *Change.* The network undergoes change as a result of its stimulation.

If we let $w_{kj}(n)$ denote the value of synaptic weight w_{kj} at time n, applying an *adjustment* $\Delta w_{kj}(n)$ at time n to synaptic weight w_{kj} yields an *updated* value for the synaptic weight given by $w_{kj}(n+1)$ such that

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n). \tag{3.16}$$

A *learning algorithm* is a prescribed set of well-defined rules for the solution of a learning problem. There is no unique learning rule which encompasses the training of all neural networks; there are rather a diverse number of learning algorithms all of which have their own advantages and disadvantages, and which differ in the way in which the adjustment Δw_{kj} is formulated. The description of learning algorithms, is split in the remainder of this section into *learning paradigms* and *learning rules.*

3.5.1 Learning paradigms

3.5.1.1 Supervised learning

Consider an environment which is unknown to a neural network but which is shared by a *teacher* who has knowledge of the environment represented by a set of input–output examples. If both the neural network and the teacher are exposed to a training vector (example) drawn from the environment, the teacher, by virtue of knowledge of the environment, is able to provide a *desired response* to the neural network for that particular input vector. It is then the responsibility of the learning algorithm to adjust the network parameters using the information both in the input vector and in the difference between the desired response and the actual response of the network, the *error signal.* By carrying out these adjustments in an iterative fashion the neural network comes to emulate the teacher (or expert); this constitutes the *supervised learning paradigm* (see Figure 3.9). In other words, the neural network comes to have as much knowledge of the environment as the teacher (or at least knowledge which is optimal in some statistical sense). Once this condition is reached, the network is sufficiently trained and so supervised learning is stopped and the neural network is left to respond to the environment on its own (i.e., unsupervised) without any further synaptic weight adjustments.

Supervised learning can take place off-line or on-line. For off-line learning, the neural network is trained until the desired performance is accomplished and then learning is stopped and the network used without any further changes to the synaptic weights. In on-line learning, learning is accomplished in *real time*, meaning that the network parameters are constantly changing.

A disadvantage of supervised learning is that, without a teacher or expert, no new

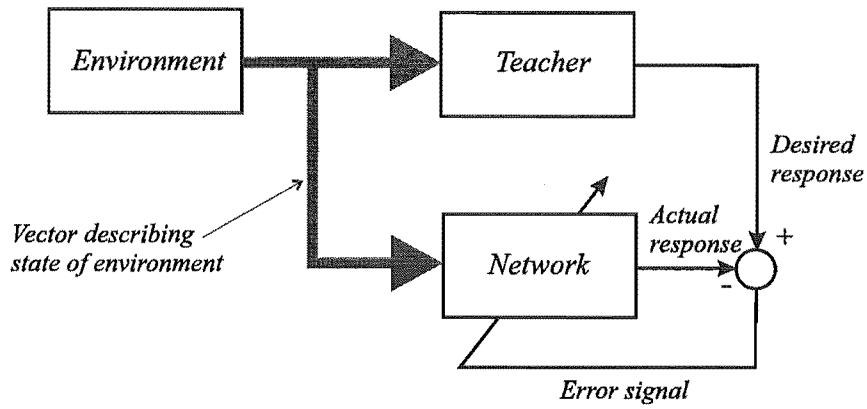


Figure 3.9 Block diagram of supervised learning.

strategies can be learned if they are not covered by the set of examples used to train the network in the first place.

3.5.1.2 Reinforcement learning

Reinforcement learning is the on-line learning of an input-output mapping through a process of trial-and-error designed to maximize a scalar performance index called a *reinforcement signal.* Early studies in artificial intelligence by Minsky [1961] seem to have coined the term "reinforcement learning", although the basic idea of reinforcement has its origins in experimental studies of animal learning in psychology. The following is Thorndike's classical *law of effect* [Thorndike 1911] which is particularly illuminating in this context:

> "Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond."

The paradigm of reinforcement learning can be of a nonassociative type, in which the reinforcement signal is the *only* input that the learning system receives from its environment, or an associative type where the environment provides additional forms of information other than the reinforcement signal.

3.5.1.3 Unsupervised (self-organised) learning

In unsupervised, or self-organised, learning there is no external teacher with knowledge of the environment to oversee the learning process, as can be seen in Figure 3.10. This means that there are no specific examples of the input-output relationships to be drawn from the environment. In this case the parameters of the network are altered in order to optimise a task-independent measure, this being a measure of the quality of representation that the network is required to learn. Once the network has become tuned to the statistical regularities of the input data, it develops the ability to form internal representations for encoding features of the input. By rights, calling unsupervised learning 'learning without a teacher' is not the most appropriate terminology, because learning without a teacher is not possible at all. Although the teacher does not have to be involved in every training step, they have to set goals – even in an unsupervised learning mode.

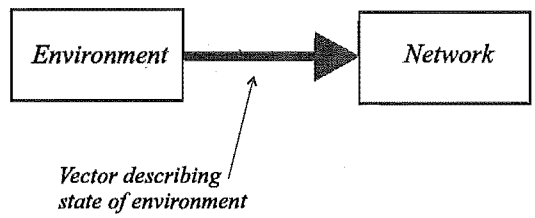


Figure 3.10 Block diagram of unsupervised learning.

3.5.2 Learning rules

3.5.2.1 Hebbian learning

The oldest and most famous of all learning rules is *Hebb's postulate of learning* named after neuropsychologist Hebb [Hebb 1949]. The rule implements the interpretation of the classic statement:

> "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased."

The above statement is made in a neurobiological context. It essentially says that if two neurons on either side of a synapse are activated near simultaneously then the strength of that synapse is selectively increased, whereas for a synapse where the above does not happen, the synapse is selectively weakened. Such a synapse can be called a *Hebbian synapse.*

In mathematical terms, Hebb's postulate of learning can be formulated as follows

$$\Delta w_{kj}(n) = F(a_k(n), p_j(n)) \tag{3.17}$$

where $w_{kj}(n)$ represents the synaptic weight connecting neuron j to neuron k at time n, $\Delta w_{kj}(n)$ the adjustment applied to the synaptic weight at time n, $a_k(n)$ and $p_j(n)$ the output of, and input to, neuron k, respectively, at time n and $F(\cdot, \cdot)$ is a function of both the presynaptic and postsynaptic activities.

One mathematical interpretation of the postulate (i.e., equation 3.17) is

$$\Delta w_{kj}(n) = \alpha a_k(n) p_j(n), \tag{3.18}$$

where α is a positive constant that determines the rate of learning. Note that the expression given by Equation 3.18 actually extends Hebb's postulate beyond its strict interpretation. As the change in the weight is proportional to a product of the activity on either side of the synapse, not only is the weight increased when both a_k and p_j are positive but it is increased when both are negative too.

Since its inception, the Hebbian rule has evolved in a number of directions. In some cases the Hebbian rule needs to be modified to counteract unconstrained growth of synaptic weight values, which takes place when presynaptic and postsynaptic activities consistently agree in sign. Note that the Hebbian learning rule represents purely feedforward, unsupervised learning. However, by replacing the actual output a_k in Equation 3.18 above with the *desired* output d_k the Hebbian rule represents supervised learning.

3.5.2.2 Error-correction learning

When an input $\mathbf{p}(n)$ is applied at time n to a network in which neuron k is embedded, a response $a_k(n)$ is produced. If $d_k(n)$ denotes some *desired response* for neuron k at time n, then both $\mathbf{p}(n)$ and $d_k(n)$ constitute a particular example presented to the network at time n. Typically, the actual response $a_k(n)$ of neuron k is different from the desired response $d_k(n)$. Hence, we may define an *error signal* as the difference between the desired response $d_k(n)$ and the actual response $a_k(n)$, as shown by

$$e_k(n) = d_k(n) - a_k(n). \tag{3.19}$$

The ultimate purpose of error-correction learning is to minimize a cost function based on the error signal $e_k(n)$, such that the actual response of each neuron approaches

the desired response for that neuron is some statistical sense. Once a cost function is selected, error-correction learning becomes strictly an optimization problem. A criterion commonly used for the cost function is the mean-squared-error criterion (MSE criterion), defined as the mean squared value of the sum of squared errors

$$J = E\left[\frac{1}{2}\sum_{k} e_k^2(n)\right] \tag{3.20}$$

where E is the statistical expectation operator and the summation takes place over all neurons in the output layer of the network. Equation 3.20 assumes that the underlying processes are wide-sense stationary.

Attempting to minimize the cost function J with respect to the network parameters leads to the *method of gradient descent* (Haykin [1991], Widrow and Stearns [1985]). However, since this optimization procedure requires knowledge of the statistical characteristics of the underlying processes, which are generally not known, an approximation is made to the optimization problem by using the instantaneous value of the sum of squared errors as the criterion of interest,

$$\varepsilon(n) = \frac{1}{2}\sum_{k} e_k^2(n). \tag{3.21}$$

The network is then optimized by minimizing $\varepsilon(n)$ with respect to the network synaptic weights. This forms the basis of the error-correction learning rule, or the delta rule as it is also called, which states that the adjustment $\Delta w_{kj}(n)$ made to synaptic weight w_{kj} at time n is given by (Widrow and Hoff [1960])

$$\Delta w_{kj}(n) = \alpha e_k(n) p_j(n), \tag{3.22}$$

where α is a positive constant that determines the rate of learning. Note that the delta rule is covered in more detail in Chapter 4.

To ensure stability of the learning process, care has to be exercised in the choice of the value assigned to the learning-rate parameter α. If α is small, the learning proceeds steadily but it may take a long time for the system to converge to a stable solution. If α is large, then the rate of learning is accelerated but the danger that the learning process may diverge and of the system becoming unstable is increased.

The error-performance surface, or the error surface, is a plot of the cost function versus the synaptic weights characterising the neural network. The error surface can be bowl-shaped with a unique minimum point (as is the case with a neural network consisting entirely of neurons with linear activation functions) as can be seen in Fig-

ure 3.11a for a two-dimensional (two-weight) case, or else the error surface can have a global minimum as well as local minima (as is the case when the neural network consists of neurons with nonlinear activation functions) as in Figure 3.11b. For both cases, error-correction learning starts from an arbitrary point on the error surface (determined by the initial values of the synaptic weights) and then proceeds to move in the direction of a minimum in an iterative fashion.

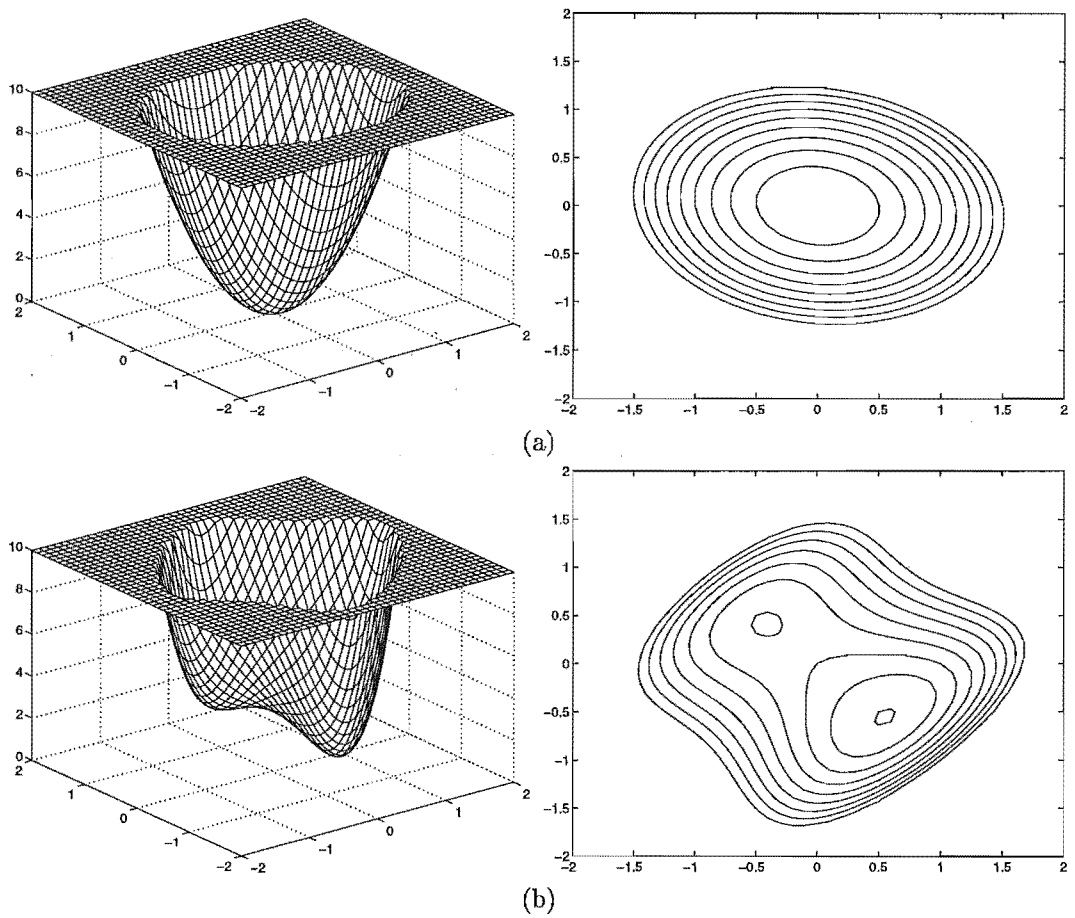


Figure 3.11 Performance error surfaces (two synaptic weights only) depicting (a) a unique global minimum and (b) both local and global minima.

Many methods of minimizing the cost function exist such as the method of steepest descent, Newton's method and the conjugate gradient method. Error-correction learning also takes many forms such as the delta rule (mentioned above), the Widrow-Hoff learning rule (also known as the least-mean-square rule, and can be considered as a special case of the delta rule) and the perceptron learning rule [Haykin 1994]. Error-correction learning along with methods of cost function minimization are discussed in more detail in Chapter 4.

3.5.2.3 Competitive learning

As the name implies, in competitive learning the output neurons of a network compete amongst themselves to be the active neuron. So, whereas in a neural network based on Hebbian learning several output neurons may be active simultaneously, in the case of competitive learning only one output neuron may be active at any one time. There is substantial evidence for competitive learning playing an important role in the formation of topographic maps in the brain [Durbin *et al.* 1989] and recent experimental work by Ambros Ingerso *et al.* [1990] provides further neurobiological justification for competitive learning.

There are three basic elements to a competitive learning rule, namely:

1. A neural network based on competitive learning contains a set of neurons that are all the same except for some randomly distributed synaptic weights which respond differently to a given set of input patterns.
2. A limit is imposed on the "strength" of each neuron.
3. A mechanism that allows neurons to compete for the right to respond to a given subset of inputs, such that only one output neuron is active at a time (*winner-takes-all*).

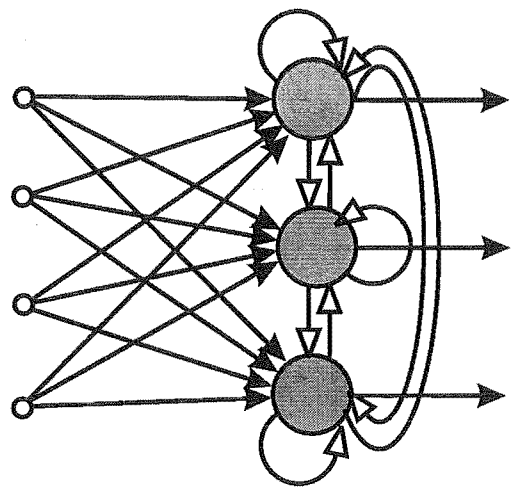


Figure 3.12 Inhibitory lateral connections in a neural network.

In the simplest form of competitive learning, the neural network consists of a single layer of neurons each of which is fully connected to the input nodes. There may also be lateral connections amongst the neurons as shown in Figure 3.12. These lateral connections tend to be of an inhibitory nature, with each neuron tending to inhibit the neuron to which it is laterally connected. The lateral connections thus form lateral inhibition whilst the feedforward synaptic connections are excitatory.

For neuron k, say, to be the winner, its net internal activity n_k for a specified input vector (or pattern) $\mathbf{p}$, must be the largest amongst all the output neurons in the network. The output signal a_k of the winning neuron is set to 1 while the output signals of all others (the losers) are set to zero.

The change Δw_{kj} applied to synaptic weight w_{kj} according to the standard competitive learning rule is defined by

$$\Delta w_{kj} = \begin{cases} \alpha(p_j - w_{kj}) & \text{if neuron } k \text{ wins the competition} \\ 0 & \text{if neuron } k \text{ loses the competition} \end{cases} \tag{3.23}$$

where α is the learning rate parameter. This rule has the overall effect of moving the synaptic weight vector $\mathbf{w}_k$ of winning neuron k towards the input pattern $\mathbf{p}$.

The competitive learning rule is also known as the winner-take-all learning rule which is essentially an unsupervised learning rule.

Table 3.2 provides a summary of learning rules and their properties including the particular learning paradigm as well as the requisite activation function for the neurons in each particular network.

Learning rule	**Weight adjustment**	**Paradigm**	**Act. fcn.**
Hebbian learning → Hebbian rule	$\Delta w_{kj} = \alpha a_k p_j$	Unsupervised	Any
Error correction learning → delta rule → LMS rule → Perceptron rule	 $\Delta w_{kj} = \alpha(d_k - a_k)p_j$ $\Delta w_{kj} = 2\alpha(d_k - (\mathbf{w}_k\mathbf{p}))p_j$ $\Delta w_{kj} = \alpha(d_k - \text{sgn}(\mathbf{w}_k\mathbf{p}))p_j$	 Supervised Supervised Supervised	 Continuous Continuous Hardlimiting
Competitive learning → winner-takes-all rule	$\Delta w_{kj} = \begin{cases} \alpha(p_j - w_{kj}) & k \text{ winner} \\ 0 & \text{otherwise} \end{cases}$	Unsupervised	Continuous

Table 3.2 A summary of the learning rules and their properties.

3.6 SUMMARY

This chapter consists mainly of an introduction to the concept of artificial neural networks, their various possible architectures and the means of learning or acquiring knowledge. In order to conclude the chapter a chart is shown in Figure 3.13 indicating the classification of the most common neural network architectures. The chart is by no means exhaustive and is intended simply to show the great diversity of the neural architectures and their associated learning algorithms.

There can be numerous ways of grouping neural network architectures together. One meaningful basis for classification is to differentiate neural networks by their learn-

ing paradigm, the main forms of learning being supervised, unsupervised and reinforcement learning. Another way of grouping can be to group networks in terms of their input and output values as discrete (binary) or continuous.

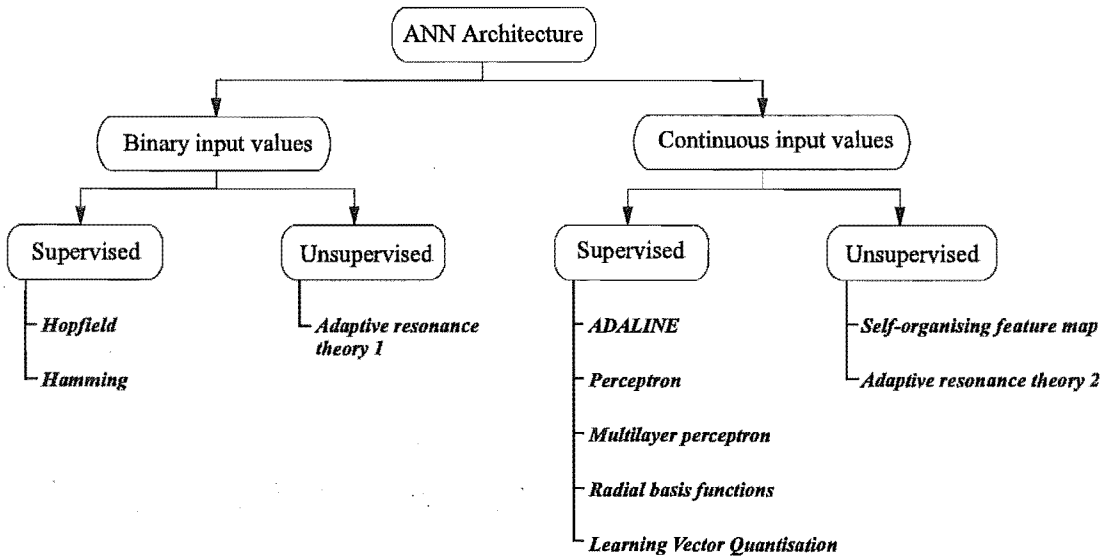


Figure 3.13 A summary of artificial neural networks.

Although the many neural network architectures that are around provide different tools in the solution of different problems, neural networks are not able to perform all tasks. For example, they are particularly poor in formal logic and arithmetic and they are not good in storing and retrieving large amounts of information with a high degree of accuracy [Haykin 1994], [Hagan *et al.* 1996].

While neural networks can in some cases effectively provide a 'black box' solution to a problem, it is difficult to determine what a network has learnt because the knowledge of the network is embodied in a large number of synaptic weights and biases. As a result the theoretical and practical limitations of neural networks are not that well understood at this time. For example, it is unclear as to what architecture is best suited for a given problem. In the case of the multilayer perceptron, it is known that a two-layer network (input nodes, 1 hidden layer and an output layer) can separate arbitrary classes of input patterns, however it is not known how many neurons should be used in each layer, and what activation functions should be used to guarantee good performance of the network [Hagan *et al.* 1996].

Chapter 4

PERCEPTRON NETWORKS

4.1 INTRODUCTION

Whereas in the previous chapter artificial neural networks were introduced as a concept along with methods of training neural networks in general, attention in this chapter turns to some specific architectures and the particular learning algorithms used to train each. The chapter introduces neural network architectures evolving from Rosenblatt's single-layer perceptron [Rosenblatt 1958] and the learning algorithms associated with each network architecture. The similarities between the single-layer perceptron and the adaptive linear combiner are shown along with the major limitations of each. Finally the multi-layer perceptron is introduced along with the now ubiquitous backpropagation algorithm which can be used to train it. Methods of making the algorithm faster and making sure of convergence are also discussed.

4.2 THE SINGLE-LAYER PERCEPTRON

Frank Rosenblatt and several other researchers developed the single-layer perceptron (SLP – then known simply as "perceptron") in the late 1950s. Rosenblatt's key contribution was the introduction of a learning rule for training perceptron networks to solve pattern recognition problems [Rosenblatt 1958]. The SLP is inherently limited, however, as was widely publicized by Minsky and Papert [1969]. The limitations of the SLP and its learning rule were not overcome until the 1980s with the introduction of a learning rule capable of training multi-layer perceptron networks. Nonetheless, the SLP is still considered a fast and reliable network for a specific class of problems.

4.2.1 SLP architecture

The SLP is shown in Figure 4.1, with an arbitrary number of neurons S in the output layer, each with a hardlimiting activation function. The output of the network is given

by

$$a_k = f(\sum_{j=1}^{R} w_{kj}p_j + b_k), \quad k = 1, 2, \ldots, S, \tag{4.1}$$

where $f(\cdot)$ represents the hardlimiting activation function of Equation 3.5 and $\mathbf{p}$ the input vector of order R presented to the network.

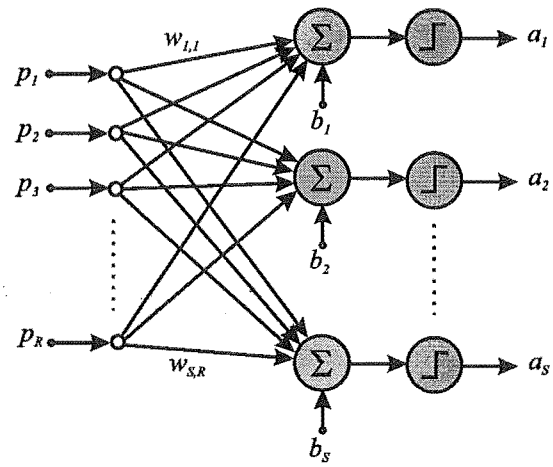


Figure 4.1 Single-layer perceptron (SLP) architecture.

Let us consider first the case of a two-input SLP with a single output neuron ($R = 2$, $S = 1$). The output of this network is determined by

$$a = f(\sum_{j=1}^{2} w_j p_j + b) \tag{4.2}$$

$$= f(w_1 p_1 + w_2 p_2 + b). \tag{4.3}$$

The decision boundary produced by the neuron is determined by the input vectors for which the net input n is zero, that is,

$$n = w_1 p_1 + w_2 p_2 + b = 0. \tag{4.4}$$

Figure 4.2 depicts the two input, single neuron SLP and its corresponding decision boundary when

$$w_1 = 1, w_2 = 1, b = -1. \tag{4.5}$$

The shaded region of Figure 4.2 represents an output of 1 for any input vector within the shaded region and an output of zero otherwise.

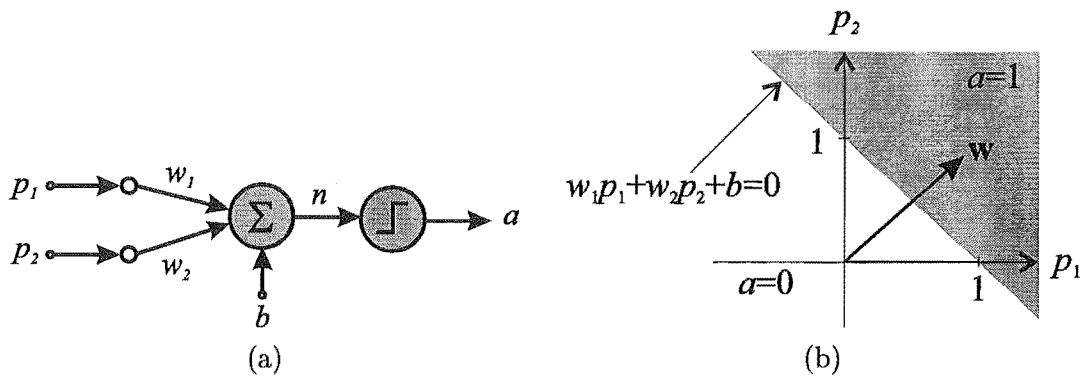


Figure 4.2 A two input single neuron perceptron and the decision boundary corresponding to $w1 = 1$, $w2 = 1$ and $b = -1$.

For multiple neuron SLPs, there will be one decision boundary for each neuron. The decision boundary for neuron k will be defined by

$$\sum_{j=1}^{R} w_{kj}p_j + b_k = 0. \tag{4.6}$$

For S neurons in the network, input vectors can be classified into a total of 2^S possible categories.

4.2.2 Perceptron learning rule

The perceptron learning rule is a supervised learning rule and so requires a training set of input vectors along with a corresponding set of desired outcomes, such as

$$\{\mathbf{p}_1, \mathbf{d}_1\}, \{\mathbf{p}_2, \mathbf{d}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{d}_Q\}, \tag{4.7}$$

where $\mathbf{p}_q$ is an input vector to the network and $\mathbf{d}_q$ is the corresponding desired output in a training set of Q vector pairs.

The perceptron learning rule is a form of error-correction learning, so for a multiple neuron perceptron and with an error vector defined by

$$\mathbf{e} = \mathbf{d} - \mathbf{a} \tag{4.8}$$

the perceptron rule is given by

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \mathbf{e}\mathbf{p}^T \tag{4.9}$$

and

$$\mathbf{b}(k+1) = \mathbf{b}(k) + \mathbf{e}, \tag{4.10}$$

where $\mathbf{W}(k+1)$ is the updated weight matrix and $\mathbf{b}(k+1)$ the updated bias vector for discrete time step $k+1$.

The activation function used in the study here has been the hardlimiting activation function as used by the McCulloch-Pitts model of a neuron [McCulloch and Pitts 1943]. It is possible to use other 'soft-limiting' activation functions, such as the log-sigmoid activation function, and it is tempting to think that that the network might do better in such a case. It turns out, however, that regardless of the type of nonlinearity used, an SLP can perform pattern classification only on linearly separable patterns [Lippmann 1987].

4.3 THE ADAPTIVE LINEAR COMBINER

In 1960 Bernard Widrow and graduate student Marcian Hoff, inspired by the SLP network, introduced the ADALINE (ADAptive LInear NEuron) network, and a learning rule which they called the LMS (Least Mean Square) algorithm [Widrow and Hoff 1960]. The LMS algorithm is also known as the Widrow-Hoff learning algorithm or as a special case of the Delta rule.

The ADALINE is very similar to the SLP, except that its activation function is linear instead of the hard-limiter of the SLP. They both also suffer from the same inherent limitation, in that they both can only solve linearly separable problems. The LMS algorithm is, however, more powerful than the perceptron learning rule. While both algorithms are guaranteed to converge to a solution that correctly categorizes the training patterns (assuming linear separability), the resulting network for the SLP can be sensitive to noise as the patterns often lie close to the decision boundaries. The LMS algorithm is less susceptible to noise as it tries to move the decision boundaries as far from the training patterns as possible by minimizing the mean square error.

4.3.1 The ADALINE architecture

The ADALINE network shown in Figure 4.3 has the same basic structure as the perceptron network, the only difference being the linear activation function.

The output of the network is given by

$$a_k = f(\sum_{j=1}^{R} w_{kj}p_j + b_k) = \sum_{j=1}^{R} w_{kj}p_j + b_k \tag{4.11}$$

where f denotes the linear activation function $f(n) = 1$ and $\mathbf{p}$ is an input vector of order R.

Considering the case of a single ADALINE with two inputs ($R = 2, S = 1$). The output of the network is given by

$$a = \sum_{j=1}^{2} w_j p_j + b \tag{4.12}$$

$$= w_1 p_1 + w_2 p_2 + b. \tag{4.13}$$

So, as for the SLP, the ADALINE has a decision boundary at

$$w_1 p_1 + w_2 p_2 + b = 0 \tag{4.14}$$

and so the ADALINE can be used to classify objects into two categories if the objects are linearly separable.

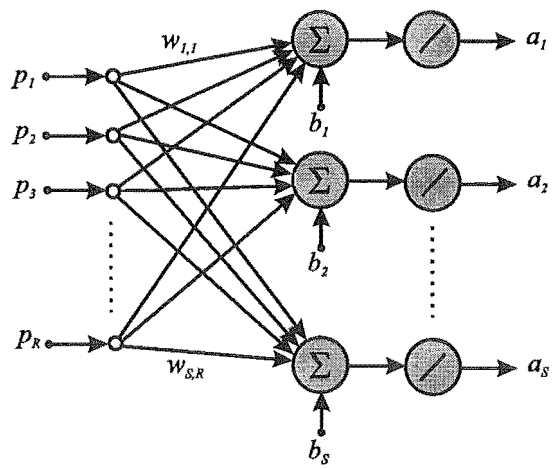


Figure 4.3 Adaptive linear combiner architecture.

4.3.2 LMS algorithm

The LMS algorithm is a supervised learning rule and so requires a training set consisting of input patterns and corresponding desired outcomes as in Equation 4.7. Consider first the case of a single neuron. To simplify the development of the LMS algorithm the weights and bias will be lumped into one vector, ω, thus

$$\omega = [\mathbf{w}^T, b]^T. \tag{4.15}$$

Similarly the input vector $\mathbf{p}$ will be re-written as

$$\rho = [\mathbf{p}^T, 1]^T. \tag{4.16}$$

In this way the output of the network can be given by

$$a = \omega^T \rho. \tag{4.17}$$

The mean square error (MSE) for the ADALINE is then given by

$$F(\omega) = E[e^2] = E[(d-a)^2] = E[(d-\omega^T\rho)^2], \tag{4.18}$$

where $E[\cdot]$ is used to denote expected value and the expectation is taken over all sets of input/output pairs.

Equation 4.18 can be expanded to give:

$$\begin{aligned} F(\omega) &= E[d^2 - 2d\omega^T\rho + \omega^T\rho\rho^T\omega] && (4.19)\\ &= E[d^2] - 2\omega^T E[d\rho] + \omega^T E[\rho\rho^T]\omega. && (4.20) \end{aligned}$$

which can be re-written in the following convenient form

$$F(\omega) = c - 2\omega^T\mathbf{h} + \omega^T\mathbf{R}\omega, \tag{4.21}$$

where $c = E[d^2]$, $\mathbf{h} = E[d\rho]$ and $\mathbf{R} = E[\rho\rho^T]$. Here the vector $\mathbf{h}$ gives the cross-correlation between the input vector and the desired outcome, while $\mathbf{R}$ is the input correlation matrix.

The stationary point of the performance index can be located by finding the gra-

dient of Equation 4.21 and equating it to zero

$$\nabla F(\boldsymbol{\omega}) = \nabla \left(c - 2\boldsymbol{\omega}^T \mathbf{h} + \boldsymbol{\omega}^T \mathbf{R} \boldsymbol{\omega} \right) = -2\mathbf{h} + 2\mathbf{R}\boldsymbol{\omega} = 0. \tag{4.22}$$

Therefore, if the correlation matrix is positive definite there will be a unique stationary point, which will be a strong minimum at:

$$\boldsymbol{\omega}^* = \mathbf{R}^{-1}\mathbf{h} \tag{4.23}$$

The existence of a unique solution depends only on the correlation matrix $\mathbf{R}$ and therefore the characteristics of the input vectors. If we could calculate the statistical quantities $\mathbf{h}$ and $\mathbf{R}$, we could find the minimum point directly from Equation 4.23. Generally speaking, however, it is not desirable to calculate $\mathbf{h}$ and $\mathbf{R}$ and even less so to calculate $\mathbf{R}^{-1}$. Widrow and Hoff had the key insight to estimate the mean squared error $F(\boldsymbol{\omega})$ by

$$\hat{F}(\boldsymbol{\omega}) = (d(k) - a(k))^2 = e^2(k), \tag{4.24}$$

where the expectation of the squared error has been replaced by the squared error at iteration k. Using the gradient of the estimate,

$$\hat{\nabla} F(\boldsymbol{\omega}) = \nabla e^2(k), \tag{4.25}$$

allows an approximate steepest descent algorithm to be generated as follows.

$$\begin{aligned} [\nabla e^2(k)]_j &= \frac{\partial e^2(k)}{\partial w_j} \\ &= 2e(k)\frac{\partial e(k)}{\partial w_j} \quad \text{for } j = 1, 2, \ldots R, \end{aligned} \tag{4.26}$$

and

$$\begin{aligned} [\nabla e^2(k)]_{R+1} &= \frac{\partial e^2(k)}{\partial b} \\ &= 2e(k)\frac{\partial e(k)}{\partial b}. \end{aligned} \tag{4.27}$$

The partial derivative of $e(k)$ with respect to the weight w_j is given by

$$\begin{aligned}\frac{\partial e(k)}{\partial w_j} &= \frac{\partial[d(k) - a(k)]}{\partial w_j} \\ &= \frac{\partial}{\partial w_j}\left[d(k) - \left(\sum_{i=1}^{R} w_i p_i(k) + b\right)\right]. \end{aligned} \tag{4.28}$$

This simplifies to

$$\frac{\partial e(k)}{\partial w_j} = -p_j(k). \tag{4.29}$$

In a similar way

$$\frac{\partial e(k)}{\partial b} = -1. \tag{4.30}$$

So the gradient of the squared error at iteration k can be written

$$\hat{\nabla} F(\boldsymbol{\omega}) = \hat{\nabla} e^2(k) = -2e(k)\boldsymbol{\rho}(k). \tag{4.31}$$

This approximation to $\nabla F(\boldsymbol{\omega})$ can now be used in the steepest algorithm which is of the form

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k - \alpha \nabla F(\boldsymbol{\omega})|_{\boldsymbol{\omega}=\boldsymbol{\omega}_k} \tag{4.32}$$

where α is a constant which represents the learning rate. So, substituting $\hat{\nabla} F(\boldsymbol{\omega})$ from Equation 4.31, for $\nabla F(\boldsymbol{\omega})$ gives

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + 2\alpha e(k)\boldsymbol{\rho}(k) \tag{4.33}$$

or

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k), \tag{4.34}$$

and

$$b(k+1) = b(k) + 2\alpha e(k). \tag{4.35}$$

These last two equations specify the LMS algorithm for the case of a single neuron; for multiple output neurons the LMS algorithm can be conveniently expressed as follows:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\mathbf{e}(k)\mathbf{p}^T(k), \tag{4.36}$$

and

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k). \tag{4.37}$$

The LMS algorithm has established itself as an important functional block of adaptive signal processing. It offers some highly desirable features:

- Simplicity of implementation (in software or hardware form).
- Ability to perform satisfactorily in an unknown environment.
- Ablity to track time variations of input statistics.

While the LMS algorithm has been presented thus far as a filter operating on a number of input channels at time instant k, it can be applied to temporal filtering equally well following the introduction of a tapped-delay-line to give a tapped-delay-line filter. In such a case, the input vector $\boldsymbol{\rho}(k)$ is then defined

$$\boldsymbol{\rho}(k) = [p(k), p(k-1), \ldots, p(k-\tau+1), 1]^T \tag{4.38}$$

for a tapped-delay-line with τ taps.

4.4 THE MULTI-LAYER PERCEPTRON

Both the perceptron learning rule and the LMS algorithm are designed to train single-layer (perceptron-like) neural networks. These have the drawback that they can only solve linearly separable classification problems. Both Rosenblatt and Widrow knew about this and knew that going to multiple layers could overcome this, but they were not able to generalize their algorithms to train these more powerful networks. In the mid-1980's the backpropagation algorithm was advanced [Rumelhart and McClelland 1986] and soon became the most widely used algorithm for training the multi-layer perceptron; it still is today.

Multi-layer perceptrons (MLPs) have a set of input nodes, one or more hidden layers of computation nodes and an output layer of computation nodes. The input vector propagates through the network in a forward direction on a layer-by-layer basis.

The backpropagation algorithm (or error backpropagation algorithm) is based on error correction learning and supervised training. The algorithm consists of two principal actions: a forward pass through the layers of the MLP followed by a backward pass through the MLP. During the forward pass an input vector is applied to the network input layer and it propagates through the network in a forward direction layer-by-layer until it produces a set of outputs at the output layer; these constitute the actual response of the network. This is then compared with the desired response to generate an error signal and this is propagated back through the network layer-by-layer. During the forward propagation, the network parameters are fixed, but during the backpropagation of the error, the synaptic weights of the network are adjusted so as to minimize the error.

A MLP has three distinctive characteristics:

1. Each neuron in the network is generally modelled with a nonlinear activation function. What is more, the nonlinear activation function must be smooth (i.e., it is differentiable everywhere) as opposed to the hardlimiter of the SLP. The sigmoid activation function is by far the most popular in use. If the layers of the MLP did not have nonlinear activation functions, the whole MLP could be reduced to an equivalent SLP.

2. There are one or more hidden layers in the MLP which enable the network to learn complex tasks by extracting progressively more meaningful features from the input vectors.

3. There is a high degree of connectivity in the network. A change in the connectivity of the network requires a change in the population of synaptic connections or a change in their weights.

4.4.1 The multi-layer perceptron architecture

Figure 4.4 depicts a 3-layer perceptron (1 output layer and 2 hidden layers). Each layer can have a different number of neurons and even a different activation function. The MLP depicted may therefore be denoted a R-$S1$-$S2$-$S3$ network. As before, superscripts are used to denote the layer to which each vector/matrix belongs, so the overall output of the MLP is given by

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3). \tag{4.39}$$

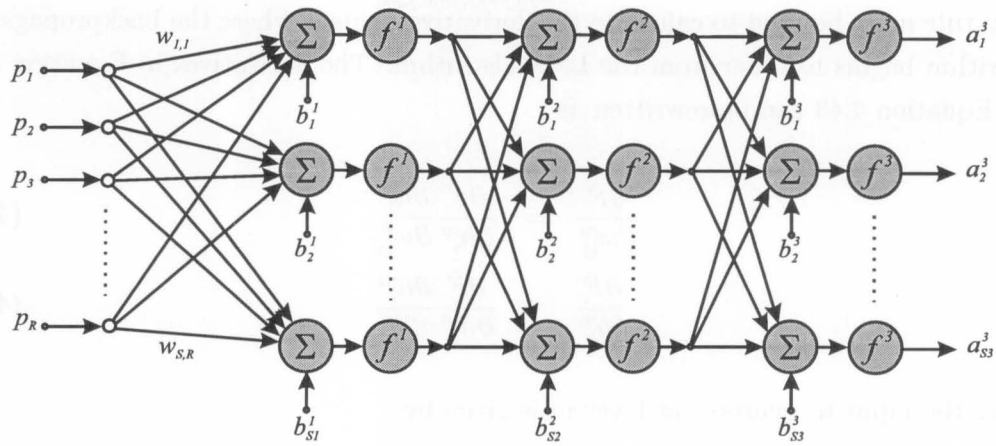


Figure 4.4 A three layer (R-$S1$-$S2$-$S3$) perceptron.

4.4.2 Backpropagation algorithm

The backpropagation algorithm is a generalization of the LMS algorithm and uses the same performance measure, i.e., the mean square error. The network needs to be presented with a set of examples of proper network behavior as given in Equation 4.7, where $\mathbf{p}_q$ represents an input vector to the system of order R and $\mathbf{d}_q$ is the corresponding desired response vector of order $S3$. The performance measure is given by:

$$F(\boldsymbol{\omega}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{d}-\mathbf{a})^T(\mathbf{d}-\mathbf{a})], \tag{4.40}$$

where $\boldsymbol{\omega}$ is the vector of network weights and bias as defined by Equation 4.15. As with the LMS algorithm, the performance measure is approximated by replacing the expectation of the squared error by the squared error at iteration k, so that

$$\hat{F}(\boldsymbol{\omega}) = \mathbf{e}^T(k)\mathbf{e}(k) = (\mathbf{d}(k)-\mathbf{a}(k))^T(\mathbf{d}(k)-\mathbf{a}(k)). \tag{4.41}$$

Using the steepest descent algorithm for the approximate mean square error gives

$$w_{ij}^m(k+1) = w_{ij}^m(k) - \alpha\frac{\partial\hat{F}}{\partial w_{ij}^m}, \tag{4.42}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha\frac{\partial\hat{F}}{\partial b_i^m}, \tag{4.43}$$

where w_{ij}^m and b_i^m represent the synaptic weight and bias, respectively, of layer m, connected to neuron i, and α is the learning rate.

Because the error is not an explicit function of the weights in the hidden layer, the

chain rule must be used to calculate the derivatives; this is where the backpropagation algorithm begins to differ from the LMS algorithm. The derivatives in Equation 4.42 and Equation 4.43 can be rewritten as

$$\frac{\partial \hat{F}}{\partial w_{ij}^m} = \frac{\partial \hat{F}}{\partial n_i^m}\frac{\partial n_i^m}{\partial w_{ij}^m}, \tag{4.44}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m}\frac{\partial n_i^m}{\partial b_i^m}, \tag{4.45}$$

where the input to neuron i at layer m is given by

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{ij}^m a_j^{m-1} + b_i^m. \tag{4.46}$$

Therefore

$$\frac{\partial n_i^m}{\partial w_{ij}^m} = a_j^{m-1}, \tag{4.47}$$

$$\frac{\partial n_i^m}{\partial b_i^m} = 1. \tag{4.48}$$

If we now define the *sensitivity* of $\hat{F}$ to changes of the input to the ith neuron in layer m as

$$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m}, \tag{4.49}$$

then Equation 4.44 and Equation 4.45 can be simplified to

$$\frac{\partial \hat{F}}{\partial w_{ij}^m} = s_i^m a_j^{m-1}, \tag{4.50}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m. \tag{4.51}$$

The approximate steepest descent algorithm can now be expressed as

$$w_{ij}^m(k+1) = w_{ij}^m(k) - \alpha s_i^m a_j^{m-1}, \tag{4.52}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m. \tag{4.53}$$

In matrix form this becomes

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T, \tag{4.54}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m, \tag{4.55}$$

where

$$\begin{aligned} \mathbf{s}^m &= \frac{\partial \hat{F}}{\partial \mathbf{n}^m} \\ &= \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix}. \end{aligned} \tag{4.56}$$

We now have to calculate the sensitivities $\mathbf{s}^m$, and in order to do so we must use the following Jacobian matrix:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix}. \tag{4.57}$$

Considering the i,j^{th} element of the matrix, we have

$$\begin{aligned} \frac{\partial n_i^{m+1}}{\partial n_j^m} &= \frac{\partial \left(\sum_{l=1}^{S^m} w_{il}^{m+1} a_l^m + b_i^{m+1} \right)}{\partial n_j^m} \\ &= w_{ij}^{m+1} \frac{\partial a_j^m}{\partial n_j^m} \\ &= w_{ij}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} \\ &= w_{ij}^{m+1} \dot{f}^m(n_j^m), \end{aligned} \tag{4.58}$$

where

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}. \tag{4.59}$$

So the Jacobian matrix can be expressed as

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1}\dot{\mathbf{F}}^m(\mathbf{n}^m). \tag{4.60}$$

Using the chain rule, the sensitivity can be rewritten in the form

$$\begin{aligned}
\mathbf{s}^m &= \frac{\partial \hat{F}}{\partial \mathbf{n}^m} \\
&= \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)\frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} \\
&= \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} \\
&= \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1}.
\end{aligned} \tag{4.61}$$

It is this recurrence relation which gives the backpropagation algorithm its name, as the sensitivities are propagated backward through the network from the last layer to the first layer:

$$\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \ldots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1. \tag{4.62}$$

The starting point, $\mathbf{s}^M$, for the recurrence relation of Equation 4.62 is obtained as follows:

$$\begin{aligned}
s_i^M &= \frac{\partial \hat{F}}{\partial n_i^M} \\
&= \frac{\partial(\mathbf{d}-\mathbf{a})^T(\mathbf{d}-\mathbf{a})}{\partial n_i^M} \\
&= \frac{\partial \sum_{j=1}^{S^M}(d_j - a_j)^2}{\partial n_i^M} \\
&= -2(d_i - a_i)\frac{\partial a_i}{\partial n_i^M}.
\end{aligned} \tag{4.63}$$

Since

$$\begin{aligned}
\frac{\partial a}{\partial n_i^M} &= \frac{\partial a_i^M}{\partial n_i^M} \\
&= \frac{\partial f^M(n_j^M)}{\partial n_i^M} \\
&= \dot{f}^M(n_j^M),
\end{aligned} \tag{4.64}$$

we can write

$$s_i^M = -2(d_i - a_i)\dot{f}^M(n_j^M) \tag{4.65}$$

or

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{d} - \mathbf{a}). \tag{4.66}$$

The complete backpropagation algorithm then consists of the following steps:

1. Present the input vector to the network, and propagate forward through the network:

$$\begin{aligned} \mathbf{a}^0 &= \mathbf{p}, \\ \mathbf{a}^{m+1} &= \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \quad \text{for } m = 0, 1, \ldots, M-1, \end{aligned} \tag{4.67}$$

2. Now propagate the sensitivities backward through the network using Equation 4.66 and Equation 4.61.

3. Finally, update the weights and biases of the network using the approximate steepest descent rule, Equation 4.54 and Equation 4.55.

4.4.3 Using the backpropagation algorithm

4.4.3.1 The nonlinear activation function

The computation of the sensitivity s_i^m for each neuron i in each layer m requires knowledge of the derivative of the activation function $f(\cdot)$ associated with that neuron. For this derivative to exist, the activation function must be continuous. For the case of the sigmoid nonlinearity, the nonlinearity is of the form

$$f(n_j) = \frac{1}{1 + \exp(-n_j)}, \tag{4.68}$$

where n_j is the net internal activity of neuron j. The derivative is then given by:

$$\begin{aligned} \frac{\partial f}{\partial n_j} &= \frac{\exp(-n_j)}{[1 + \exp(-n_j)]^2} \\ &= \left(1 - \frac{1}{1 + \exp(-n_j)}\right)\left(\frac{1}{1 + \exp(-n_j)}\right) \\ &= (1 - f(n_j))f(n_j). \end{aligned} \tag{4.69}$$

Any other nonlinear (or linear) function can be used as long as it is differentiable.

4.4.3.2 Rate of learning

The backpropagation algorithm uses the method of approximate steepest descent to minimize the squared error of the network. The synaptic weights and biases are changed in such a way as to minimize this error, the amount of change is governed by the learning rate α. The smaller the learning rate, the smaller the changes will be to the synaptic weights and biases from one iteration to the next and the smoother will be the trajectory in 'weight space'. This will be coupled, however, with a slower rate of learning. Making the learning rate too large, although speeding up the learning process, results in large changes in synaptic weights and biases which may cause the network to become unstable. So a learning rate must be found which results in adequate training speed while keeping the network stable.

4.4.3.3 Choice of network architecture

As stated earlier, multi-layer networks can be used to approximate almost any function, if they contain enough neurons in the hidden layers. The problem is that we cannot say, in general, how many layers or how many neurons are necessary for adequate performance. It is safe to say, however, that as the complexity of the system to be modelled increases, the number of hidden neurons required will be greater. Note however that having too many neurons can result in overfitting of the data [Hagan *et al.* 1996].

Figure 4.5 illustrates the network response of a 1-S1-1 MLP with sigmoid activation functions in the hidden layer and a linear neuron in the output layer. The network is set the task of representing the function $g(p)$ given by

$$g(p) = 1 + \sin\left(\frac{6\pi}{4}p\right), \quad \text{for } -2 \leq p \leq 2. \tag{4.70}$$

The figure illustrates the network response as the number of neurons in the hidden layer is increased. Unless there are at least 5 neurons in the hidden layer the network cannot adequately represent $g(p)$. It is clear that for the 1-S1-1 network with sigmoid type activation functions and a linear output neuron, a response can be produced which is a superposition of $S1$ sigmoid functions. For functions which have a greater number of inflection points, more neurons will be required in the network.

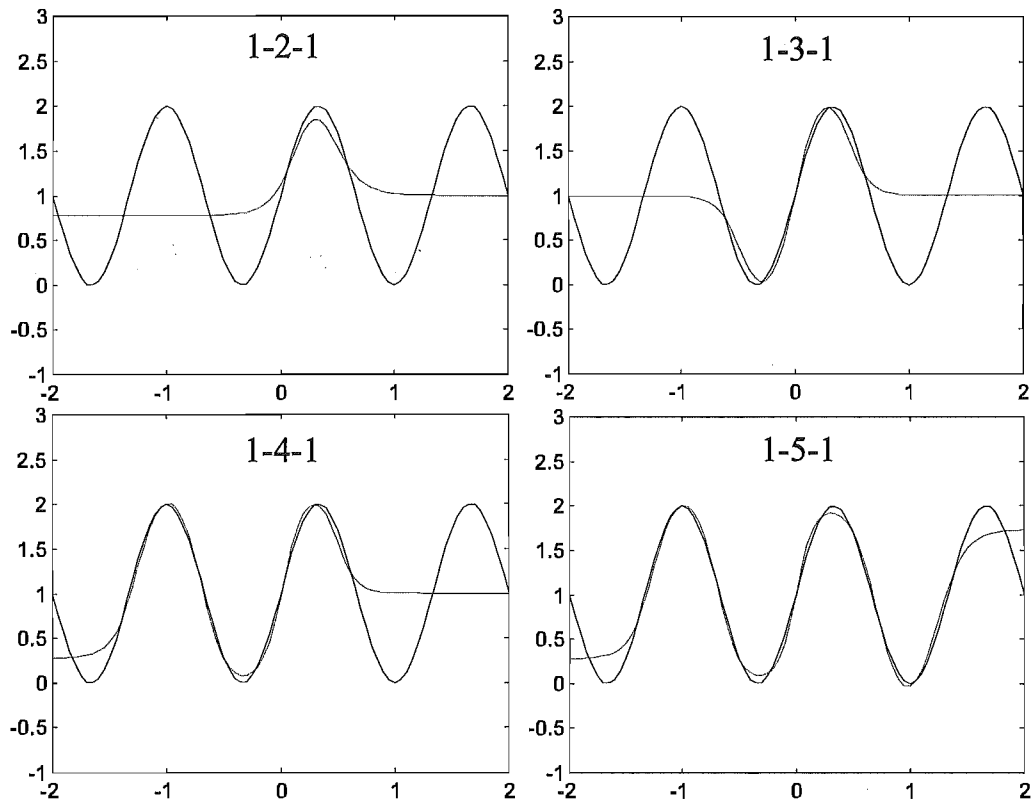


Figure 4.5 The effect of increasing the number of hidden neurons in a 1-S1-1 network. As the number of hidden neurons is increased, the network output approximates the input data better.

4.4.3.4 Convergence

In the previous subsection, we have seen how the backpropagation algorithm is capable of producing network parameters that minimize the mean squared error, although the network response did not give an accurate approximation to the desired function. This occurred because the performance of the network was inherently limited by the number of hidden neurons it contained. It is also possible to have networks which, although capable of approximating the desired functions, do not produce network parameters that accurately approximate the function, i.e., they do not converge.

Figure 4.6 illustrates two solutions to a particular example where a MLP network is to approximate a function, $g(p)$, given by:

$$g(p) = 1 + \sin(\pi p) \quad \text{for } -2 \leq p \leq 2. \tag{4.71}$$

To approximate this function a 1-3-1 network was used with sigmoid activation functions in the neurons of the hidden layer and a linear activation function for the single output neuron. Figure 4.6a represents a case where the algorithm converged to a solution that minimizes the mean square error and the final solution results in a good approximation of the desired signal. Figure 4.6b represents a case where minimizing the mean square error does *not* result in a good approximation of the desired signal, this using the same network architecture as before. The only difference between the two solutions is the initial condition. From one initial condition the algorithm converged to a global minimum, while from another initial condition the algorithm converged to a local minimum, which consequently meant that the actual response is a poor representation of the desired response of the network.

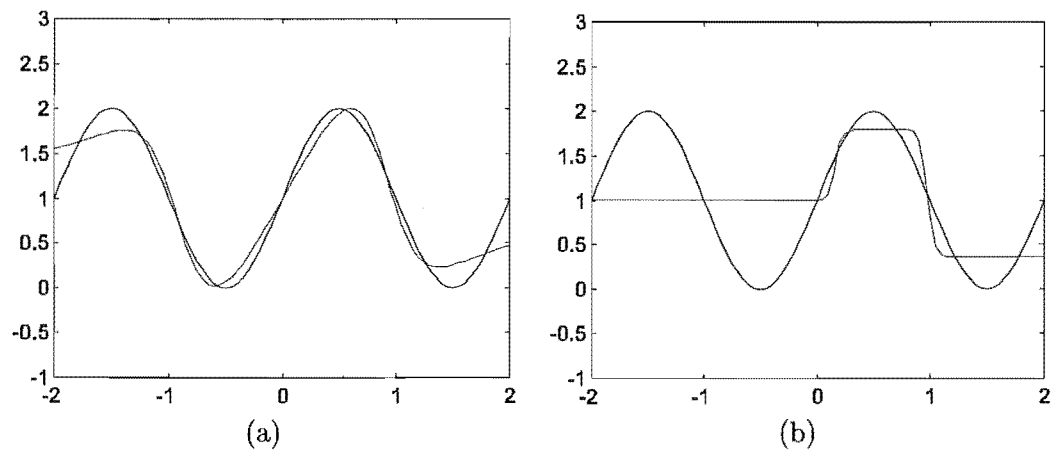


Figure 4.6 Convergence to different minima with a 1-3-1 MLP. (a) Global minimum and (b) local minimum.

It is important to note that such a problem could not have occurred with the LMS

algorithm. The mean square error performance index for the ADALINE network is a quadratic error surface with a single minimum point, so the LMS algorithm is guaranteed to converge to the global minimum point, provided a small enough learning rate is used [Hagan *et al.* 1996], [Widrow and Stearns 1985]. As the mean square error performance index for the multi-layer perceptron is generally much more convoluted, often with many local minima, the convergence of the backpropagation algorithm is not an absolute guarantee that the optimum solution has been reached (i.e., the global minimum has been reached). It is always best to try different initial conditions in order to ensure that an optimum solution has been obtained [Hagan *et al.* 1996], [Haykin 1994].

4.4.3.5 Generalization

In most cases the multi-layer perceptron is trained with a finite set of input vector/desired response pairs. The training set is usually representative of a much larger population of possible input vector/desired response pairs. An important requirement for the network is therefore that it can generalize well what it has learned to the total population.

Consider the following example training set which is obtained by sampling the following function at discrete points between -2 and 2:

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right). \tag{4.72}$$

Figure 4.7a shows the response of a 1-2-1 network that has been trained on this data (11 input/output sets). The '+' symbols represent the training data and the dotted line depicts the response of the network to novel input data. We can see that the network response is an accurate description of $g(p)$ using training set alone and that the network generalizes well to novel data. Figure 4.7b shows the response of a 1-9-1 network that has been trained on the same data set. Once more the network response is an accurate description of the training set data, however, the network does not generalize well at all to novel data.

In the case of Figure 4.7b; as we have a 1-9-1 network we have a total of 28 adjustable parameters (18 weights and 10 biases) whilst we have only 11 data points in the training set – so the 1-9-1 network has too much flexibility. The 1-2-1 network has only 7 parameters and therefore is much more restricted in the types of functions it can implement.

This leads us to say that: *For a network to be able to generalize, it should have fewer parameters than there are data points in the training set* [Hagan *et al.* 1996].

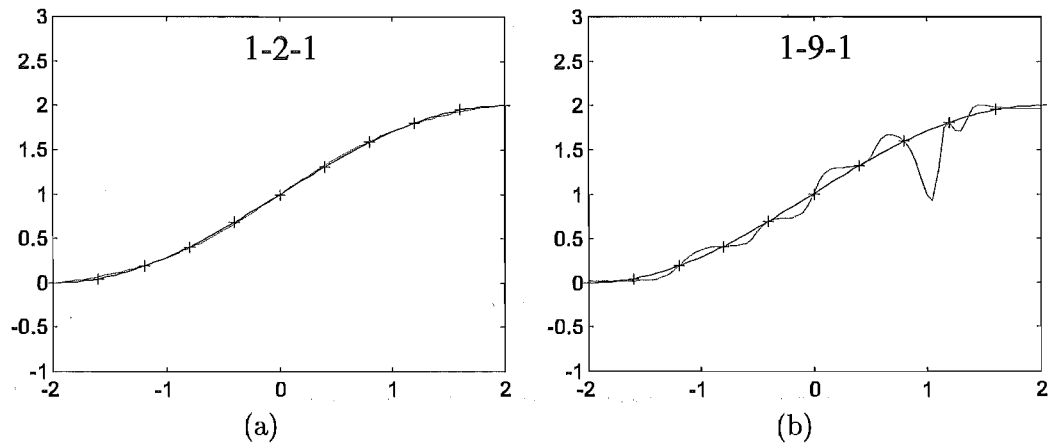


Figure 4.7 Network approximation of $g(p)$ with (a) a 1-2-1 network and (b) a 1-9-1 network.

4.4.3.6 Initial conditions

It has already been stated that the convergence to a global minimum in the multi-layer perceptron network relies heavily on the initial conditions of the network. The choice of different initial conditions can lead to wildly varying outcomes for a given network. Ideally the initial conditions of the network, i.e., the initial values for the synaptic weights and biases, should be uniformly distributed inside a small range. The reason for making the range small is to reduce the likelihood of any neuron in the network becoming saturated. A saturated neuron is one whose response is constant for variations in net input (for the log-sigmoid type activation function, the output saturates at -1 and $+1$). If the neurons are in saturation, the backpropagation algorithm will produce small error gradients and hence little change will be made to the synaptic weights and biases, resulting in the network becoming 'stuck' at a particular error level. This corresponds to a "saddle-point" on the mean square error surface. However, it is not desirable to make the range of initial conditions too small, as it may in turn cause the error gradients to be small and result in slow initial learning. The best values to use to initialise the network, are those that place the net input of each neuron in the linear portion of the activation function, away from saturation [Nguyen and Widrow 1990].

4.4.4 Improving the backpropagation algorithm

Although a major breakthrough in neural network research, the basic backpropagation algorithm is considered too slow for most practical applications. Several variations of the backpropagation algorithm exist and seldom is the algorithm used without some form of heuristic technique being used in order to speed up the learning process ([Vogl *et al.* 1988], [Jacobs 1988], [Tollenaere 1990] and [Rigler *et al.* 1990]). Another way in which the algorithm is speeded up is to use alternative, numerical opti-

mization techniques (i.e., techniques other than steepest descent) (e.g., [Shanno 1990], [Barnard 1992], [Battiti 1992] and [Charalambous 1992]).

Heuristic modifications to the standard backpropagation algorithm, whilst often providing very fast convergence for some problems, are also coupled with two major drawbacks. The first drawback is that all modifications require that several parameters be set up, with the more complex modifications requiring quite a few parameters. The choice of the parameters is often highly problem dependent and the performance of the modified algorithm is quite often highly sensitive to the choice of these parameters. The second drawback is that the modified algorithms can sometimes fail to converge to a satisfactory solution on problems for which the standard algorithm will eventually find a solution.

Although other numerical optimization techniques (other than steepest descent) can be used to alter the performance of the algorithm, it can still be described as "backpropagation" as the error is still propagated back through the layers of the network.

4.4.4.1 Momentum

Figure 4.8 shows the mean square error performance surface of a 1-2-1 multi-layer perceptron with log-sigmoid type activation functions in each neuron. The figure illustrates the mean square error whilst only the synaptic weights given by w_{11}^1 and w_{11}^2 are adjusted, and the remaining network parameters are kept constant.

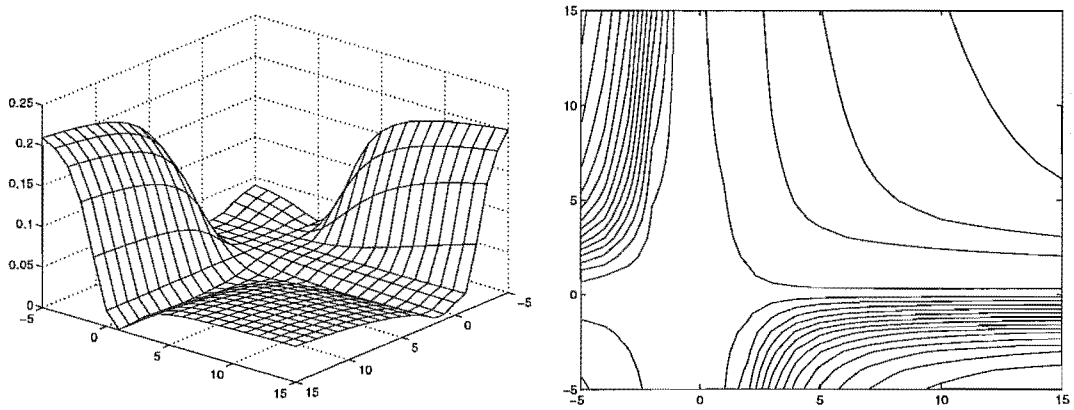


Figure 4.8 Mean square error surface versus w_{11}^1 and w_{11}^2 for 1-2-1 network.

Figure 4.9a shows the trajectory of the mean square error as the two parameters w_{11}^1 and w_{11}^2 are adjusted. The initial condition is at label "x" and it can be seen that the algorithm does eventually converge to the optimal solution (given by the label "o") but convergence takes place after many iterations at this particular learning rate.

Figure 4.9b shows what happens if we were to increase the learning rate in order for the algorithm to converge faster. Whilst the algorithm converges faster whilst traversing the initial flat surface, once the algorithm reaches the narrow valley that contains the minimum point, the algorithm begins to diverge and hence the system becomes unstable.

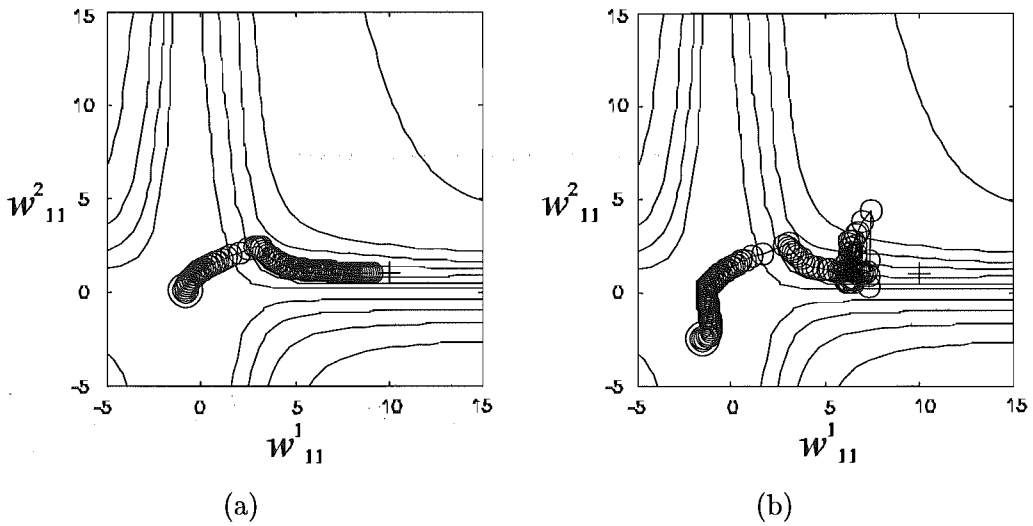


Figure 4.9 Two backpropagation trajectories with (a) small and (b) large learning rates.

Applying momentum to the algorithm has the effect of smoothing out the oscillations of the trajectory. Momentum is implemented in the form of a first-order low-pass filter, as follows:

$$y(k) = \gamma y(k-1) + (1-\gamma)x(k), \tag{4.73}$$

where $x(k)$ is the input to the filter, $y(k)$ is the output and γ the momentum coefficient which is

$$0 \leq \gamma \leq 1. \tag{4.74}$$

At $\gamma = 0$ no damping (or smoothing) takes place, whilst the closer that γ gets to 1 the more damped the response gets and hence the slower the response.

From Equation 4.54 and Equation 4.55 the synaptic weight update step is

$$\Delta \mathbf{W}^m(k) = -\alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T, \tag{4.75}$$

$$\Delta \mathbf{b}^m(k) = -\alpha \mathbf{s}^m. \tag{4.76}$$

The *momentum modified* backpropagation algorithm is thus given by

$$\Delta \mathbf{W}^m(k) = \gamma \Delta \mathbf{W}^m(k-1) - (1-\gamma)\alpha \mathbf{s}^m(\mathbf{a}^{m-1})^T, \quad (4.77)$$

$$\Delta \mathbf{b}^m(k) = \gamma \Delta \mathbf{b}^m(k-1) - (1-\gamma)\alpha \mathbf{s}^m. \quad (4.78)$$

Applying the momentum modification to the example of Figure 4.9b with the momentum set at $\gamma = 0.8$ gives Figure 4.10. We can see here that the algorithm is now stable and results in a speedy convergence to the minimum point. In general it can be said that momentum allows us to use a larger learning rate (and hence speeding up convergence) whilst maintaining the stability of the algorithm.

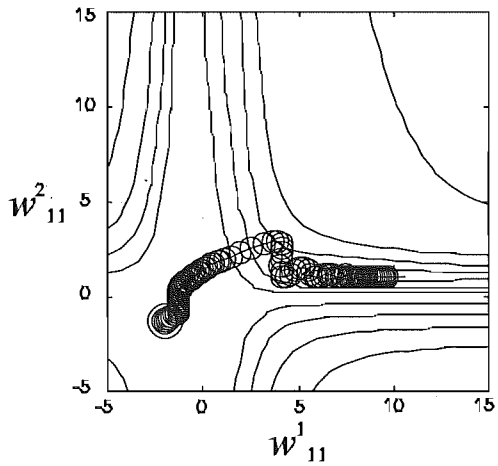


Figure 4.10 Trajectory with momentum.

4.4.4.2 Variable learning rate

As has already been stated, the error surface for the multi-layer perceptron is convoluted, consisting of many local minima - as well as a global minimum. The error surface consists of many 'flat' surfaces (points where the gradient is equal to zero) as well as many 'steep' surfaces (points on the error surface which has large gradients). It is easy to see that the speed of convergence can be enhanced if the learning rate was allowed to increase on 'flat' parts of the error surface, and allowed to decrease on 'steep' parts - this could be done whilst still maintaining stability. The trick is to determine *when* to adjust the learning rate and *by how much.*

For a very simple variable learning rate adaptive learning algorithm, the learning rate is adjusted according to the following rules [Vogl *et al.* 1988]:

1. If the mean square error increases by more than some set percentage after a weight update, then the weight update is discarded, the learning rate is reduced

by some fixed amount and the momentum coefficient γ is set to zero (if it is used at all).

2. If the mean square error decreases after a weight update, then the weight update is accepted and the learning rate is multiplied by same factor greater than one, and γ is set to its previous value if it had been set to zero.

3. If the mean square error increases by less than the set percentage of 1 above, then the weight update is accepted but the learning rate and the momentum coefficient are left unchanged.

Of the many variations in this variable learning rate algorithm, there is the *delta-bar-delta* learning rule [Jacobs 1988]. In this version of the algorithm each synaptic weight or bias has its own learning rate. The learning rate for a particular network parameter is increased if the parameter change has been in the same direction for several iterations. If there is an alternating change in direction for the parameter, the learning rate is reduced. The delta-bar-delta rule is employed in the MRANC system described in Chapter 8. Another algorithm similar to the delta-bar-delta algorithm is the *perSAB* algorithm [Tollenaere 1990].

4.4.4.3 Pattern vs batch mode training

In a practical application of the backpropagation algorithm, learning results from the many presentations of a prescribed set of training examples to the multi-layer perceptron. One complete presentation of the entire training set during the learning process is called an *epoch*. The learning process proceeds, from epoch to epoch, until the synaptic weights and biases stabilize and the mean square error over the entire training set converges to some minimum value. For a given training set, backpropagation learning may proceed in one or two basic ways:

1. **Pattern mode training.** In pattern mode training synaptic weights and biases are updated after the presentation of each training example.

2. **Batch mode training.** In batch mode training, updating of the synaptic weights and biases is held off until *all* input/output pairs forming an epoch have been presented to the network.

Pattern mode training is preferred from an "on-line" training/operation point of view because it requires *less* local storage for each synaptic connection. What is more, if the input patterns are presented in a random order, the search in weight space becomes stochastic in nature, which makes it less likely for the backpropagation algorithm to be trapped in a local minimum. On the other hand, batch mode training does provide a more accurate estimate of the gradient vector. The final choice between pattern or batch modes of training is highly problem specific [Hertz *et al.* 1991].

4.4.4.4 Conjugate gradient

The standard backpropagation algorithm uses the method of steepest descent. While the steepest descent is possibly the simplest algorithm in numerical optimization, it is often slow in converging. Newton's method is much faster but requires that the Hessian matrix and its inverse be calculated [Hagan *et al.* 1996], [Zurada 1992]. A compromise can be found in the conjugate gradient algorithm.

The conjugate gradient algorithm, as applied to a quadratic performance index, can be summarised in the following steps [Hagan *et al.* 1996]:

1. Select the first search direction $\mathbf{p}_0$ to be the negative of the gradient, that is,

$$\mathbf{p}_0 = -\mathbf{g}_0, \tag{4.79}$$

   where

$$\mathbf{g}_k = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}. \tag{4.80}$$

2. Take a step along the first search direction, selecting the learning rate α_k to minimize the function along the search direction:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k. \tag{4.81}$$

3. Select the next search direction, $\mathbf{p}_k$, such that it is orthogonal to $\Delta\mathbf{g}_k$ using:

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}, \tag{4.82}$$

   where

$$\beta_k = \frac{\Delta\mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}. \tag{4.83}$$

4. If the algorithm has not converged, continue from step 2.

So, the conjugate gradient works by carefully selecting the directions in which minimization takes place. In this way, for a quadratic error surface with n parameters, the exact minimum will be reached in at most n searches. However, the standard conjugate gradient algorithm cannot be applied directly to the multi-layer perceptron neural network because the performance index is not quadratic. This will affect the

standard algorithm in two ways, firstly the calculation of the learning rate at iteration k (α_k) will have to be done using some iterative procedure for locating the minimum of a function in a specified direction, and secondly the exact minimum (global minimum) will not normally be reached in finite number of steps.

In order to calculate optimum learning rate α_k the Golden section search algorithm is used [Scales 1985]. To avoid the failure of the algorithm to converge to the minimum, the simplest method is to reset the search direction to the steepest descent direction (negative of the gradient) after n iterations [Scales 1985].

The algorithm generally converges in many fewer iterations than the standard (steepest descent) backpropagation algorithm, although this is a little deceptive as the algorithm requires much computation and iterative calculations for each iteration of the backpropagation algorithm. Nevertheless, the conjugate gradient backpropagation algorithm has been shown to be one of the fastest batch training algorithms for multi-layer networks [Charalambous 1992].

4.4.4.5 The Levenberg-Marquardt algorithm

Once more the backpropagation algorithm can be modified by choosing a numerical optimization technique other than steepest descent. The Levenberg-Marquart algorithm is a variation on Newton's method. The principle behind Newton's method is to locate the stationary point of the quadratic approximation of the performance index and step to that point in order to minimize the performance index. (If the performance index is quadratic by nature, as for the ADALINE, then Newton's method will converge in a single step). While Newton's method usually produces faster convergence than steepest descent, it is possible for the algorithm to oscillate or diverge, or to converge to a saddle point. Steepest descent, in contrast, is guaranteed to converge provided the learning rate is not too large. Another problem with Newton's method is that it requires the computation and storage of the Hessian matrix ($\nabla^2 F(\mathbf{x})$) as well as its inverse. The Levenberg-Marquardt algorithm uses the principle of Newton's method, i.e., approximates the performance as quadratic, but tackles the divergence problem possible with Newton's method by reducing to steepest-descent steps whenever divergence begins to occur.

The algorithm works by starting off with a step calculated by an approximate Newton's method (Gauss-Newton), if this step does not yield a smaller value for the performance index another, smaller, step is taken and so on until eventually a decrease in the performance index should be seen. In the limit the algorithm resembles the steepest-descent algorithm. The algorithm provides a compromise between the speed of Newton's method and the guaranteed convergence of steepest descent.

As for the conjugate gradient method, the Levenberg-Marquardt algorithm applied

to the multi-layer perceptron can be seen to converge in much fewer iterations than standard (steepest descent) backpropagation. This is, however, coupled with much more computation per iteration than any of the other algorithms as it involves the calculation, storage and inversion of an approximation to the Hessian matrix. This notwithstanding, the Levenberg-Marquardt backpropagation algorithm can be said to be the fastest neural network training algorithm around for moderate numbers of network parameters [Hagan and Menhaj 1994].

Chapter 5

THE SELF-ORGANISING FEATURE MAP

5.1 INTRODUCTION

This chapter considers the special class of artificial neural networks known as the self-organising feature map (SOFM). These are unsupervised (hence "self-organising") neural networks and, in particular, are based on the competitive learning rule described in Chapter 3. The SOFM is an artificial neural network (ANN) developed by Teuvo Kohonen at the Helsinki University of Technology [Kohonen 1982] and consists of a single layer feed-forward network or lattice, the neurons of which become specifically tuned to various input patterns through a self-organising process. The spatial location of a neuron in the lattice then corresponds to a particular feature, or group of features, of the input patterns. The SOFM roughly resembles the way similar *computational maps* are formed in the cortex of the brain. The learning results achieved seem very natural, indicating that the adaptive processes at work in the SOFM may be similar to those at work in the brain [Kohonen 1990].

First the biological motivation for such a network is briefly discussed, followed by the introduction of the SOFM model and training algorithm as developed by Kohonen. The importance of the proper choice of parameters for good training of the SOFM is then discussed. Kohonen also developed a supervised learning scheme, known as Learning Vector Quantization (LVQ), which is introduced as a means of "fine-tuning" the trained SOFM and allows the SOFM to be used as a pattern classifier. In the following chapter a number of computer simulations are carried out on SOFMs used as pattern classifiers. The simulations are carried out in order to obtain the optimum working values of the parameters needed to train and "fine-tune" the SOFM well.

5.2 BIOLOGICAL MOTIVATION FOR THE SOFM MODEL

As has already been discussed briefly in earlier chapters, many regions of the brain are organised in such a way that the different sensory inputs are represented by *topologically ordered computational maps.* Different sensory inputs, such as tactile, visual

and acoustic are mapped onto different areas of the cerebral cortex in a topologically ordered manner. Such spatial ordering of the sensory information constitutes a basic building block in the information-processing infrastructure of the nervous system. A computational map can be defined as an array of neurons that represent slightly different filters which operate on the sensory information-bearing signals in parallel. Such a map could then be used to transform input signals into a place-coded probability distribution that represents the computed values of parameters by sites of maximum relative activity within the map [Knudsen *et al.* 1987].

Cytoarchitectural maps of the cerebral cortex have been produced by many researchers [Brodal 1981], [Shepherd 1988], where different areas of the cerebral cortex are identified by their thickness and the types of neurons within them. Of the many areas on the cortex, some of the most important specific areas are the motor cortex, the somatosensory cortex, the visual cortex and auditory cortex. Distinct areas of the cortex can be mapped to these functions of the nervous system. These *cortical maps* are not entirely genetically predetermined, but develop along with the early development of the nervous system, although exactly *how* this development takes place is not entirely known. Once formed, the cortical maps retain some plasticity, allowing the maps to adapt to accommodate changes in the environment or the sensors themselves. Different cortical regions have different plasticities [Kaas *et al.* 1983].

5.2.1 Kohonen's basic feature-mapping model

The basic premise for the formation of topographic maps is that the spatial location of an output neuron in the topographic map corresponds to a particular domain or feature of the input data [Kohonen 1990]. Output neurons of a topographic map are usually arranged such that each neuron has a set of neighbours. One–dimensional (1D), two–dimensional (2D) or even three-dimensional (3D) *lattices* are ideal structures for such a setup, although the dimensions of the lattice are usually restricted to 1D or 2D simply for ease of visualisation. This is because the original idea behind the SOFM was for it map similar input patterns onto contiguous locations in the output space in order to visualise the groupings of the inputs based on their similarities [Kohonen 1995].

The first demonstration of self-organisation through a computer simulation was by von der Malsburg [1973], one of the pioneers of self-organising studies. Willshaw and von der Malsburg [1976] proposed a model on biological grounds, to explain the problem of retinotopic mapping from the retina to the visual cortex in higher vertebrates. Topologically ordered mapping was made possible through self-organising, although the Willshaw-von der Malsburg model is restricted to mappings where the input dimension is the same as the output dimension.

A second model was introduced by Kohonen [1982] which was not intended to explain any neurobiological details. This model tries to capture the essential features

of the computational maps in the brain while remaining computationally feasible. It is this model which forms the basic building block for the SOFM.

5.2.2 The mechanism of lateral feedback

In neural networks, lateral feedback describes the special form of feedback that is dependent on lateral distance from the point of its application. Figure 5.1 depicts a 1D lattice of neurons with both feedforward connections and lateral connections. The input signals are applied in parallel to the neurons via the feedforward connections. Each neuron responds according to the weighted sum of the input signals. The lateral feedback at each neuron, on the other hand, can produce both excitatory and inhibitory effects, depending on the distance from each neuron. The form of the lateral feedback is usually depicted by a function as shown in Figure 5.2 (following biological motivation). Three distinct areas of lateral interaction between neurons can be distinguished from this figure:

1. A local, short-range area of lateral excitation.

2. A penumbra of inhibitory action.

3. A third area of weaker excitation surrounding the inhibitory penumbra; this third area of interaction is usually ignored.

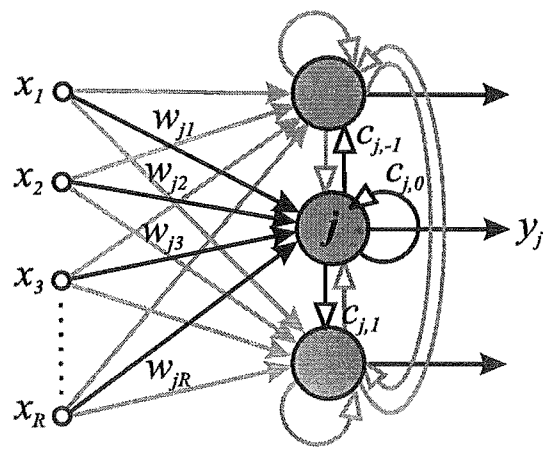


Figure 5.1 A one dimensional lattice of neurons incorporating both feedforward connections and lateral connections.

Kohonen showed that for such a network, the activity of the network will be concentrated into local clusters called *activity bubbles* [Kohonen 1989a] and that the location of each activity bubble will be determined by the nature of the input signals.

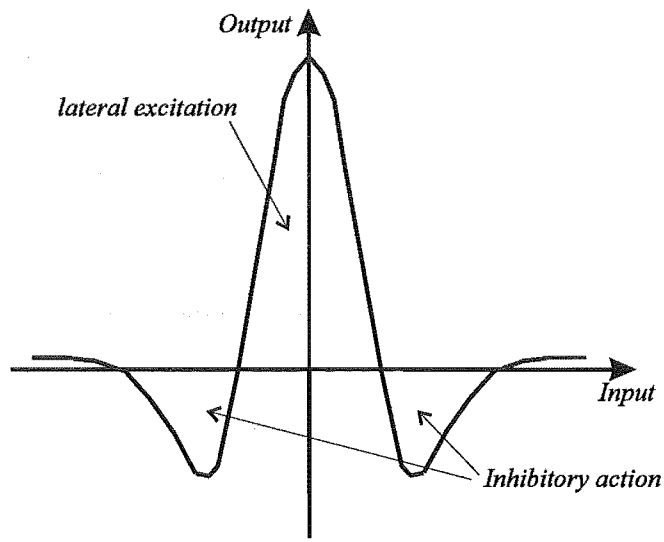


Figure 5.2 Lateral feedback as a function of distance from the most active output neuron.

For the network given in Figure 5.1, let $x_1, x_2, \ldots, x_R$ denote the inputs to the network for an input of order R. Let the synaptic weights in the feedforward part of the network be denoted by $w_{j1}, w_{j2}, \ldots, w_{jR}$ for neuron j. Let $c_{j,-K}, \ldots, c_{j,-1}, c_{j,0}, c_{j,1}, \ldots, c_{j,K}$ represent the *lateral* feedback weights connected to neuron j for a "radius" of K. Finally, for N neurons in the network, let $y_1, y_2, \ldots, y_N$ denote the output signals of the network. The response of neuron j can now be given as

$$y_j = \phi\left(\sum_{l=1}^{R} w_{jl}x_l + \sum_{k=-K}^{K} c_{jk}y_{j+k}\right), \quad \text{for } j = 1, 2, \ldots, N \tag{5.1}$$

where $\phi(\cdot)$ is some nonlinear limiting function.

The solution to the nonlinear equation Equation 5.1 is found iteratively, using a *relaxation* technique. Equation 5.1 can be reformulated as

$$y_j(n+1) = \phi\left(\sum_{l=1}^{R} w_{jl}x_l + \beta \sum_{k=-K}^{K} c_{jk}y_{j+k}(n)\right), \quad \text{for } j = 1, 2, \ldots, N \tag{5.2}$$

at discrete time interval n. A parameter β is introduced in Equation 5.2 in order to control the rate of convergence in the relaxation process.

What tends to happen is that if β is large enough, in the final state of the system (corresponding to $n \to \infty$) the values of y_j tend to concentrate inside a spatially bounded cluster, or activity bubble. What is more, the activity bubble is centred at a point where the initial response $y_j(0)$ was at a maximum due to the stimulus

$\sum_{l=1}^{R} w_{jl}x_l$. The *width* of the activity bubble depends on the ratio of the excitatory and inhibitory lateral connections.

If β is too small, then the formation of activity bubbles will be prevented by too much negative feedback.

Figure 5.3a and Figure 5.3b represent computer simulations of the bubble formation on a 1D array of 50 neurons ($N = 50$). The inputs (x_l) and weights (w_{jl}) are assumed to have fixed values which produce a net stimulus ($\sum_{l=1}^{R} w_{jl}x_l$) acting on neuron j given by

$$I_j = 2\sin\left(\frac{\pi j}{50}\right), \quad 0 \leq j \leq 50. \tag{5.3}$$

This results in a half sinusoid with a peak value of 2 at the middle neuron and zero at either end of the lattice. The feedback factor β is varied in order to see the formation of an activity bubble centered at the point of maximum excitation due to the feedforward weights alone. For these experiments the nonlinear function $\phi(\cdot)$ is taken to be a piecewise-linear function as shown in Figure 5.3c, whereas the "Mexican-hat" function of c_{jk} is approximated by the values given in Figure 5.3d. Figure 5.3a shows the formation of an activity bubble after 10 steps of the iteration process. The large value assigned to β ($\beta = 2$) results in the gradual formation of an activity bubble centred around the maximally responsive neuron. Figure 5.3b shows the activity after 10 iterations for small β ($\beta = 0.75$), in this case the formation of an activity bubble is prevented by too much negative feedback.

So, given that activity bubbles are formed due to lateral feedback in such arrays, Kohonen extracted the following points:

1. The output at neuron j can be idealized such that

$$y_j = \begin{cases} a, & \text{neuron } j \text{ is inside the bubble} \\ 0, & \text{neuron } j \text{ is outside the bubble} \end{cases} \tag{5.4}$$

   where a is the limiting value of the nonlinear function $\phi(\cdot)$.

2. The formation of activity bubbles centred around the maximally responsive neuron due to the feedforward synaptic weights alone, allows for a computational "shortcut" so as to emulate the effect of having lateral feedback. In short, lateral feedback may be done away with altogether by introducing a *topological neighbourhood of active neurons* that corresponds to the activity bubble.

3. Adjusting the size of the neighbourhood described in the previous point can simulate the adjustment of lateral connections. Making the neighbourhood size larger

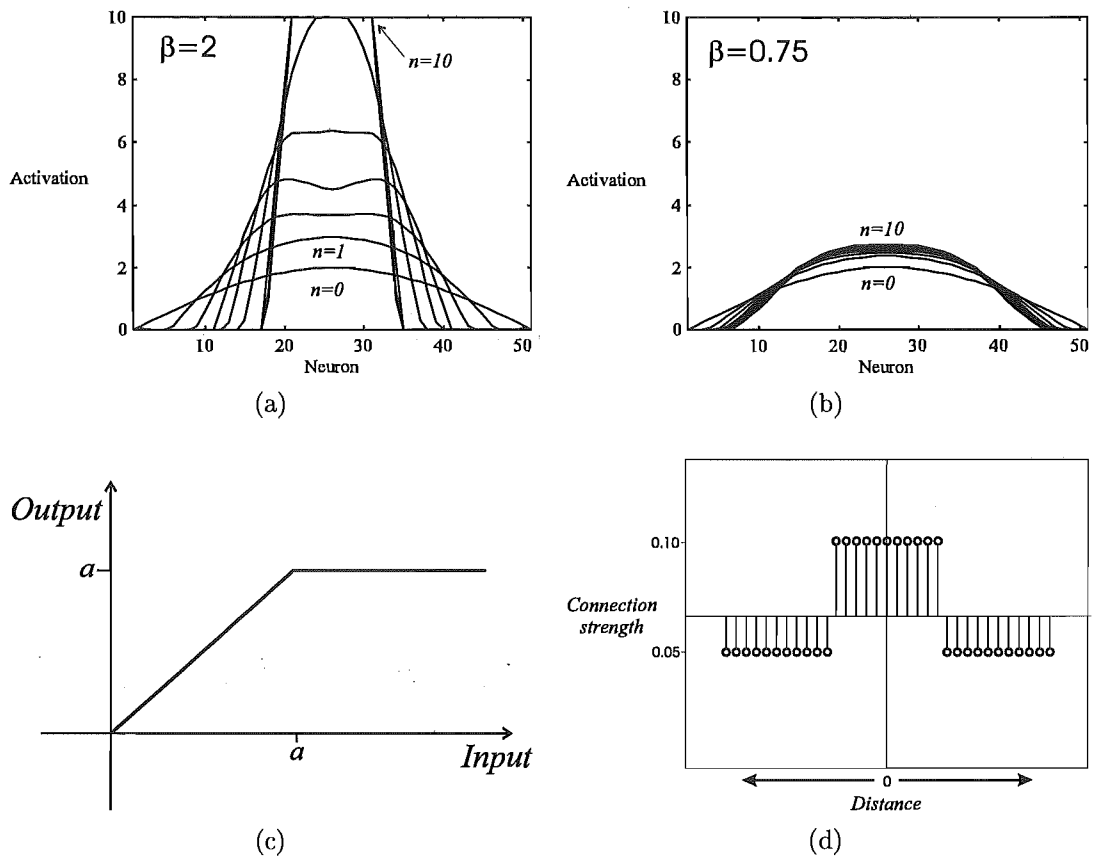


Figure 5.3 Computer simulation of bubble formation for a 1D array of 50 neurons for (a) large β and (b) small β. The piece-wise linear activation function $\phi(\cdot)$ is described in (c) and the lateral interaction function is approximated by the function of (d).

simulates making the lateral *excitatory* feedback stronger, whereas making the neighbourhood narrower corresponds to making the lateral *inhibitory* feedback stronger.

These points, made by Kohonen, can be used on maps of any dimensionality in developing the SOFM learning algorithm, but will be explained in the next section in the context of 1D and 2D SOFMs.

5.3 THE SELF-ORGANISING FEATURE MAP ALGORITHM

The SOFM as developed by Kohonen consists of a 1D or 2D lattice of neurons fully connected to a set of inputs of arbitrary dimensions. As described in Section 3.4.4 these lattice networks are simply single-layer, fully-connected, feed-forward networks for which the spatial arrangement of the neurons has significant meaning. Once trained, the spatial location of a neuron in the network corresponds to a particular domain of input patterns. The self-organising process is an iterative process, where many input

patterns are presented to the network, one at a time. Each input pattern presented to the SOFM causes a corresponding localized group of neurons in the lattice to be active.

Consider the 2D lattice of neurons shown in Figure 5.4. Each neuron in the lattice is accompanied by a synaptic weight vector $\mathbf{m}_i$ such that

$$\mathbf{m}_i = [m_{i1}, m_{i2}, \ldots, m_{iR}]^T, \quad \text{for } i = 1, 2, \ldots, N. \tag{5.5}$$

The input vector is given by $\mathbf{x}$ such that

$$\mathbf{x} = [x_1, x_2, \ldots, x_R]^T. \tag{5.6}$$

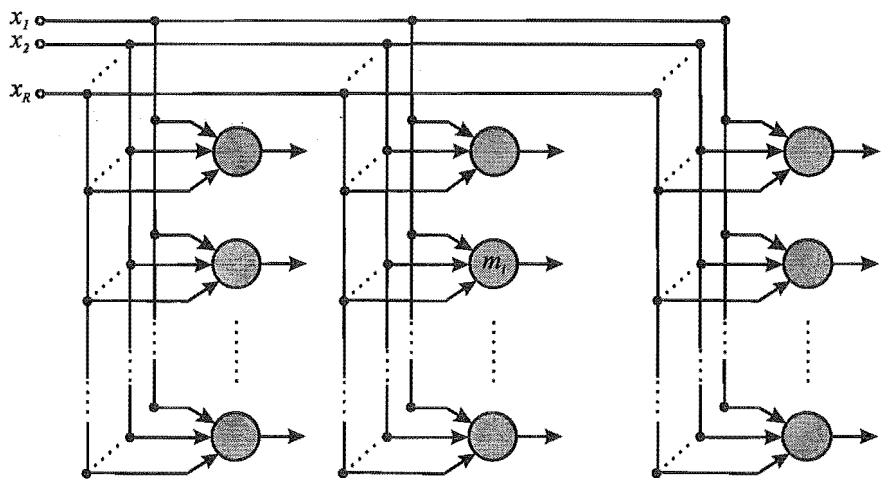


Figure 5.4 A 2D lattice of neurons depicting the synaptic weight vectors $\mathbf{m}_i$ at each neuron i (for $i = 1, 2, \ldots, N$).

Once the input vector is presented to the SOFM, the localized area of response of the network to the input must be found. This is done by comparing the input vector $\mathbf{x}$ to each $\mathbf{m}_i$ and using a matching criterion to find the "best" matching $\mathbf{m}_i$. Many matching criteria may be used as described by Kohonen [1995], although the two most widely used criteria are:

1. inner product: find $\mathbf{m}_c$ such that

$$\mathbf{x}^T \mathbf{m}_c = \max_i \left\{ \mathbf{x}^T \mathbf{m}_i \right\}, \tag{5.7}$$

2. Euclidean distance: find $\mathbf{m}_c$ such that

$$\| \mathbf{x} - \mathbf{m}_c \| = \min_i \left\{ \| \mathbf{x} - \mathbf{m}_c \| \right\}. \tag{5.8}$$

In each case, the neuron of index c is called the "winner" or "winner-takes-all" neuron.

Kohonen suggests that the matching criterion best suited for a given problem depends on the problem at hand. However, if the SOFM algorithm is to be used for natural signal patterns relating to metric vector spaces, the Euclidean norm is the better of the two to use [Kohonen 1990].

The actual *magnitude* of the response is of no great importance but, depending on the application of interest, the outcome of the network could be either the index of the winning neuron (c) or its synaptic weight vector ($\mathbf{m}_c$).

The simplest approach to training could be as follows:

Once the best matching weight vector $\mathbf{m}_c$ is found, it is updated to match even more closely the current $\mathbf{x}$. If the matching criterion used is the Euclidean distance then the distance between $\mathbf{x}$ and $\mathbf{m}_c$ is decreased, and all the other weight vectors $\mathbf{m}_i$ with $i \neq c$, are left intact. In this way the weight vectors tend to become specifically "tuned" to different domains of the input variable $\mathbf{x}$.

Thus following the imposition of the input at time step t,

$$\begin{aligned} \mathbf{m}_i(t+1) &= \mathbf{m}_i(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)], \quad \text{for } i = c, \\ \mathbf{m}_i(t+1) &= \mathbf{m}_i(t), \qquad\qquad\qquad\quad \text{for } i \neq c, \end{aligned} \tag{5.9}$$

where $\alpha(t)$ is the *gain term* or *learning rate* $(0 < \alpha(t) < 1)$.

However, since the aim of the SOFM is to form *ordered* maps, it is essential that the weight vectors $\mathbf{m}_i$ do not learn independently of each other, but as topologically related subsets where a similar correction is applied to topologically close weight vectors. During the training process these subsets will change depending on the winning neuron; in this way the net change applied to the weight vector of any neuron will tend to be smoothed out in the long run. One way to achieve these subsets is to use lateral connections between neighbouring neurons and use the method described in the previous section to form activity bubbles which encompass the winning neuron as well as a number of its neighbours. However, using Kohonen's computational "short-cut", it is possible to define a topological neighbourhood function, N_c, that is centred around the winning neuron c and encompasses a number of the surrounding neurons. So at each learning step, the weight vectors of neurons within the neighbourhood function defined by N_c are updated whereas those of neurons outside of the neighbourhood function are left intact.

The actual width (or radius) of N_c can be time-variable and it has been experimentally shown that it is advantageous to let N_c be very wide in the beginning and shrink monotonically with time. The idea is that the initially large N_c first induces global ordering in the $\mathbf{m}_i$ of the SOFM and as the N_c shrinks, then the spatial resolution of the map improves without losing the acquired global ordering [Kohonen 1990]. This

corresponds to initially using a strong positive lateral feedback and then enhancing the negative lateral feedback. If the N_c were to shrink to $N_c = \{c\}$, then only the winning neuron's weights would be adjusted which leads to simple competitive learning without spatial ordering.

So, once spatial ordering is introduced, the weight update equations of Equation 5.9 become

$$\begin{aligned} \mathbf{m}_i(t+1) &= \mathbf{m}_i(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)], \quad \text{if } i \in N_c(t), \\ \mathbf{m}_i(t+1) &= \mathbf{m}_i(t), \qquad\qquad\qquad\qquad\quad \text{if } i \notin N_c(t). \end{aligned} \tag{5.10}$$

The scalar learning rate $\alpha(t)$ should also decrease monotonically with time such that initially, when the N_c is large, the changes applied to the $\mathbf{m}_i$ are greater than the changes made during the stage where N_c is small, and only local spatial ordering is taking place around the winning neuron c.

So the algorithm therefore leads to a *topological ordering* of the SOFM in the sense that neurons that are adjacent in the lattice will tend to have similar synaptic weight vectors $\mathbf{m}_i$. Figure 5.5 summarizes the SOM algorithm, breaking it up into its major components (after initialization) which are: sampling, similarity matching, and updating. These steps are repeated until the training is complete and the SOFM weights have settled to their final values [Kohonen 1990], [Haykin 1994].

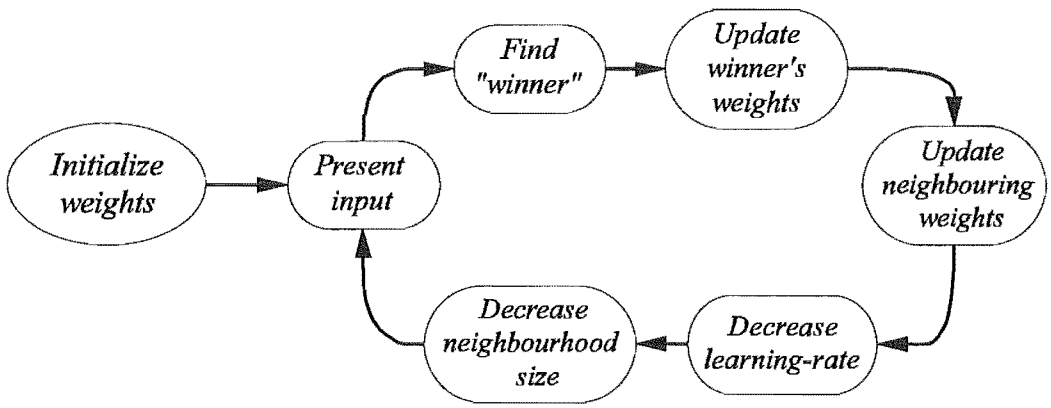


Figure 5.5 The SOFM training algorithm broken up into its major components.

5.4 CHOOSING THE SOFM PARAMETERS

The success of SOFM formation is critically dependent on how the two main parameters in the algorithm, the learning rate $\alpha(t)$ and the neighbourhood function $N_c(t)$, are selected. The accuracy of the map also depends on the number of iterations in the training cycle, as the learning process is stochastic in nature. Unfortunately, there is

no theoretical basis for the selection of these parameters other than by means of trial and error. However, Kohonen puts forward a number of "rules-of-thumb" to be used when selecting the values of the various parameters, based on observations made during computer simulations. In the following chapter a number of computer simulations are described that were performed in order to obtain realistic measures on the learning algorithm parameters with differing SOFM architectures and input dimensions.

The following are the the primary guidelines to selection of parameters as given by Kohonen [Kohonen 1988], [Kohonen 1990], [Kohonen 1995].

5.4.1 Initialization

The initial values for the $\mathbf{m}_i(0)$ can be arbitrary, including random, values. The only restriction is that they are different.

5.4.2 Number of training steps

Due to the stochastic nature of the learning process, the final accuracy of the trained system depends on the number of iterations or training steps, which must be reasonably large. The rule-of-thumb is that, for good statistical accuracy, the number of steps should be 500 times the number of neurons in the lattice. The dimension of the input vector $\mathbf{x}$ has no effect on the number of training steps and hence high input dimensions can be accommodated by the SOFM with relative ease. Anywhere from $10,000$ to $100,000$ training steps have been used by Kohonen but the actual number depends on the application and the nature and quantity of the input vectors. If only a small number of input vectors are available for training then they must be recycled in a random manner for the desired number of steps. Unfortunately, the large number of steps required for training is important for system accuracy and means that training is computationally expensive. Note, however, that the learning algorithm for each step is relatively inexpensive.

5.4.3 Learning rate

The monotonic decrease in $\alpha(t)$ $(0 < \alpha(t) < 1)$ can be linear, exponential, or inversely proportional to t. That is, the exact profile of $\alpha(t)$ is not important. What is important is that $\alpha(t)$ is large initially during the so called *ordering phase* of the map (of tuhe order of 1000 steps). Towards the end of the training $\alpha(t)$ should take on small values (e.g., $\alpha(t) \leq 0.01$) over a fairly long period of time (typically thousands of iterations). This final longer period is known as the *fine-adjustment phase.*

5.4.4 Neighbourhood functions

Special care is required in choosing the values of $N_c = N_c(t)$. Starting with a neighbourhood which is too small results in a map which will not be globally ordered and consists of small pockets of local ordering between which the ordering direction changes discontinuously. So, by starting with a large $N_c = N_c(0)$ and letting it shrink with time, global ordering is achieved during the crucial first stage of training. The radius can be more than half the radius of the network initially. The rule of thumb is, that during the first 1000 steps or so, when $\alpha = \alpha(t)$ is relatively large (the *ordering phase*), the radius of N_c can shrink to encompass the winning neuron and its immediate neighbours only. After that, during the *fine-adjustment phase*, N_c can remain as just the winning neuron and its immediate neighbours ($N_c = 1$).

The actual shape of the neighbourhood function N_c can consist of a square topological neighbourhood of varying size around the winning neuron c, as seen in Figure 5.6a or can also take other forms such as hexagonal as shown in Figure 5.6b.

The monotonic decrease of $N_c = N_c(t)$ can be achieved either through some piecewise linear or exponential function.

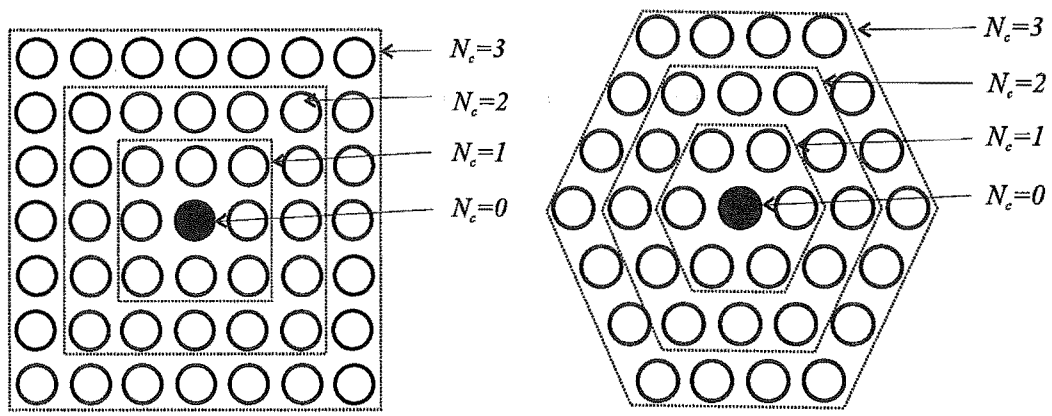


Figure 5.6 Different neighbourhood topographies (a) square and (b) hexagonal.

5.5 CALIBRATING THE SOFM

Once the input vectors have been repeatedly applied to the SOFM and the algorithm has converged, the feature map computed by the SOFM algorithm displays the important statistical characteristics of the input data. The aim of the SOFM is to store a large number of input vectors ($\mathbf{x}$) by finding a smaller set of prototypes ($\mathbf{w}_i$) in an attempt to provide a "good" approximation of the input space. The theoretical basis for this process can be found in *vector quantization theory* [Gray 1984].

The technique of vector quantization exploits the underlying structure of input vectors for the purpose of data compression. With vector quantization the input space is divided into regions and for each region a *reproduction vector* is defined. When a new input vector is presented to the quantizer, the region in which the vector lies is first determined and then the input vector is represented by the reproduction vector assigned to that region. Using the reproduction vector for storage or transmission in place of the original input vector results in considerable savings in storage or transmission bandwidth (albeit with some distortion). The collection of such reproduction vectors is called a *codebook* of the quantizer. In particular, a *Voronoi quantizer* is a vector quantizer with minimum encoding distortion. Each vector partitions the input space into *Voronoi cells* which contain all those points in the input space that are closest to each particular Voronoi vector (according to the nearest-neighbour rule based on the Euclidean metric [Gersho 1982]). The SOFM provides an "approximate" method of computing the Voronoi vectors (in a self-organising manner), which are represented by the weights $\mathbf{m}_i$ [Kohonen 1995] after training.

If the SOFM is to be used as a classifier, each weight vector must be assigned a "class label" based on the type of input vector it comes to represent after SOFM training. The assigning of class labels to all $\mathbf{m}_i$ of the trained SOFM is termed *calibration*. This is a supervised operation where input vectors of known class membership are presented to the map after training is complete and the "winning" neuron (best matching weight vector, $\mathbf{m}_c$) is said to be representative of that input class. In a form of *majority voting*, the final label assigned to each weight vector is that of the input class which most frequently causes the neuron to win. The input vectors used for calibration, the so-called *calibration vectors*, are usually a small subset of the input vectors which have been labelled according to the class they represent [Kohonen 1990].

5.6 USING THE SOFM FOR PATTERN CLASSIFICATION

As the SOFM is primarily intended to visualise the topological relationship of input vectors, it is not particularly suited to be used by itself for pattern classification. This is because the placement of the codebook vectors is based on an approximate technique as explained in the previous section. For pattern classification the requirement is to classify input vectors into a finite number of classes such that the average probability of misclassification is minimized [Haykin 1994]. The trained and calibrated SOFM thus should be followed by a supervised optimizing technique which attempts to minimize misclassification, in effect "fine-tuning" the weights of the SOFM.

Methods for "fine-tuning" the codebook vectors in order to minimize misclassification are described in the next sections.

5.6.1 Adding auxiliary class information

It is possible to 'embed' discrete symbolic information available on the identity of the input signal (e.g., class information) by making the input to the SOFM out of several parts [Kohonen 1989a] such as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_c \end{bmatrix} = \begin{bmatrix} \mathbf{x}_s \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{x}_c \end{bmatrix}, \tag{5.11}$$

where $\mathbf{x}$ is the input vector composed of the input signal information held in the vector $\mathbf{x}_s$ and the class information held in $\mathbf{x}_c$. The "class component" in the input vectors then causes a bias to the reproduction vectors whilst the SOFM is training. The simplest form of "class component" would be to introduce a unit vector $[0, 0, \cdots, 0, 1, 0, \cdots, 0]^T$ where the 1 is in the same position for input signals of the same class. The idea is to have the class-component weighted such that it does not dominate the training but influences the clustering. Then, if during recognition of input information the class information is missing, the closest matching codebook vector is selected solely on the basis of the input signal part.

The same reasoning has been used by Kohonen when constructing semantic maps [Kohonen 1989b], [Kohonen 1990]. In forming semantic maps for linguistic representations, linguistic symbols are assigned vectorial representations and the mutual distances between vectors of different symbols cannot be said to have any relationship with the observable characteristics of the corresponding items. This makes it difficult to display 'logical similarity' between items and to display them topographically. By presenting the symbols *in context* during learning, the inherent similarities and differences can be made apparent. In this way, the input vector for the SOFM is made up of two parts as in the previous section; a vector $\mathbf{x}_s$ representing the symbolic item and a vector $\mathbf{x}_c$ representing contextual information relevant to the symbol part. Both parts are then presented during training and topographic maps of the symbolic items can be formed, helped along by the contextual information. Then during recognition, if the contextual part is weak or missing altogether, the codebook vector selected will be selected due to the symbolic part of the vector alone. In this way contextual information has been used to further "coax" the map along when forming clusters with input vectors.

5.6.2 Using learning vector quantization

The second method to achieve improved classification performance is to follow the training and calibration of the SOFM with a supervised learning technique. *Learning vector quantization* (LVQ) is one such technique, developed by Kohonen to "fine-tune" the weights of the trained SOFM in a supervised manner [Kohonen 1988], [Kohonen

et al. 1988] and the one employed in the system described in Chapters 6 and 9. LVQ is a supervised technique that uses class information to move the Voronoi vectors slightly, in such a way as to improve the quality of the classifier decision regions. Kohonen put forward three versions of the LVQ algorithm: LVQ1, LVQ2 and LVQ3, each of which is described below.

5.6.2.1 LVQ1

The initial values of the $\mathbf{m}_i$ before the fine-tuning with LVQ can start must be such that the $\mathbf{m}_i$ represent the overall statistical density function $p(\mathbf{x})$ of the input. The SOFM is suited to achieve this. The weight vectors $\mathbf{m}_i$ of the trained SOFM are then assigned a class label following the calibration process. Once an input $\mathbf{x}$ is applied to the SOFM, the classification of $\mathbf{x}$ is based on the class label of the nearest $\mathbf{m}_i$ in a Euclidean sense.

The LVQ algorithm works by "pulling" the weight vectors (or the Voronoi vectors) away from the decision surfaces to demarcate the class borders more accurately [Kohonen 1988],[Kohonen 1989a],[Kohonen *et al.* 1988].

For each input vector for which the classification is already known, the $\mathbf{m}_i = \mathbf{m}_i(t)$ are updated as follows

$$\begin{aligned}
\mathbf{m}_i(t+1) &= \mathbf{m}_i(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)], && \text{if } i = c \text{ and } \mathbf{x} \text{ is classified correctly,} \\
\mathbf{m}_i(t+1) &= \mathbf{m}_i(t) - \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)], && \text{if } i = c \text{ and } \mathbf{x} \text{ is classified incorrectly,} \\
\mathbf{m}_i(t+1) &= \mathbf{m}_i(t), && \text{for } i \neq c,
\end{aligned} \tag{5.12}$$

where $\alpha(t)$ is the usual learning rate factor ($0 < \alpha(t) < 1$), which decreases monotonically with time as for the previous algorithm. Being a fine-tuning algorithm the learning rate should not be too large to begin with (suggested values are of $\alpha(0) = 0.01$ or 0.02) and should decrease to zero after around 100,000 iterations of the algorithm [Kohonen 1990]. The algorithm described here is known as type one LVQ, or LVQ1, and a rigorous mathematical discussion of the algorithm can be found in LaVigna [1989].

5.6.2.2 LVQ2

LVQ2 involves a slight modification to the LVQ1 algorithm. We can consider two weight vectors, $\mathbf{m}_i$ and $\mathbf{m}_j$, which belong to different classes (C_i and C_j) and are closest neighbours in vector space. Initially these vectors are placed in the wrong position in vector space. The discrimination surface is always defined as the mid-plane between $\mathbf{m}_i$ and $\mathbf{m}_j$ (if $\mathbf{x}$ is closest to $\mathbf{m}_i$ then $\mathbf{m}_i$ is the nearest neighbour and so on), this

is in most cases incorrectly defined. Kohonen suggests defining a symmetric window of non-zero width around the midplane and making corrections to both $\mathbf{m}_i$ and $\mathbf{m}_j$ if $\mathbf{x}$ falls into the window on the wrong side of the midplane. Using this criterion, the changes to the weight vectors are made as follows

$$\begin{aligned} \mathbf{m}_i(t+1) &= \mathbf{m}_i(t) - \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)], \\ \mathbf{m}_j(t+1) &= \mathbf{m}_j(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_j(t)], \end{aligned} \tag{5.13}$$

where C_i is the nearest class, but $\mathbf{x}$ belongs to $C_j \neq C_i$ and $\mathbf{x}$ falls into the window, and

$$\mathbf{m}_k(t+1) = \mathbf{m}_k(t), \quad \text{for } k \notin \{i, j\}. \tag{5.14}$$

With corrections taking place as described in Equation 5.13 and Equation 5.14 to *both* $\mathbf{m}_i$ and $\mathbf{m}_j$ it can be seen that, on average, the corrections will result in the midplane between the two codebook vectors shifting towards the point where membership of the two classes is equally likely [Kohonen 1990]. Figure 5.7 depicts one such example.

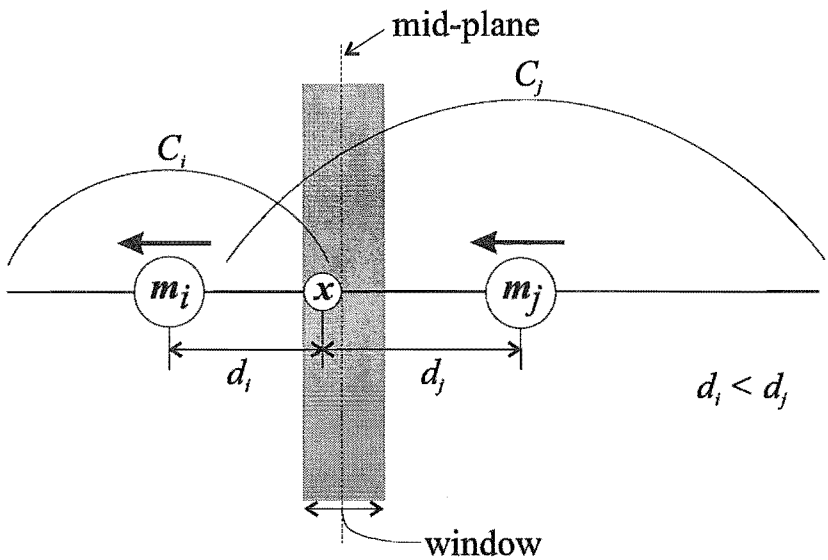


Figure 5.7 The operation of the window in LVQ2. $\mathbf{x}$ belongs to class C_j but $\mathbf{m}_i$ is the nearest code-book vector (belonging to class C_i). LVQ2 works by moving $\mathbf{m}_i$ *away* from $\mathbf{x}$ and $\mathbf{m}_j$ *towards* $\mathbf{x}$; hence moving the mid-plane closer to the point where membership of the two classes is equally likely.

The window width is defined by the relative window width w and is expressed as a fraction of the total distance between $\mathbf{m}_i$ and $\mathbf{m}_j$ (i.e., $d_i + d_j$). The optimal w must be determined experimentally, although a width of 10 to 20% of the distance between $\mathbf{m}_i$ and $\mathbf{m}_j$ is put forward as a good value for a relatively small number of training samples. For high-dimensional input space, a reasonable definition of the window is in

terms of a constant ratio between the distances (d_i and d_j) respectively of $\mathbf{x}$ from $\mathbf{m}_i$ and $\mathbf{m}_j$ [Kohonen *et al.* 1988]. The input $\mathbf{x}$ is then said to fall within the window if

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s, \quad \text{where } s = \frac{1-w}{1+w}. \tag{5.15}$$

Once more, this algorithm should be applied with a small learning rate $\alpha(t)$ which decreases monotonically with time as the training progresses.

5.6.2.3 LVQ3

The problem with LVQ2 is that the algorithm is based on the idea of *differentially* shifting the decision borders towards the boundary between the two class types, with no attention being paid to what might happen to the eventual location of the $\mathbf{m}_i$ as the process continues. In LVQ2, the correction made to the correct class codebook vector $\mathbf{m}_j$ is of a larger magnitude to that on the wrong class codebook vector $\mathbf{m}_i$, resulting in monotonically decreasing distances between $\mathbf{m}_i$ and $\mathbf{m}_j$.

The improved algorithm which takes these drawbacks into account is called the LVQ3 algorithm and consists of Equation 5.13 where $\mathbf{m}_i$ and $\mathbf{m}_j$ are the two closest codebook vectors to $\mathbf{x}$; $\mathbf{x}$ and $\mathbf{m}_j$ belong to the same class, while $\mathbf{x}$ and $\mathbf{m}_i$ belong to different classes; at the same time $\mathbf{x}$ must fall into the window. Furthermore if $\mathbf{x}$, $\mathbf{m}_i$ and $\mathbf{m}_j$ belong to the same class then

$$\mathbf{m}_k(t+1) = \mathbf{m}_k(t) + \epsilon\alpha(t)[\mathbf{x}(t) - \mathbf{m}_k(t)], \quad \text{for } k \in \{i, j\}. \tag{5.16}$$

The value of ϵ must be experimentally obtained; the optimal value seems to depend on the size of the window, being smaller for narrower windows. This algorithm does not alter the optimal placement of the $\mathbf{m}_i$ with continued learning [Kohonen *et al.* 1988].

Notice that in the progression from LVQ1 to LVQ2 and LVQ3, the number of codebook vectors being altered at any one time was one for LVQ1 and became two for LVQ2 and LVQ3.

5.7 SUMMARY

The SOFM developed by Teuvo Kohonen is a self-organising single-layer ANN whose weight vectors become specifically tuned to the various features in the input patterns presented to it. It is primarily developed as a means of visualising similarities in certain features of the inputs through the 1D or 2D arrangement of the neurons in the

ANN. However, the SOFM becomes a useful first stage in the development of a pattern recognition system when followed by LVQ techniques such as LVQ1, LVQ2 or LVQ3.

Two factors which make the SOFM particularly desirable are the following. Firstly, the fact that the SOFM is self-organising means that large quantities of unlabelled data can be used in the training of the ANN. This is particularly valuable if labelled data is not readily available or available only in small quantities. Following the SOFM training with calibration and LVQ does require a labelled data set, however this set can be small when compared to the SOFM training data set. Secondly, although training the SOFM requires the data to be cycled a large number of times through the algorithm making the process rather lengthy initially, the computational 'short-cut' developed by Kohonen makes the process very 'light' computationally. Once the SOFM is trained and calibrated, it becomes fast to implement operationally as it consists of a single layer of neurons (and weight vectors) and requires only distance computations.

Chapter 6

THE SELF-ORGANISING FEATURE MAP: COMPUTER SIMULATIONS

6.1 INTRODUCTION

From Chapter 5 it can be seen that there are a number of parameters important to the training of a SOFM. The purpose of the work presented in this chapter was to obtain an understanding of the effect of the various parameters on the training of a SOFM and on its eventual performance. To this end, a number of computer simulations were carried out for various SOFM architectures and input characteristics. The major parameters considered are (a) the learning rate α and how it decreases with time and (b) the neighbourhood size N_c and how it decreases with time during training, (c) the weighting applied to the weight vectors within the neighbourhood of the winning neuron, (d) the number of times the data set must be presented to the SOFM for adequate training and (e) the optimum size of the SOFM. For each simulation, a number of these parameters are varied whilst observing the performance of the trained SOFM on novel input data. Each simulation utilises a different input data set with different characteristics. In all of the simulations, the initial weight values in each case are small random numbers and are not changed between tests for a given simulation.

In summary, these simulations were carried out in order to (a) confirm some rules-of-thumb as suggested by Kohonen [1990] and Kohonen [1995], (b) obtain working values for the most important parameters needed for training the SOFM and fine-tuning with LVQ and (c) to assess the robustness of the above-mentioned parameters on different network sizes, input quantities and input dimensions.

6.2 PRELIMINARIES TO COMPUTER SIMULATIONS

The parameters to be varied in the computer simulations are described next, where in each case the values to be used are discussed.

6.2.1 Learning rate and neighbourhood size decay

As stated in Section 5.4, both the learning rate and the neighbourhood size should start off with large values and decrease gradually with time. It is important that they are both reasonably large during the ordering phase and that both reach their minimum, and maintain it, during the longer fine-adjustment phase. In particular, the learning rate $\alpha = \alpha(t)$ should start off with a value at or near unity and decrease monotonically during the ordering phase, finally reaching a minimum value $\alpha_{\min}$ which must be maintained during the fine-adjustment phase. Kohonen suggests a value for $\alpha_{\min}$ of the order of 0.001 [Kohonen 1990]. The neighbourhood size $N_c = N_c(t)$ should start with a large radius which can be as large as half the widest dimension of the SOFM (or larger). This too should decrease monotonically until some minimum radius $N_{\min}$ is reached towards the end of the ordering phase. During the fine-adjustment phase, N_c should stay at $N_{\min}$. $N_{\min}$ is typically set to '1' so that only the winning neuron and its immediate nearest neighbouring neurons are influenced.

An important point during training is where the ordering phase finishes and the fine-adjustment phase starts. Kohonen suggests that the ordering phase should take place within the first thousand or so iterations of the input data [Kohonen 1990]. If each iteration of the data is represented by the discrete time variable k, then the total number of iterations of the input data during the training process can be represented by $k_{\max}$. If the training process is set up such that the total number of iterations of the data to be undertaken is known (as is generally the case), then the change-over between ordering and fine-adjustment phases can be defined to occur at a given fraction ω of the total number of iterations.

Two types of monotonically decreasing functions are considered here, a *piecewise linear* decreasing function and a *piecewise exponential* decreasing function.

The piecewise linear function is given by:

$$\begin{aligned} \alpha(k) &= \alpha_o(1 - \tfrac{k}{\omega k_{\max}}) + \alpha_{\min}(\tfrac{k}{\omega k_{\max}}), && \text{if } k \leq \omega k_{\max}, \\ \alpha(k) &= \alpha_{\min}, && \text{if } k > \omega k_{\max}, \end{aligned} \tag{6.1}$$

$$\begin{aligned} N_c(k) &= N_o(1 - \tfrac{k}{\omega k_{\max}}) + N_{\min}(\tfrac{k}{\omega k_{\max}}), && \text{if } k \leq \omega k_{\max}, \\ N_c(k) &= N_{\min}, && \text{if } k > \omega k_{\max}. \end{aligned} \tag{6.2}$$

The piecewise exponential decay function is given by:

$$\begin{aligned} \alpha(k) &= \alpha_o^{(1-\frac{k}{\omega k_{\max}})} \alpha_{\min}^{\frac{k}{\omega k_{\max}}}, && \text{if } k \leq \omega k_{\max}, \\ \alpha(k) &= \alpha_{\min}, && \text{if } k > \omega k_{\max}, \end{aligned} \tag{6.3}$$

$$\begin{aligned} N_c(k) &= N_o^{(1-\frac{k}{\omega k_{\max}})} N_{\min}^{\frac{k}{\omega k_{\max}}}, \quad \text{if } k \leq \omega k_{\max}, \\ N_c(k) &= N_{\min}, \qquad\qquad\qquad\quad \text{if } k > \omega k_{\max}. \end{aligned} \tag{6.4}$$

where, in both cases, α_o and N_o give the initial learning rate and neighbourhood radius respectively, whereas $\alpha_{\min}$ and $N_{\min}$ give the respective minimum values. Since the neighbourhood radius is defined by a positive real integer, Equation 6.2 and Equation 6.4 are implemented so the next greater integer than $N_c(k)$ is used.

6.2.2 Neighbourhood taper

So far, the updates performed on the weight vectors of the winning neuron and its neighbours have been assumed to be uniform within the neighbourhood. However, it is also possible to apply the updates in a non-uniform manner described henceforth as the *neighbourhood taper*. This means that it is possible to alter the update of each weight vector within the winning neighbourhood by a different amount depending on the Euclidean distance d_i between the weight vector $\mathbf{m}_i$ and the winning weight vector $\mathbf{m}_c$, given by

$$d_i = \| \mathbf{m}_i - \mathbf{m}_c \|, \quad \text{for } i \in N_c. \tag{6.5}$$

Figure 6.1 depicts the three tapering schemes used in the computer simulations to follow.

1. Uniform taper: Figure 6.1a shows a uniform weighting of the updates across the neighbourhood. The taper function here is simply

$$h_i = 1, \quad \text{for } i \in N_c, \tag{6.6}$$

   and is the standard for the SOFM algorithm as described by Kohonen [1990].

2. Gaussian taper: Figure 6.1b shows Gaussian tapering of the neurons based on the distance of each neuron in the neighbourhood from the winning neuron c. The taper function is given by

$$h_i = \exp \frac{-d_i^2}{(N_c+1)^2}, \quad \text{for } i \in N_c. \tag{6.7}$$

3. Quadratic taper: Figure 6.1c shows a scheme where the taper is inversely related to the square of the distance between the winning neuron and each of its

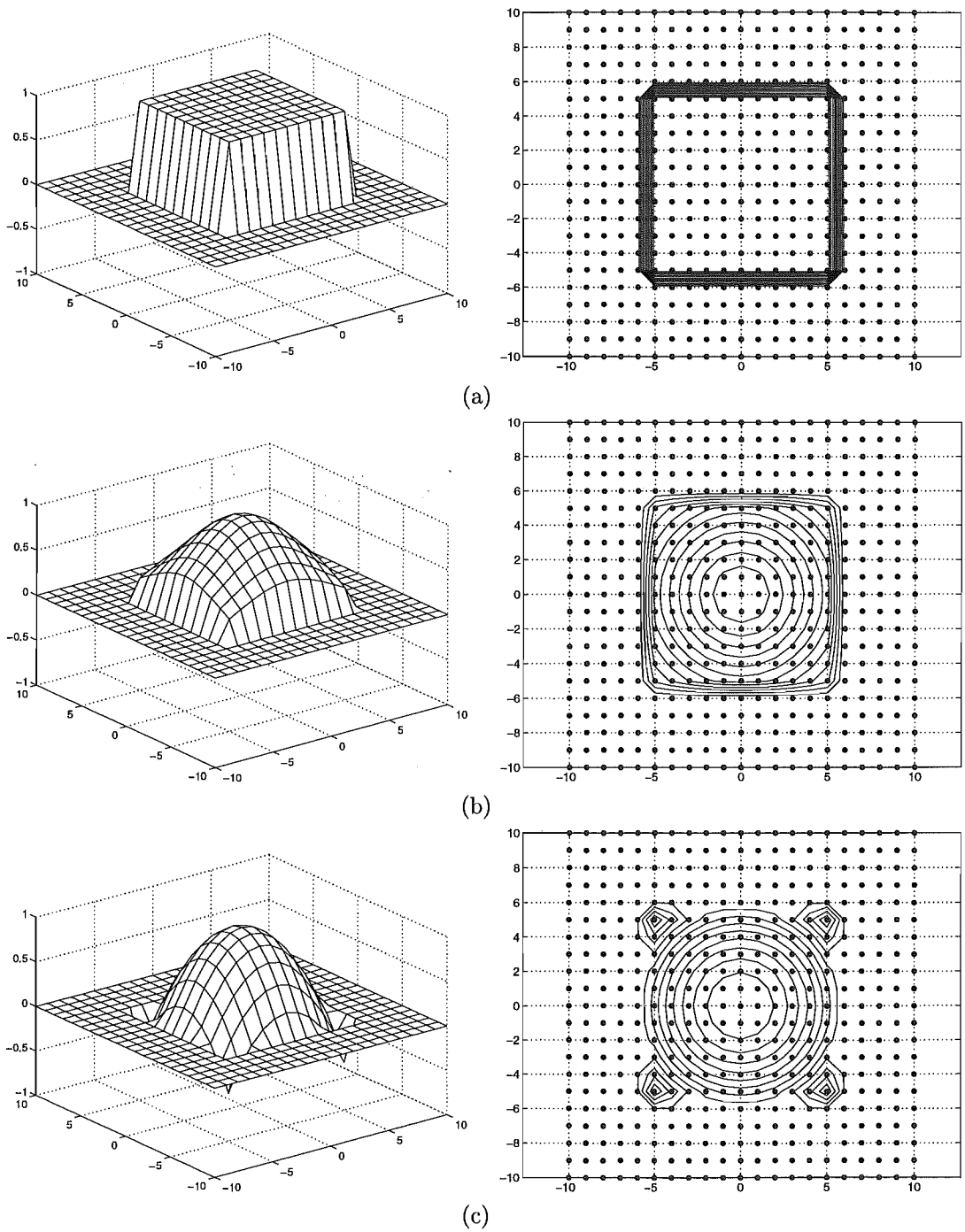


Figure 6.1 Different neighbourhood tapering functions for a neighbourhood of $N_c = 5$. (a) Uniform taper, (b) Gaussian taper and (c) quadratic taper of the neighbourhood.

neighbours, given by

$$h_i = 1 - 1.05\frac{d_i^2}{(N_c+1)^2}, \quad \text{for } i \in N_c. \tag{6.8}$$

This tapering scheme was proposed by Roberts and Tarassenko [1992] and is devised such that it has points at the extremities of the neighbourhood where slight inhibitory action is performed on the weight vectors. This effectively "pushes away" those weight vectors by a small amount in the direction opposite to that of the winning neuron.

The taper function h_i for each of the schemes is then used to modify the weight updating step of the SOFM training algorithm, given by Equation 5.10. The updated equations then become

$$\begin{array}{rcll} \mathbf{m}_i(t+1) & = & \mathbf{m}_i(t) + \alpha(t)h_i[\mathbf{x}(t) - \mathbf{m}_i(t)], & \text{if } i \in N_c(t), \\ \mathbf{m}_i(t+1) & = & \mathbf{m}_i(t), & \text{if } i \notin N_c(t). \end{array} \tag{6.9}$$

6.2.3 A measure of network stability during training

As the SOFM training algorithm requires a large number of iterations (often hundreds of thousands) of the input data, a useful measure for the map is that of stability during training. A measure of how "settled" the weights become during training can be obtained by measuring the mean Euclidean distance $\delta(k)$ between the weight vectors, from one step to the next, of the SOFM at discrete epochs during the training period, that is,

$$\delta(k) = \frac{1}{N}\sum_{i=1}^{N} \parallel \mathbf{m}_i(k) - \mathbf{m}_i(k-1) \parallel, \tag{6.10}$$

where $\mathbf{m}_i(0)$ gives the initial weight settings. The effect of the various parameters of the SOFM such as the learning rate, neighbourhood taper and SOFM size, for example, can be observing by observing the "stability" measure for the SOFM at various points during the training process. In this context, an "unstable" map is taken to be a map that has large $\delta(k)$ values during the training process and does not settle to a minimum value during the fine-adjustment phase (i.e., $\delta(k)$ continues to oscillate about some mean value). This can be interpreted as being a result of one or more of the SOFM parameters causing the SOFM to train poorly, the net result being that when the training process is complete, the SOFM weights might not be optimally placed in the input space. In this way, they may not best represent the underlying characteristics of the input patterns (e.g., this may reflect an insufficient number of neurons). In

contrast, a "stable" map is taken to be a map that has a mean Euclidean distance that converges to a minimum during the fine-adjustment phase in a "stable manner". This is interpreted as resulting in weights which are optimally placed to represent the input characteristics once training is complete.

The change in Euclidean distance $\delta(k)$ converges to some non-zero minimum due to the fact that $\alpha_{\min}$ is non-zero and $N_{\min}$ is such that more than one weight vector will be altered with each presentation of the data. The actual value of the minimum about which $\delta(k)$ 'oscillates' is determined in part by the SOFM size.

6.2.4 Measures of performance

As stated in Section 5.6, the SOFM on its own is not necessarily suitable as a pattern classifier but, when followed by a 'fine-tuning' mechanism such as LVQ, it can become an important first phase in a pattern classification system.

A performance measure gives an indication of how good (or bad) a system performs. Once a SOFM is trained, calibrated and followed by LVQ, a measure of the system's performance is required. The two measures adopted to assess performance in this case are system *sensitivity* and *selectivity* and are defined [Wilson *et al.* 1991] [Webber *et al.* 1994] as

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\% \tag{6.11}$$

and

$$\text{Selectivity} = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\%, \tag{6.12}$$

where TP (true positives) is the number of true events *correctly* detected, FP (false positives) the number of false events *incorrectly* classified as true, and FN (false negatives) the number of true events which were missed. These measures can also be written as

$$\text{Sensitivity} = \frac{\text{correct detections}}{\text{total number of true events}} \times 100\% \tag{6.13}$$

and

$$\text{Selectivity} = \frac{\text{correct detections}}{\text{total number of detections}} \times 100\%. \tag{6.14}$$

Ideally, both sensitivity and selectivity would be 100% but, in practice, this is often not possible, in which case both must be as close as possible to 100%. It is, however, always possible to improve the performance of one measure at the cost of

decreasing the performance of the other. The choice of which is the more important of the two, and hence should be improved over the other, depends on the application of the pattern classifier. In general, a balance between sensitivity and selectivity is the most appropriate and useful goal when assessing the performance of the SOFMs in the simulations to follow.

6.3 COMPUTER SIMULATIONS

6.3.1 Simulation 1: Pattern classification I

This simulation deals with the problem of classifying a pair of overlapping 2D Gaussian processes labeled C_a and C_b. Class C_a has a mean of $(-1, -1)$ and a variance of 1.0 and class C_b has a mean of $(0.5, 0.5)$ and a variance of 0.5. Figure 6.2 shows a scatter-plot of the training set which consisted of 1000 points in the 2D input space. Both classes were equally represented in the training set.

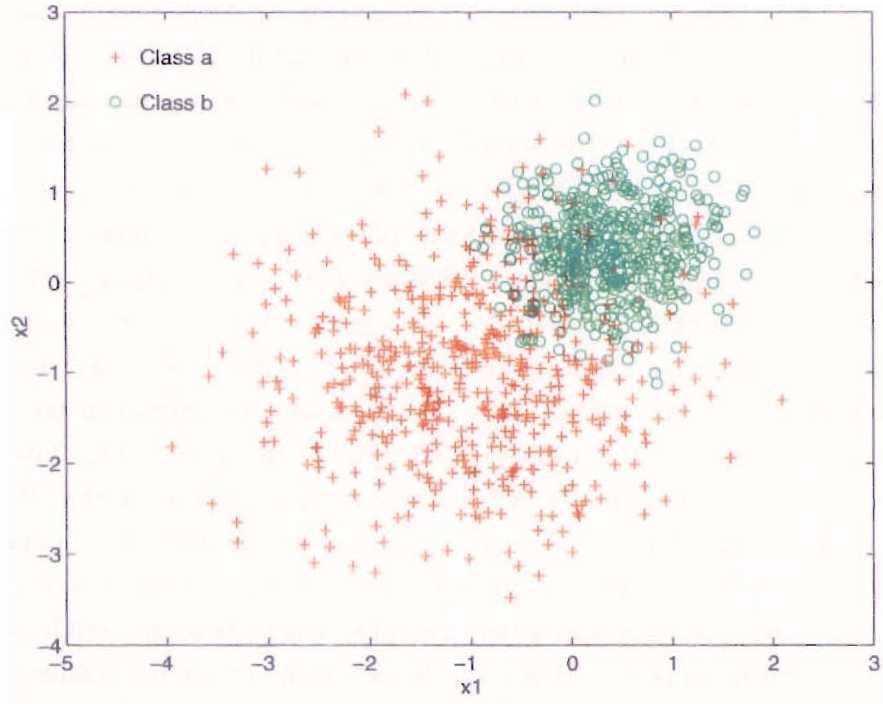


Figure 6.2 A scatter plot of two clusters of 2D Gaussian processes, C_a and C_b. Both classes are equally represented.

The training set was used to repeatedly train a 2D lattice of neurons of fixed size (8×8) but with varying SOFM training parameters including:

1. Using linear decay and exponential decay for the learning rate and neighbourhood size.

2. Setting the maximum number of iterations of the input data to 200 and 500 times the number of neurons.

3. Setting the minimum learning rate $\alpha_{\min}$ to 0.01 and 0.001.

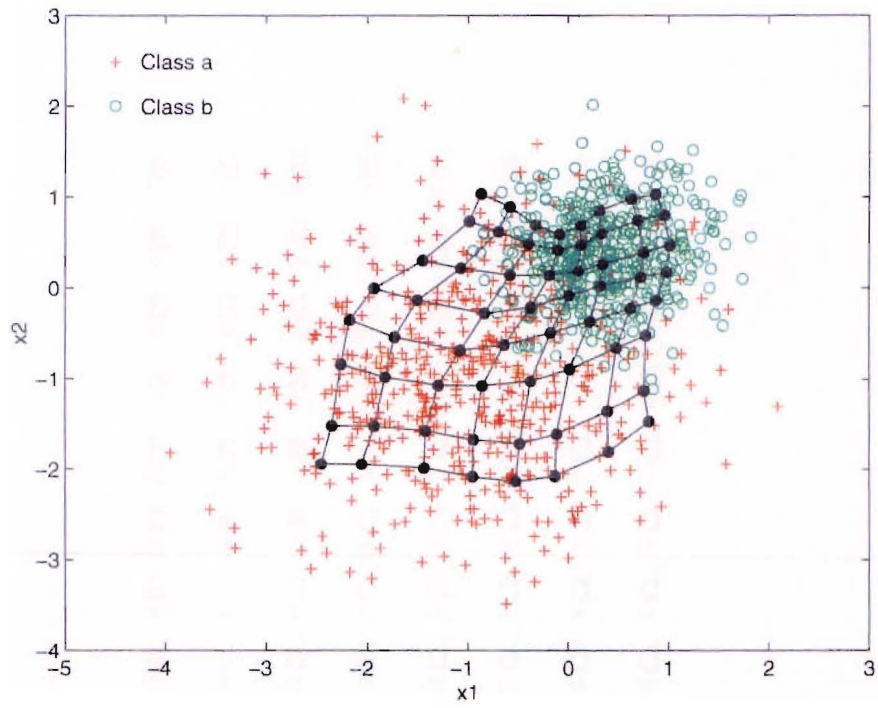
4. Setting the neighbourhood taper functions to uniform, Gaussian and quadratic.

5. Performing straight SOFM training, SOFM training followed by LVQ1 and SOFM training followed by LVQ2.

For this simulation and the simulations to follow an SOFM of fixed size was trained using all the above-mentioned combinations of training parameters. For every SOFM trained, the measures of sensitivity and selectivity (as defined in Section 6.2.4) were calculated for each class after straight SOFM and after fine-tuning with LVQ1 and LVQ2. Once all the tests were performed each parameter was identified in turn and the mean performance was calculated (i.e., the mean of sensitivity and selectivity) over all classes and for all combinations of parameters excepting the parameter being scrutinized. In this way a single measure of performance is available for each particular setting of the parameter under scrutiny and is given in tabular form for each simulation. The actual performance measures calculated for each test for this and the following simulations can be found in Appendix B. The performance was measured in each case using novel data drawn from the same distributions as the training data.

As the input space is 2D, it is possible to plot the trajectory of the weight vectors in the input space. Figure 6.3a depicts a 2D $[8 \times 8]$ SOFM after training. The input data was presented to the SOFM 12,800 times with $\alpha_{\min} = 0.01$, a quadratic neighbourhood taper function and using exponential decay for the learning rate and neighbourhood size. The weights of the trained SOFM have become positioned in the input space such that they represent the underlying densities of the inputs. The lines connecting the weight vectors in Figure 6.3a indicate neighbouring neurons in the 2D lattice that makes up the SOFM. Figure 6.3b depicts the *calibrated* SOFM where a class label has been assigned to each neuron according to the maximum voting criteria explained in Section 5.5. Each neuron is also assigned a value which gives the *confidence level* with which each neuron represents that class, based on the maximum voting principle. In this way, a 100% confidence level means that a neuron responded in all cases to a particular class *during calibration*, whereas a 50% confidence level means that a neuron responded equally to both classes during calibration. This confidence level cannot be taken to be a probability as it stands, as it takes no account of the *frequency* with which each neuron wins during calibration. This can be done using Bayesian probabilities and is explained in further detail in Chapter 9.

It is important to note that topological ordering is observed, both in the plots of the weight vectors in the input space and in the assigning of labels during calibration. Labels form 'neat' clusters with easily distinguishable boundaries. It is at the boundaries



(a)

a 100	a 100	a 100	a 100	a 100	a 100	a 100	a 100
a 77	a 100	a 100	a 100	a 100	a 100	a 100	a 100
b 66	b 57	a 87	a 100	a 100	a 100	a 100	a 100
b 84	b 92	b 86	a 62	a 82	a 100	a 100	a 100
b 95	b 100	b 100	b 84	b 52	a 91	a 100	a 100
b 96	b 100	b 96	b 100	b 85	a 60	a 83	a 100
b 91	b 100	b 100	b 83	b 95	b 68	a 66	a 100
b 100	b 90	b 95	b 89	b 100	b 80	b 66	a 100

(b)

Figure 6.3 A trained [8 × 8] SOFM for simulation 1. The input data was presented to the SOFM 12,800 times with $\alpha_{\min} = 0.01$, a quadratic neighbourhood taper function and exponential decay for the learning rate and neighbourhood size. (a) The position of the weight vectors in relation to the input vectors after SOFM training and (b) the calibrated SOFM after straight SOFM training. The value under each class label indicates the confidence level with which that neuron represents its assigned class.

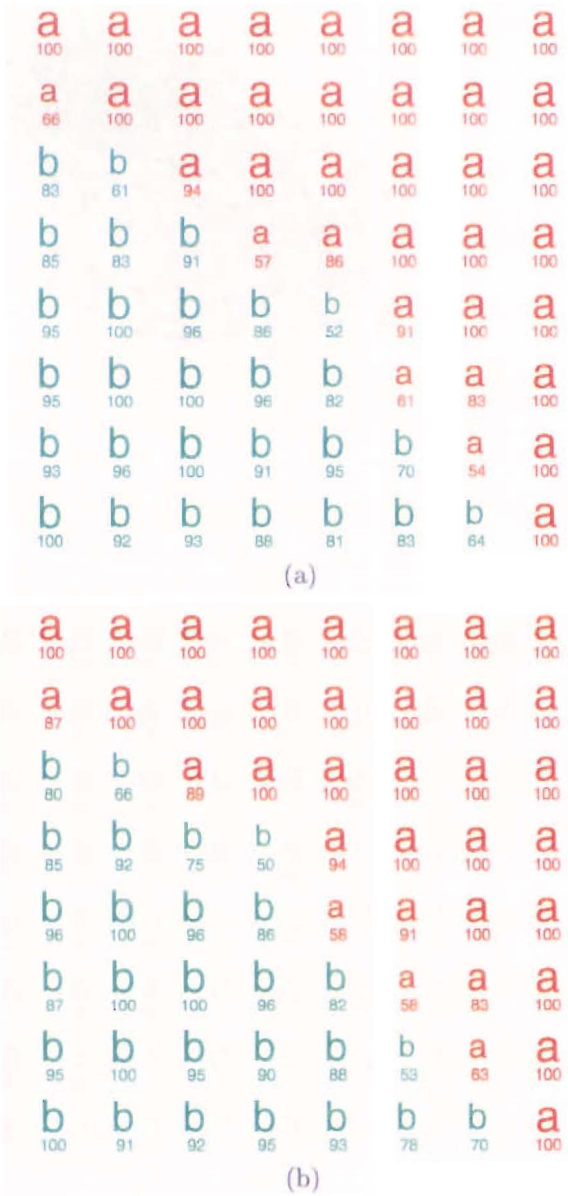


Figure 6.4 The trained [8 × 8] SOFM for simulation 1. The calibrated SOFM after (a) LVQ1 and (b) LVQ2. The value under each class label indicates the confidence level with which that neuron represents its assigned class.

that the confidence levels assigned to neurons are at their weakest, due to overlapping of the input distributions. Another important point is that the SOFM has assigned equal numbers of neurons for each class since both classes were equally represented in the training set. When a particular class is under-represented in the input set, it will also be under-represented in the SOFM (i.e., fewer weight vectors will be assigned to represent that class of input data) [Kohonen 1990].

A map trained with just the SOFM algorithm cannot be reliably used as a classifier because the placement of the weight vectors during training has not be performed in such a way as to minimize misclassification in any way. The trained (and calibrated) SOFM should be followed by LVQ in order to be used as a pattern classifier. Figure 6.4a depicts the same SOFM as in Figure 6.3 which has been re-calibrated following the operation of the LVQ1 algorithm on it. Figure 6.4b shows the same SOFM after LVQ2. During the process of LVQ, the position of the weight vectors in the input space have been 'fine-tuned' and this is reflected in the changed confidence levels of neurons in the SOFM, especially around the boundary between the two clusters of data. As the weight vectors are 'pulled away' from the boundaries, the confidence levels of neurons at the boundaries strengthen towards one class or the other. For example, this can be seen in the changed confidence levels of the two neurons in the second and third rows, first column, of Figure 6.3b and Figure 6.4b, before and after LVQ2 respectively.

Parameter	**Mean performance (%)**		
	Exponential	Linear	
Decay	90.8	90.6	-
	200 ×	500 ×	
Iterations	90.8	90.6	-
	0.001	0.01	
$\alpha_{\min}$	90.7	90.6	-
	Gaussian	Uniform	Quadratic
N_c taper	90.7	89.9	89.7
	LVQ2	SOFM	LVQ1
Algorithm	90.8	90.7	90.6

Table 6.1 The mean performance (i.e., mean of sensitivity and selectivity) of the [8 × 8] SOFM for simulation 1 for each value assigned to the SOFM parameters.

The values given in Table 6.1 show that there is very little variation in the average performance over the range of parameters used. Exponential decay offers a very slight advantage over linear decay (0.2%), as does $\alpha_{\min} = 0.001$ over $\alpha_{\min} = 0.01$ (0.1%). Of more significance is the fact that iterating 200× the size of the SOFM results in almost identical performance as iterating 500× the size of the SOFM (0.2% better in fact). Fine-tuning using LVQ2 results in a slight advantage over straight SOFM training (0.1%) – LVQ1 performing slightly worse than both other methods. The largest change in performance for this simulation was due to using the different neighbourhood taper

functions. Using the Gaussian taper results in a 0.8% increase in performance over using the uniform taper and a 1% increase over quadratic taper.

Overall, there is little difference in performance between using one parameter over another for this simulation. The fact that fine-tuning with LVQ after straight SOFM training results in little or no increase in performance could be due to the already well defined nature of the input classes, meaning that LVQ could offer no additional 'sharpening' of the class boundaries. Iterating only 200× the size of the SOFM offers a distinct advantage when compared to iterating 500× in the form of reduced training time. For this simulation 12,800 inputs, drawn from the 1000 data points, were presented to the SOFM (for 200×) meaning that during the crucial first 10% or so of the training (i.e., the ordering phase) the entire training set had been presented at least once. The original data for each test performed can be found in Appendix B.

On the basis of the results presented above (Table 6.1), the best parameters to use for this simulation would be those shown in Table 6.2. Using these parameters results in the performance shown in Table 6.3. However, at least in this case, it is clear that the performance is not critically dependent on any of the parameters.

Parameters	Value
Decay	exponential
Iterations	200 ×
$\alpha_{\min}$	0.001
N_c taper	Gaussian
Algorithm	SOFM followed by LVQ2

Table 6.2 The best performing parameters to use for training the [8 × 8] SOFM of simulation 1.

	Sensitivity (%)	Selectivity (%)	Average (%)
SOFM	90.9	90.9	90.9
LVQ1	90.9	90.9	90.9
LVQ2	91.5	91.5	91.5

Table 6.3 The performance of the [8 × 8] SOFM for simulation 1 after iterating 12,800 times, using exponential decay, $\alpha_{\min} = 0.001$ and Gaussian neighbourhood taper.

Figure 6.5 shows the mean change in Euclidean distance $\delta(k)$ between the weight vectors of the SOFM calculated at successive epochs during the training process for simulation 1. This is repeated for SOFMs of varying sizes. The ordering phase and fine-adjustment phase can be clearly distinguished in the curves, where during the ordering phase the weights are more 'unsettled' and during the fine-adjustment phase they become more 'settled'. As the number of neurons in the SOFM decreases (and hence so do the number of weight vectors) the ordering phase becomes more 'unsettled'.

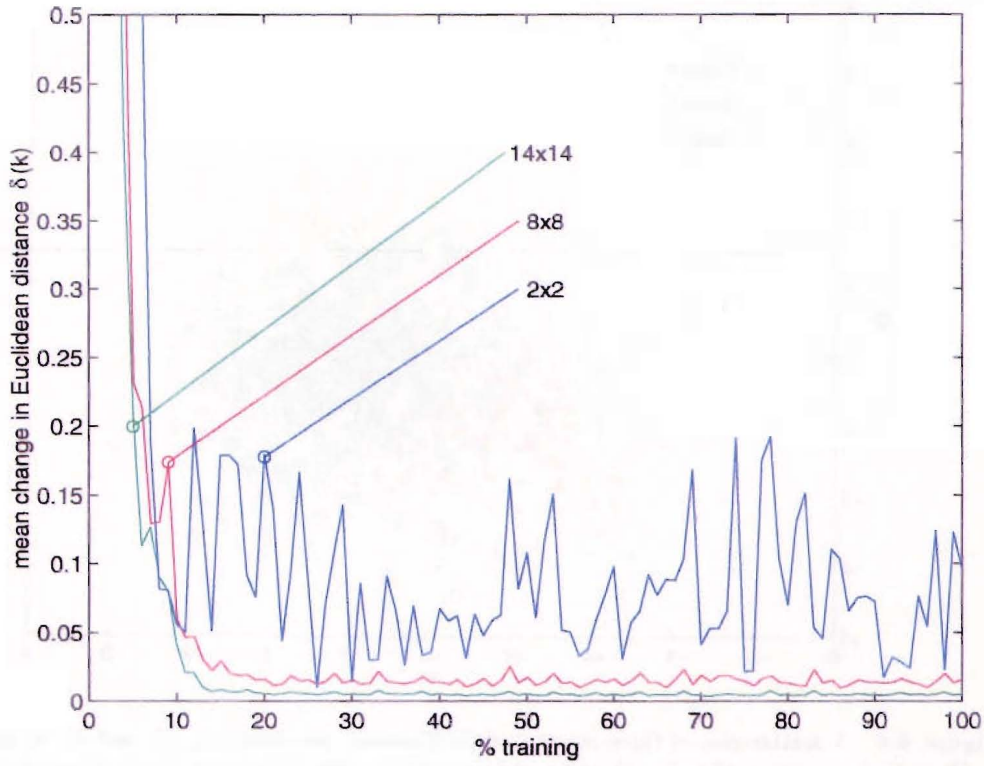


Figure 6.5 The mean change in Euclidean distance between weight vectors during training for different sized SOFMs for simulation 1. The ordering phase occurs within the first 15% of the training process and is characterised by large changes in Euclidean distances. The fine-adjustment phase is characterised by smaller changes in the distances. As the SOFM size decreases the mean changes increase and the SOFM becomes more 'unsettled'.

6.3.2 Simulation 2: Pattern classification II

The classification problem of the previous simulation was extended to a 3 class problem of differing overlapping 2D Gaussian distributions. Each class, C_a, C_b and C_c, was represented according to the ratio $2:7:1$ for $C_a:C_b:C_c$ and there were 1000 input vectors in total. Class C_a had a mean of $(-2,-2)$ and a variance of 1.0, class C_b a mean $(-1,-1)$ and variance 1.1 and class C_c a mean $(3,3)$ and variance 1.1, as depicted in the scatter plot of Figure 6.6.

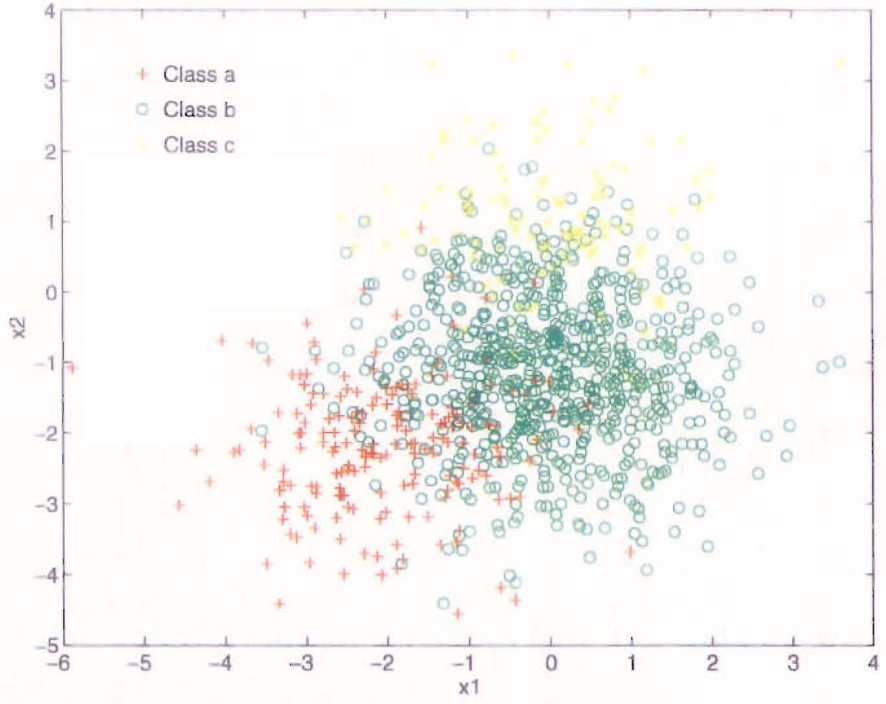
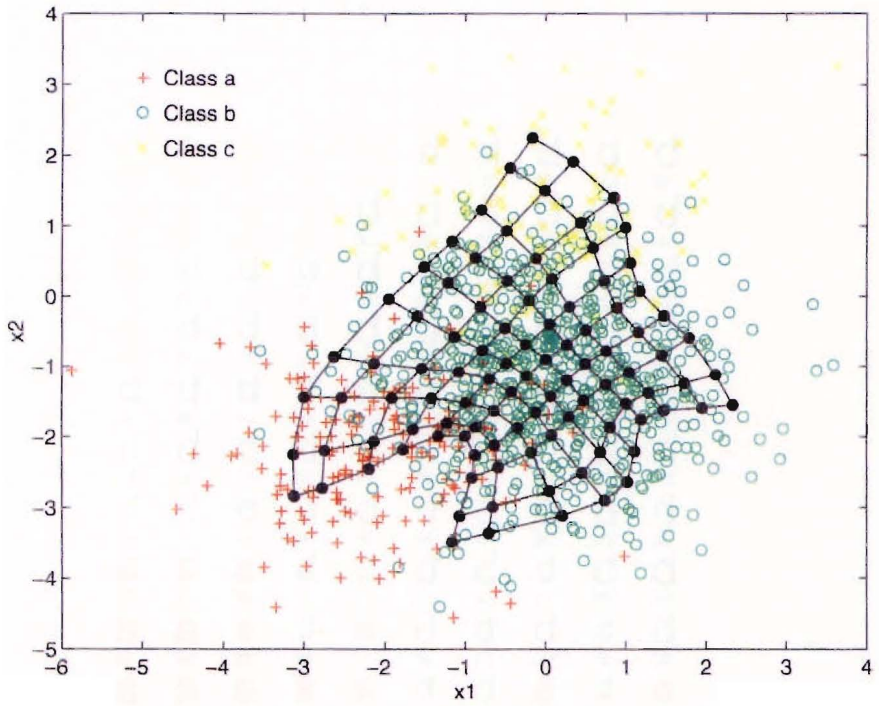


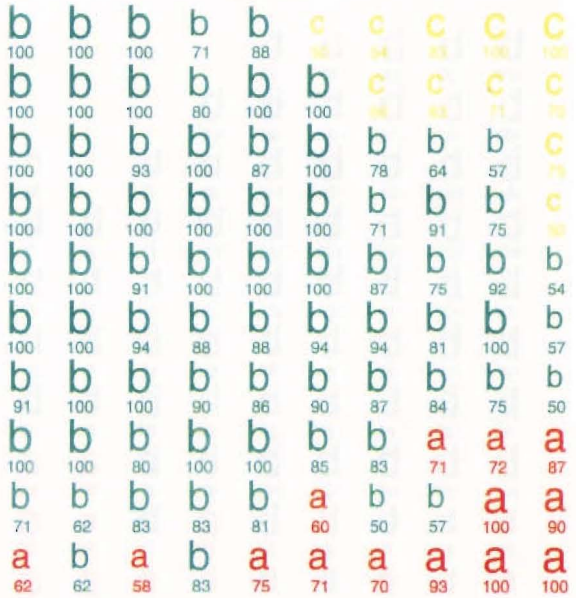
Figure 6.6 A scatter-plot of three clusters of 2D Gaussian processes, C_a, C_b and C_c for simulation 2. Class C_a has a mean of $(-2,-2)$ and a variance of 1.0, class C_b a mean $(-1,-1)$ and variance 1.1 and class C_c a mean $(3,3)$ and variance 1.1. There are 1000 input vectors in total where each class is represented according to the ratio $2:7:1$ for $C_a:C_b:C_c$.

A 2D lattice of neurons was trained as in the previous simulation. Each trained map was calibrated and LVQ1 and LVQ2 performed in each case. Figure 6.7a shows a distribution of the weights of a $[10 \times 10]$ SOFM in the (2D) input space showing the spatial organisation of the weight vectors in 2D. The neighbouring neurons are indicated by the connecting lines between weights. Figure 6.7b shows the same SOFM calibrated after SOFM training. The input data was presented to the SOFM 20,000 times with $\alpha_{\text{min}} = 0.01$, a quadratic neighbourhood taper function and exponential decay for the learning rate and neighbourhood size. Figure 6.8a shows the SOFM after LVQ1 and Figure 6.8b the SOFM after LVQ2.

The performance of each trained and 'fine-tuned' SOFM was calculated using a novel data set derived from the same distributions as the training data. The averaged



(a)



(b)

Figure 6.7 A trained [10 × 10] SOFM for simulation 2. The input data was presented to the SOFM 20,000 times with $\alpha_{min} = 0.01$, a quadratic neighbourhood taper function and exponential decay for the learning rate and neighbourhood size. (a) The position of the weight vectors in relation to the input vectors after SOFM training and (b) the calibrated SOFM after just SOFM training. The value under each class label indicates the confidence level with which that neuron represents its assigned class.

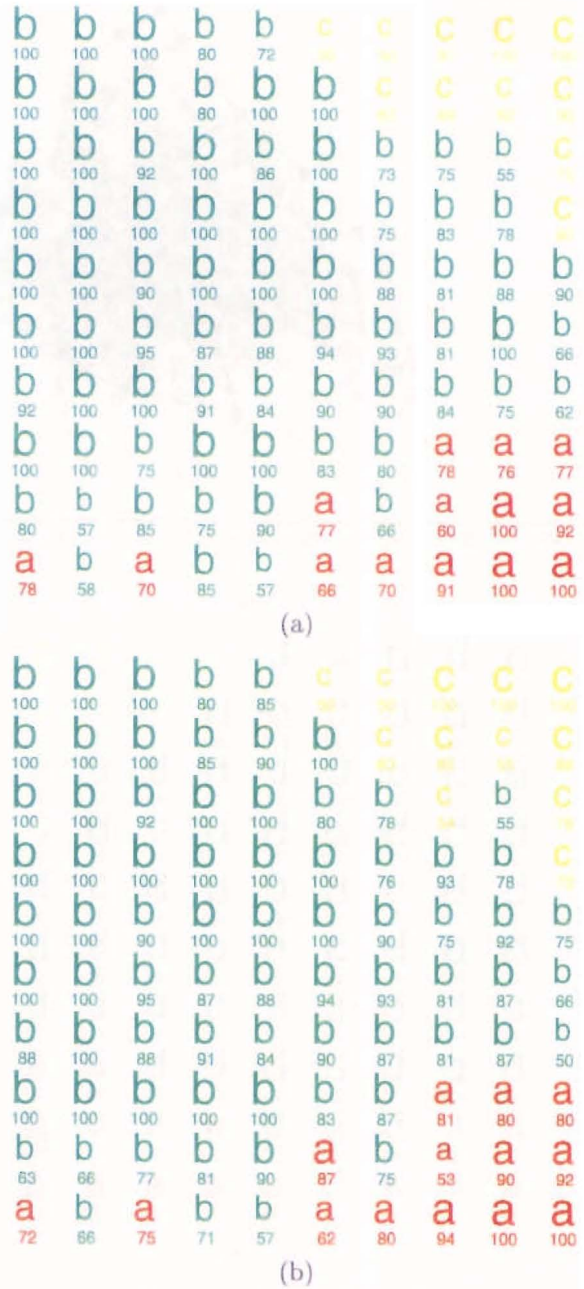


Figure 6.8 The trained [10 × 10] SOFM for simulation 2. The calibrated SOFM after (a) LVQ1 and (b) LVQ2. The value under each class label indicates the confidence level with which that neuron represents its assigned class.

results for each SOFM are given in Table 6.4.

Parameter	Mean performance (%)		
Decay	Linear 70.5	Exponential 70.2	-
Iterations	200 × 70.5	500 × 70.3	-
α_{min}	0.01 70.4	0.001 70.3	-
N_c taper	Quadratic 70.5	Gaussian 70.3	Uniform 70.1
Algorithm	LVQ1 70.4	LVQ2 70.3	SOFM 70.3

Table 6.4 The mean performance of the [10 × 10] SOFM for simulation 2 for each value assigned to the SOFM parameters.

Again, there is very little variation in the average performance over the range of parameters used, as can be seen in Table 6.4. Linear decay offers a very slight advantage over exponential decay (0.3%), as does $\alpha_{min} = 0.01$ over $\alpha_{min} = 0.001$ (0.1%). As in simulation 1, iterating 200× the size of the SOFM results in almost identical performance as iterating 500× the size of the SOFM. Fine-tuning using LVQ1 results in a slight advantage over straight SOFM training (0.1%) and LVQ2 – which performs equal to straight SOFM training. Once more a non-uniform taper scheme proves marginally better than uniform taper. For this simulation the quadratic taper results in a 0.4% increase in performance over the uniform taper and a 0.2% increase over the Gaussian taper.

As in simulation 1, on average there is little, if any, difference between using one parameter over the other for this simulation. The fact that LVQ performs only marginally better than straight SOFM training could be attributed to the fact that, as in the previous case, the class boundaries are well defined. For simulation 2 the data was presented to the SOFM 20,000 times (for 200×) which means that during the crucial first 10% or so of the training (i.e., the ordering phase) the entire training set had been presented at least twice.

From the results presented above (Table 6.4), the best parameters to use for this simulation would be those shown in Table 6.5. Using these particular parameters gives the performance shown in Table 6.6.

Plots of $\delta(k)$ for simulation 2 are shown in Figure 6.9 for various sizes of SOFM. The large changes in Euclidean distance can be seen during the ordering phase until the fine-adjustment phase where the weight changes become less pronounced. As the SOFM size is reduced, the weights become more 'unsettled'.

Once more, topological ordering can be observed when observing the map weights

Parameters	Value
Decay	linear
Iterations	200 ×
$\alpha_{\min}$	0.01
N_c taper	Quadratic
Algorithm	SOFM followed by LVQ1

Table 6.5 The best performing parameters to use for training the [10 × 10] SOFM of simulation 2.

	Sensitivity (%)	Selectivity (%)	Average (%)
SOFM	69.9	72.2	71.1
LVQ1	69.2	72.1	70.7
LVQ2	69.7	73.0	71.4

Table 6.6 The performance of the [10 × 10] SOFM for simulation 2 after iterating 20,000 times, using linear decay, $\alpha_{\min} = 0.01$ and quadratic neighbourhood taper.

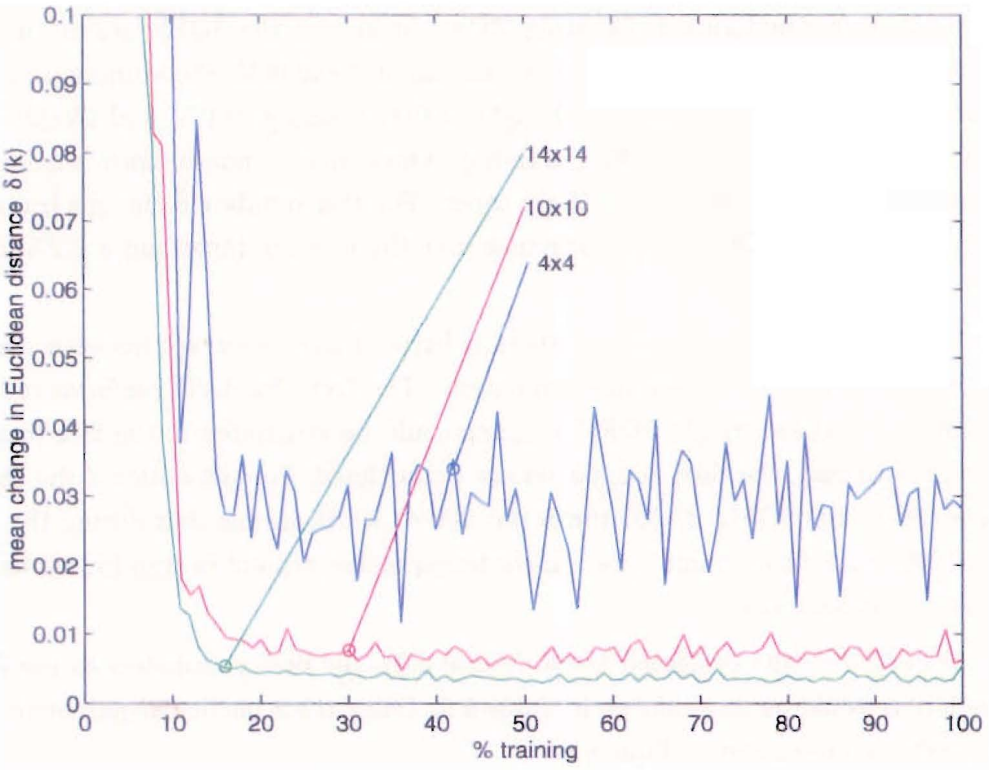


Figure 6.9 The mean change in Euclidean distance between weight vectors during training for different sized SOFMs for simulation 2. During the ordering phase large mean changes in Euclidean distances are observed and during the fine-adjustment phase the mean changes are reduced considerably. As the SOFM size is reduced then the SOFM weights become more 'unsettled'.

in the input space in relation to neighbouring weights. Topological ordering can also be seen when viewing the calibrated SOFMs. The proportion of representation of each class in the input set is also reflected well by the number of weight vectors assigned by the SOFM to represent each distribution. As the classes are represented in the input data according to the ratio 2 : 7 : 1 for $C_a : C_b : C_c$, the SOFM reflects this by mapping more neurons to C_a and less neurons to C_c during calibration.

6.3.3 Simulation 3: Pattern classification III

In this simulation, 41–dimensional input vectors are considered. Each input represents a simulated digitized waveform with a sampling frequency of 200 Hz, so that the 41–dimensional vector represents a 205 ms interval. The 41–dimensional inputs were drawn from six classes, each representing a simulated waveshape. Three classes represent positive going (i.e., upwards) waveshapes: C_a represents a triangular waveshape, C_b a square and C_c a half-sinusoid waveshape. C_d, C_e and C_f represent the equivalent negative going (downwards) waveshapes. In each case each waveform is centred about the centre sample and has a variable height a and width b, as shown in Figure 6.10. As the input data is 41–dimensional it is no longer possible to track the trajectory of the 41–dimensional weight vectors as easily as it was for the 2D case.

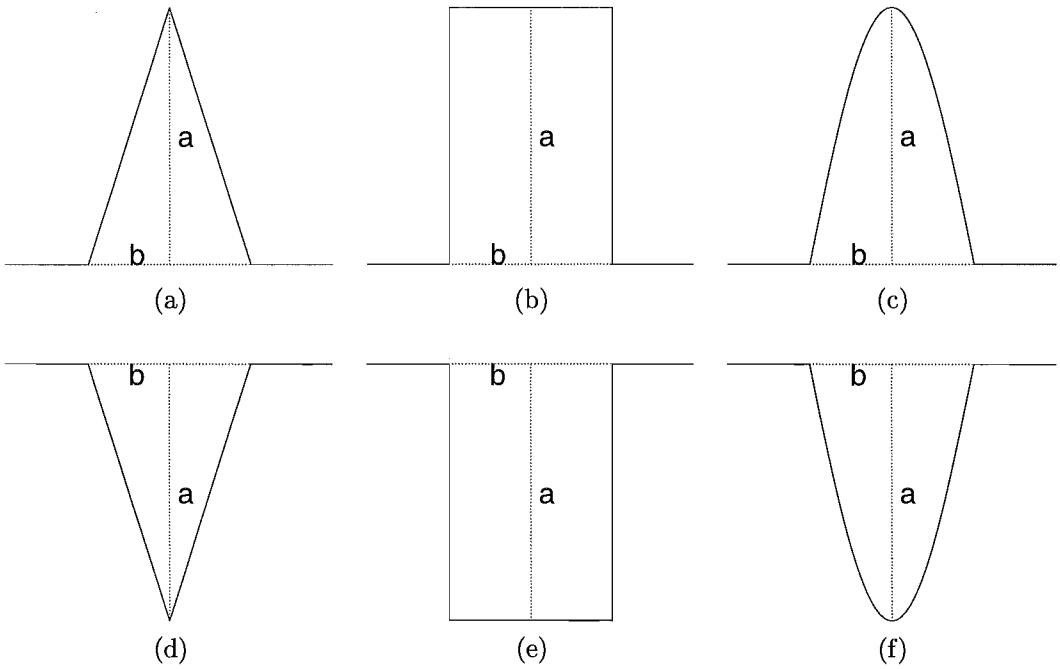


Figure 6.10 Simulated waveshapes used for training the SOFMs of simulation 3. (a) & (d) Triangular, (b) & (e) square and (c) & (f) half-sinusoid.

Simulated data was produced by randomly varying the amplitude a between the values of 0.1 and 1.0 and randomly varying the width between 10 and 30 samples. The

600 input vectors produced, which equally represented each class, were used to train a 2D lattice (10 × 10 neurons) SOFM. As each weight vector is of the same order as the input data (41–dimensions), after training each weight vector comes to represent some form of basis waveform. Figure 6.11a gives a representation of the weight vectors according to their position in the 2D lattice of the SOFM. Once more, the topological ordering of the waveshapes is quite evident.

The trained SOFM is then calibrated and followed by LVQ1 and LVQ2 as shown in Figure 6.11b, Figure 6.12a and Figure 6.12b.

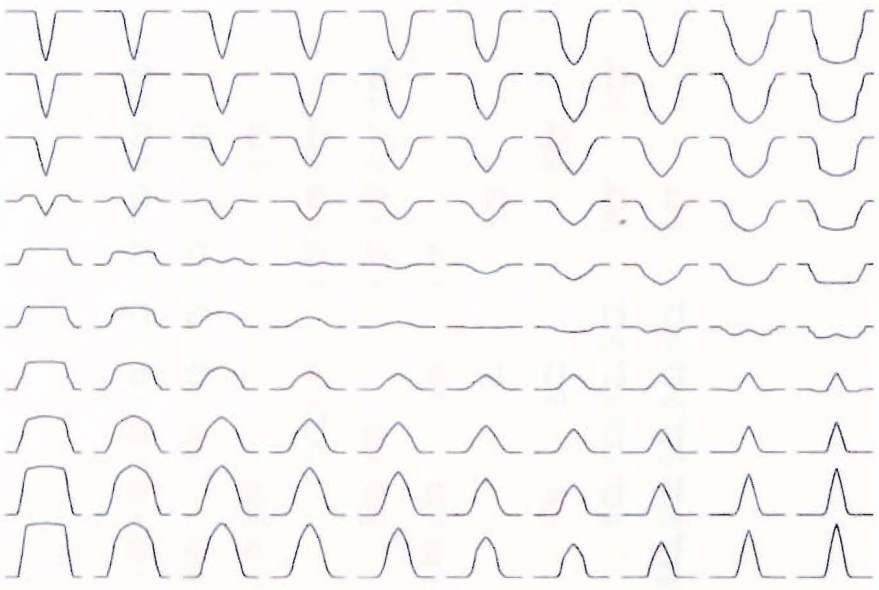
The average performance of the trained SOFM at each stage of the training is assessed in Table 6.7 using a novel data set generated using the same criteria as used for the training set. The average performance is evaluated for each parameter value.

Parameter	**Mean performance (%)**		
	Linear	Exponential	
Decay	66.6	63.4	-
	200 ×	500 ×	
Iterations	66.5	65.8	-
	0.01	0.001	
$\alpha_{\min}$	66.3	66.0	-
	Quadratic	Gaussian	Uniform
N_c taper	66.5	66.1	65.6
	LVQ2	LVQ1	SOFM
Algorithm	74.9	68.4	55.0

Table 6.7 The mean performance of the [10 × 10] SOFM for simulation 3 for each value assigned to the SOFM parameters.

For simulation 3 there is a more pronounced variation in the average performances over the range of parameters used than in the previous two simulations. Linear decay offers an advantage of 3.2% over exponential decay, whilst $\alpha_{\min} = 0.01$ offers a minimal advantage of 0.3% over $\alpha_{\min} = 0.001$. As in the previous two simulations, iterating 200× the size of the SOFM results in a slight increase in performance (0.7%) when compared to that of iterating 500× the size of the SOFM. Fine-tuning using LVQ techniques results in a substantial advantage over straight SOFM training, an increase of 19.9% for LVQ2 and an increase of 13.4% for LVQ1. Non-uniform taper provides a marginally better performance than uniform taper. The quadratic taper results in a 0.9% increase in performance over the uniform taper and a 0.4% increase over the Gaussian taper.

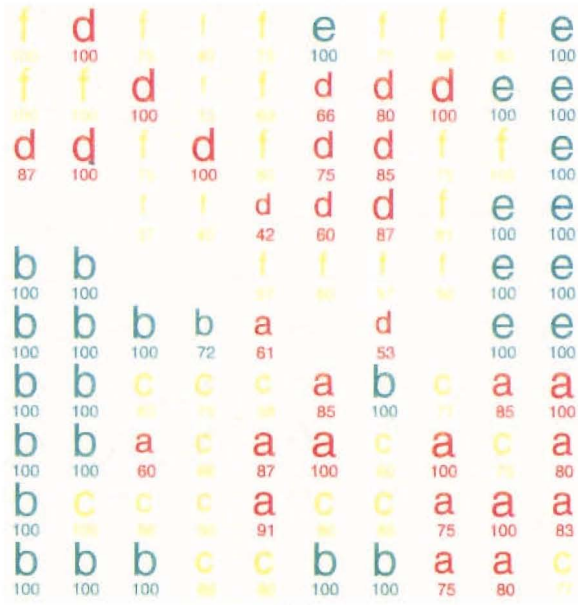
For this simulation the differences in performance are more noticeable than for the previous two cases, this may be due to the fact that the problem in this simulation is a more complex one with more complicated boundaries between each class than the previous two cases. LVQ techniques resulted in a considerable improvement over



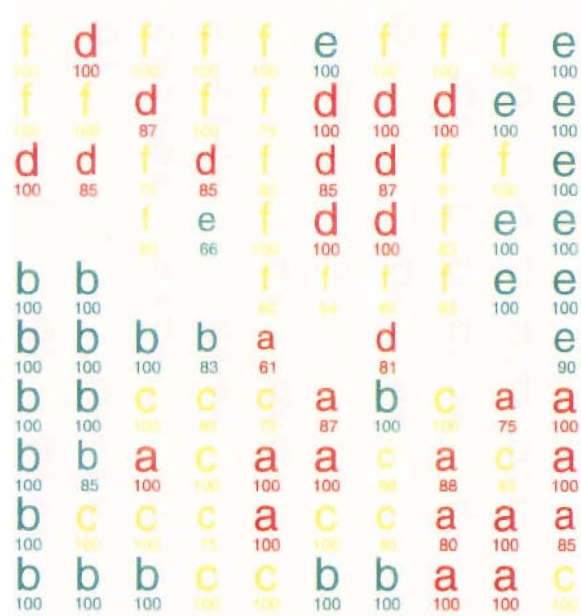
(a)

(b)

Figure 6.11 A trained [10 × 10] SOFM for simulation 3. The input data was presented to the SOFM 20,000 times with $\alpha_{min} = 0.01$, a quadratic neighbourhood taper function and exponential decay for the learning rate and neighbourhood size. (a) The weight vectors of the trained SOFM showing their topological ordering and (b) the calibrated SOFM after just SOFM training. The value under each class label indicates the confidence level with which that neuron represents its assigned class.



(a)



(b)

Figure 6.12 The trained [10 × 10] SOFM for simulation 3. The calibrated SOFM after (a) LVQ1 and (b) LVQ2. The value under each class label indicates the confidence level with which that neuron represents its assigned class.

straight SOFM training, further strengthening the argument that the class boundaries are less readily defined that in the previous simulations.

From the results presented above (Table 6.7) the best parameters to use for this simulation would be those shown in Table 6.8. Using these parameters results in the average performance given in Table 6.9.

Parameters	Value
Decay	linear
Iterations	200 ×
α_{min}	0.01
N_c taper	Quadratic
Algorithm	SOFM followed by LVQ2

Table 6.8 The best performing parameters to use for training the [10 × 10] SOFM of simulation 3.

	Sensitivity (%)	Selectivity (%)	Average (%)
SOFM	54.2	54.7	54.5
LVQ1	71.0	74.2	72.6
LVQ2	75.3	77.2	76.3

Table 6.9 The mean performance of the [10 × 10] SOFM for simulation 3 after iterating 20,000 times, using linear decay, $\alpha_{min} = 0.01$ and quadratic neighbourhood taper.

The average Euclidean distance between the weight vectors during training is shown in Figure 6.13 for various sizes of SOFM. Once more the SOFM weights become more 'unsettled' during the fine-adjustment phase as the SOFM size is reduced.

6.4 CONCLUSIONS

The computer simulations described in the previous sections were undertaken in order to gain a better understanding of the crucial SOFM training parameters. From the results obtained the following conclusions can be made.

The effect of LVQ on the SOFM: Both LVQ1 and LVQ2 were tested on the simulated data. For simulation 1 and 2, the increase in performance due to LVQ was negligible. This may be due to the fact that the input clusters were reasonably well defined and with minimal overlap. As the input data increases in complexity with the introduction of simulation 3, the effect of LVQ after the initial SOFM training becomes more apparent resulting in significant improvements in performance in terms of both sensitivity and selectivity. The increase due to LVQ2 was greater than that due to LVQ1 in simulation 3 and no significant improvements occurred in simulations 1 and 2 through the use of LVQ2 as opposed to LVQ1.

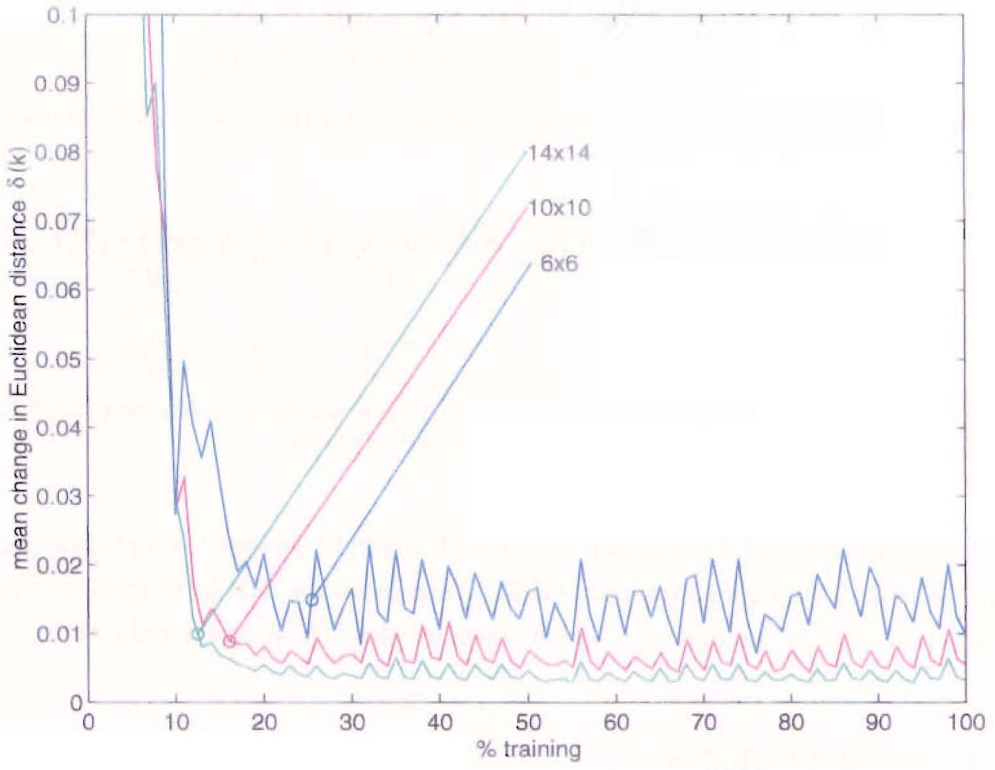


Figure 6.13 The mean change in Euclidean distance between weight vectors during training for different sized SOFMs for simulation 3. As the SOFM size is made smaller the map becomes more 'unsettled'.

For the simpler simulations (simulations 1 and 2) the effect of LVQ sometimes resulted in a slight worsening of the system performance, although this probably reflects experimental/random variations in the training.

The total number of iterations of the data: Iterating the data 500 × the size of the SOFM, as opposed to 200 ×, resulted in no significant improvement in performance in all simulations. The rule-of-thumb suggested by Kohonen [Kohonen 1990] of 500 × the size of the SOFM thus seems unnecessary for small data sets. Another way to determine the number of iterations required would be to base the number on the amount of data to be used for training. In this way the number of iterations could be such that the entire data set is presented to the SOFM *at least twice* during the ordering phase. As the ordering phase is usually around $\frac{1}{10}^{\text{th}}$ of the entire training cycle, this allows the total number of iterations to be calculated.

Neighbourhood taper: In general, a non-uniform taper of the "winning" neighbourhood results in a slightly better performance than a uniform taper, with the quadratic taper marginally better than the Gaussian taper. The difference was more noticeable for simulation 3 than for the other simulations.

Method of decreasing $\alpha(t)$ and $N_c(t)$: Using linear decay, as opposed to exponential decay, for the learning rate and neighbourhood size resulted in marginally better performance over all simulations.

Minimum learning rate α_{min}: Using $\alpha_{\text{min}} = 0.01$ as opposed to $\alpha_{\text{min}} = 0.001$ resulted in a slight improvement in most simulations.

The conclusions drawn above lead to the following recommendations for the adequate training of a SOFM in similar applications to those presented here:

1. Start with a large learning rate; suggested value: $\alpha_0 = 1$.
2. Let the minimum learning rate be a small non-zero value; suggested value: $\alpha_{\text{min}} = 0.01$.
3. Start with a large neighbourhood size, that is, at least with a radius greater than half the size of the SOFM; suggested value: a radius of one less than the widest dimension of the SOFM.
4. Let the minimum neighbourhood size be $N_{\text{min}} = 1$. That is, the winning neuron *and* its nearest neighbours.
5. Decrease the learning rate and neighbourhood size monotonically, making sure that both have reached a minimum by around $\frac{1}{10}^{\text{th}}$ of the total number of iterations of the input data to the SOFM; linear decay is recommended.
6. Use the quadratic neighbourhood taper during weight updates.

7. Iterate the data for *at least* 200 times the size of the SOFM during the training process. Alternatively, for a large input data set, iterate the data such that *all* the data has been presented at least twice during the crucial ordering phase of training (which should be set at around $\frac{1}{10}^{\text{th}}$ of the total number of iterations).
8. Observe the performance of the SOFM with novel data as the SOFM size varies in order to determine the "best" SOFM size for the problem at hand. It may prove useful to monitor the mean change in Euclidean distance ($\delta(k)$) for a number of SOFM sizes. The measure of the mean change in Euclidean distance provides a useful visualization of the *smallest* SOFM size that remains "stable" during training, hence providing a lower limit to the SOFM size.
9. For pattern classification purposes, perform LVQ1 and/or LVQ2 on the trained SOFM. For reasonably well defined input clusters in the input data, both LVQ algorithms should result in similar performance measures. For more complex divisions in the classes of the input data set, LVQ2 is recommended.

6.5 SUMMARY

In order to train a SOFM as a pattern classifier the various training parameters need to be identified and suitable values found for each. In this chapter the most important training parameters for the SOFM are examined through three simulations. The values assigned to the parameters are obtained from "rule-of-thumb" values obtained mainly from Kohonen [1990]. In particular the use of non-uniform neighbourhood weighting over uniform weighting results in marginally improved performance. The values for neighbourhood and learning rate size and decay seem to be quite adequate as long as they both begin "large" and decay monotonically to some "small" value. Of more importance is the fact that there seems to be no gain to be had from iterating the data 500 $\times$ the size of the SOFM, with 200 $\times$ working equally well in each case. This makes for faster training times. For relatively large data sets a useful "rule-of-thumb" developed is to set the maximum iterations such that the entire data set has been presented at least twice 10% of the way through the training. Observing the mean change in Euclidean distance, as well as performance on novel data, should provide a good indication of the "best" SOFM size necessary for the task at hand.

As training a SOFM may involve considerable computation time (especially when large data-sets are involved) it may prove useful to preclude complicated computations, such as Gaussians, when calculating the neighbourhood taper in order to lower the total training time. In such a situation uniform taper would be better suited.

The next chapter introduces the spike detection problem and the proposed spike detection system. As the SOFM forms part of this multistage system, the parameters and measures discussed in this chapter will be utilised in Chapter 9 which discusses the SOFM applied to the spike detection problem.

Chapter 7

THE SPIKE DETECTION PROBLEM

7.1 INTRODUCTION

As stated previously in Chapter 2, the EEG provides an excellent tool in the diagnosis of many brain disorders, in particular, epilepsy. If a subject has exhibited definite or suspected clinical manifestations of epilepsy, routine recordings of the interictal EEG are taken and closely scrutinised by an EEGer in order to detect the presence of epileptiform activity in the form of epileptiform discharges (EDs).

Although a seemingly simple task, the scrutiny requires the EEGer to closely examine a number of channels of EEG recording of around 20 minutes duration (or 200 10-second pages of recording) and detect the presence of EDs manifest as spikes in one or more channels. The presence of artifacts in the EEG makes the job considerably more difficult and the outcome is highly dependent on the EEGer's skill.

It is clear that there are substantial benefits to be gained by automating the spike detection process including reduced time spent reading EEGs, increased consistency, and the detection of EDs (cf. seizures) occurring during long-term EEG monitoring.

This chapter first defines the spike detection problem and then proceeds to list various spike detection methods reported in the literature. Next, the underlying techniques are scrutinised and a proposal for a new multi-stage spike detection system is put forward.

7.2 THE SPIKE DETECTION PROBLEM

Some confusion arises from the use of the terms spike and ED in the literature, the convention adopted throughout this thesis is to use spike and ED interchangeably to refer to epileptiform activity on a *single* channel, and to refer to activity which is in evidence across *2 or more* channels as an epileptiform event (EV).

The spike detection problem can be simply put: detect the presence of EDs in the multichannel EEG recording with high sensitivity and selectivity. That is, a high

proportion of true events must be detected with a minimum number of false detections. Although desirable, it is not realistic to expect sensitivities and selectivities of 100% if for no other reason than the imprecise definition of what constitutes a spike varies among EEGers.

The lack of a proper definition of the 'ideal' spike, other than *"transients clearly distinguished from background activity with pointed peaks at conventional paper speeds and a duration from 20 to under 70 ms approximately"* (for sharp-waves the duration is of 70–200 ms) [Chatrian *et al.* 1974], has rightly caused many researchers to ask:

What does the EEGer look for when visually detecting EDs in the EEG ?

Many researchers have attempted to answer this question by extracting certain features from the raw EEG which, in their opinion, best describe the ED morphology – i.e., *mimetic* approaches. Alternatively, others have opted to use ANNs as a means of utilising the raw EEG without having to make any decision as to what parameters are more important than others in detecting EDs. Özdamar *et al.* [1991] state that the use of preprocessing to extract parameters biases the system and defeats the very purpose of a totally trainable system when utilising ANNs. Paradoxically, Webber *et al.* [1994] report better results (in terms of accuracy and speed of spike detection) through the use of parameterized EEG (as opposed to raw EEG) but then go on to suggest that the network may need more raw test data to abstract identifying features from EDs.

Whatever the method used, the spike detection problem seems to be broken down into two major components: *feature extraction* and *classification* (Figure 7.1). This can be viewed as mapping the N–dimensional EEG space (or pattern space) to a P–dimensional feature space (which is usually of a lesser order than N) and then performing classification in the feature space. In the case of the use of raw EEG data without feature extraction (as is the case in some ANN-based classifiers), this can be seen as the case where the N–dimensional EEG space is mapped onto an identical N–dimensional feature space, where classification then takes place.

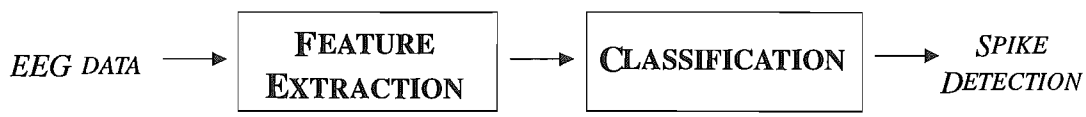


Figure 7.1 The spike detection problem can be broken down into two major components: feature extraction and classification.

If there *are* a set of features which adequately describe the ED morphology, the next obvious question is:

Which features should ideally be extracted from the raw EEG data ?

Almost all researchers answer this question by extracting those parameters which relate to the "sharpness" of the ED. In mimetic approaches, values for peak amplitude, pre-peak slope and post-peak slope, durations, 2nd derivatives, etc., are extracted. In the parametric approach, the sharpness of the ED is used in a statistical setting where the transient or non-stationary nature of the ED is used as a criterion for detection when compared to the (assumed) stationary background. The sharpness of the spike, compared to the background, should also result in a differing spectral content, with more energy being found in the 'higher' frequencies, and, to this end, the FFT is also used. The problem with the FFT is that as the window of EEG being analysed is made wider the temporal information content of the EEG segment being windowed is lost; in addition, the high frequency energy content contributed by the ED is lost by the high frequency energies of the background EEG as the window is widened. The Wavelet transform (WT) can be utilised as a means of providing time-frequency signal analysis capabilities. With the WT it is possible to analyse a relatively large window of EEG in the frequency domain without the loss of temporal features.

Without going into the pro's and con's of each feature set at this stage, it can be said that the mimetic approach does seem to rather limit the information content of the data when compared to, say, the WT approach which provides information in both the time and frequency domain.

Apart from the optimal choice of features to be used for classification, it is well established that, apart from the ED itself, other contextual information is also vital to the EEGer when classifying events as ED/non-ED. These are mainly spatial information, such as

What is happening in other channels at the same time as a candidate ED ?

and temporal information such as

Are there similar events with similar distribution elsewhere in the EEG ?

It is, therefore, somewhat surprising that most of the spike detection systems reviewed work on a channel-by-channel basis (i.e., no spatial and limited temporal information is utilised) and only Glover *et al.* [1989] and Dingle *et al.* [1993] have made any real use of spatial and contextual information, with a high degree of success, through their use of expert systems. Özdamar *et al.* [1991] make use of spatial information by integrating the outputs of individual channel spike detection ANNs (from four channels) into a single ANN module trained to recognise the common spatial distributions of EDs. Webber *et al.* [1994] use four channels simultaneously, while including spatial contextual information of a 1.0 s long window around the ED, in the training of their ANN.

Based on the foregoing, the spike detection problem depicted in Figure 7.1 can now be modified, as shown in Figure 7.2, to incorporate the use of spatio-temporal information in helping detect EDs in the multi-channel EEG.

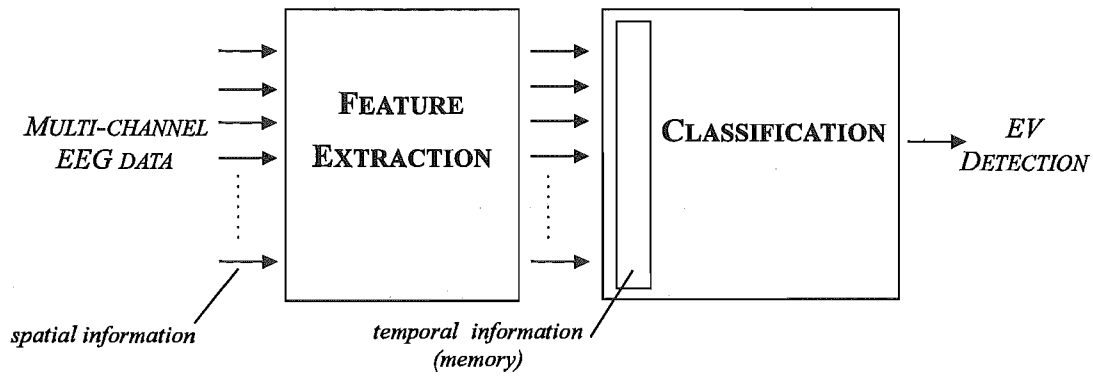


Figure 7.2 Spatio-temporal information is important in the spike detection problem.

The 'ideal' spike detection approach may now be loosely defined as:

> The N–dimensional EEG pattern space is mapped onto a P–dimensional feature space for each channel in the EEG recording. The multichannel features introduce spatial information into the system. Classification of candidate EDs (CEDs) then takes place using features extracted from the pattern space. Temporal information can then be introduced to the classification process by considering the presence of previous (and, if available, future) EDs in the EEG throughout the multi-channel recording and allowing this to strengthen or weaken the outcome due to spatial information alone.

The following provides a short summary of the most common approaches to the spike detection problem in the literature. The systems have been grouped according to their spike detection criterion.

Mimetic: A number of parameters are extracted from the EEG and are thresholded to indicate whether or not an ED has been detected. The parameters used vary amongst researchers. These include: the 2nd derivative of the EEG signal – Walter *et al.* [1973]; the amplitude and duration of EEG halfwaves – Ma *et al.* [1976]; multiple parameters – Gotman and Gloor [1976]; 10 parameters (4 durations, 3 amplitudes, 2 slopes and sharpness) quantifying the morphology of an EEG spike – Ktonas *et al.* [1981]; amplitude and duration parameters including contextual information, patient state and ECG, EMG and EOG information Glover *et al.* [1989]; multiple parameters and a wide temporal context with state of patient recorded as well – Gotman and Wang [1991], Gotman and Wang [1992].

Parametric: The background EEG is assumed to be stationary and is modelled by a number of parameters to make up an autoregressive (AR) filter. When the EEG is then passed through the inverse AR-filter the result should be statistically

stationary and any deviation from stationarity will indicate the presence of a non-stationarity (i.e., an ED or artifact) – Lopes da Silva *et al.* [1974]. Birkemeier *et al.* [1978] use the method of double differentiation followed by linear prediction.

Template: Segments of EEG are correlated with templates of true EDs. A high degree of correlation between template and EEG sample indicates a probable match and hence a detection – Saltzberg *et al.* [1971].

Wavelet transform: Schiff *et al.* [1994] examine the Fast wavelet transform (FWT) as a means of extracting features from the EEG for detecting EDs. The use of the FWT allows temporal features to be retained whilst examining the spectral content of a signal.

ANN (Raw EEG): An ANN is trained with a number of segments of EEG representing ED/non-ED. A 2-layer MLP feed-forward ANN trained in a supervised manner with the backpropagation algorithm – Eberhart *et al.* [1989], Webber *et al.* [1994]. Pradhan *et al.* [1996] use an LVQ ANN (supervised training) trained using a number of labelled epochs of EEG.

Mimetic + Expert system: A number of candidate EDs are selected by the mimetic stage and forwarded to the expert system which then classifies the candidate EDs as true or false. In addition to spatial information, the expert system utilises previous temporal information to make the ED/non-ED decision – Glover *et al.* [1989], or previous and future temporal information – Davey *et al.* [1989], Dingle *et al.* [1993].

Parametric + Template: A non-stationarity is first detected by the parametric stage in a similar manner to the system of Lopes da Silva *et al.* [1974]. The non-stationary waveform is then forwarded to the next stage which compares it to a set of ED templates – Pfurtscheller and Fischer [1978].

Mimetic + ANN: A number of parameters are extracted from the 'raw' EEG and the ANN is trained on those parameters representing ED/non-ED, rather than the 'raw' EEG data itself. Eberhart *et al.* [1989] extract 9 parameters from the EEG and forward them to the ANN. Webber *et al.* [1994] extract 15 parameters/channel across 4 channels. Gabor and Seyal [1992] utilise two ANNs, ANN1 having the pre-peak slope as input and ANN2 having the post-peak slope as input. Wilson *et al.* [1991] utilise 18 parameters/channel across 4 channels of EEG as inputs to the ANN.

Wavelet transform + ANN: Wavelet transform coefficients are extracted from centred segments of EEG (5 seconds long) containing ED/non-ED. The WT is used to obtain both time and frequency domain information. The parameters are

used singly, or in different combinations, to train feed-forward neural networks – Kalayci and Özdamar [1995].

ANN (Raw EEG) + ANN: Candidate EDs are extracted from the raw EEG by 16 identical ANNs previously trained on ED/non-ED data; the outputs of which are integrated by a further ANN to provide spatial information to help in the spike detection problem – Özdamar *et al.* [1991].

7.3 FEATURE EXTRACTION AND CLASSIFICATION

The previous section noted that feature extraction is an important preprocessing stage (i.e., before classification) in the ED detection process. Transforming data from the high-dimensional pattern space to the low-dimensional feature space usually results in improved performance of the classifier stage. Even from a human perception point of view, it is easier to observe patterns in data if certain features are extracted and the data mapped according to these features [Zurada 1992].

Since their inception, ANNs have been ideal candidates for classifiers, especially when little is known about the underlying statistics of the input data and there is a need to generalize to novel data. The following section deals with the use of ANNs in feature extraction as well as discussing the most ideal architectures and learning paradigms for use as classifiers. More detail can be found regarding the ANNs in Chapters 3, 4 and 5.

7.3.1 Artificial neural network feature extractors

As explained in Chapters 3 and 5, there is a form of unsupervised classification learning where no *a priori* knowledge is assumed to be available regarding an input's membership in a particular class. Rather, gradually detected characteristics and a history of training will be used to assist the network in defining classes and possible boundaries between them. Such unsupervised classification is called clustering and the ANN is known as a self-organizing ANN. Clustering is the grouping of similar objects and separation of dissimilar ones. As there is no information available from the teacher on the desired classifier's responses, the similarity of incoming patterns is used as the criterion for clustering.

In effect, the act of clustering input patterns can be seen as a form of feature extraction, as each self-organised cluster is based on some similarities between the constituents of each cluster. Hence, each cluster can be said to contain patterns of similar features.

The most popular of the self-organising ANNs is the SOFM (as described in Chapters 5 and 6). The principal goal of Kohonen's SOFM algorithm is to transform an

incoming pattern of arbitrary dimensions into a one or two-dimensional discrete map. The SOFM performs this transformation adaptively in a topologically ordered fashion.

A further refinement to the SOFM is learning vector quantization (LVQ) (Chapters 5 and 6). Kohonen implements LVQ as a means of 'fine-tuning' the SOFM when it is to be used as a pattern classifier. In this way, the trained SOFM is now being adjusted in a supervised manner.

7.3.2 Artificial neural network classifiers

ANN classifiers can perform three different tasks. Firstly, they can identify which class best represents an input pattern, where the input may be corrupted by noise or some other process. Secondly, the classifiers can be used as content-addressable (or associative) memory, where the class exemplar is desired and the input pattern is used to determine which exemplar to produce. Thirdly, they can vector quantise or cluster the N inputs into M clusters. The number of clusters in the latter can be pre-specified or may be allowed to grow up to a limit determined by the number of nodes in the ANN itself.

The following are just a few of the more popular ANN architectures used as classifiers in the literature. For the most part, ANNs are chosen by their ease of implementation, speed of training, capability to generalize to novel data and resistance to 'noisy' data. More detailed explanations can be found in [Lippmann 1987], [Zurada 1992], [Hush and Horne 1993] and [Haykin 1994].

The single layer perceptron — In Chapter 3 it was shown that if the inputs presented from two classes fall on opposite sides of some hyperplane, then the training algorithm will converge and position the decision hyperplane between those classes. However, when classes cannot be separated by a hyperplane (such as the case of two classes with meshed regions), the single layer perceptron is not capable of accurately predicting the respective class of input patterns.

The multilayer perceptron — It can be seen that a 2-layer perceptron (input nodes, a single hidden layer and the output layer) can form any, possibly unbounded, convex region in the space spanned by the inputs [Lippmann 1987], [Hush and Horne 1993]. Here the term convex means that any line joining points on the border of a region goes only through points within that region. However, this does not mean that there is no benefit to having more than two layers. For some problems, a small 3-layer network can be used where a 2-layer network would require an 'infinite' number (a large number) of nodes [Hagan *et al.* 1996]. A 3-layer perceptron can form arbitrarily complex decision regions and separate meshed classes. It then follows that no more than 3 layers are required in perceptron-like feed-forward ANNs.

Of the number of practical concerns in implementing the multilayer perceptron with backpropagation learning (see Chapter 4) for classification, of most concern is the choice of the network size. For the MLP in general, it is not known what (finite) size network works best for a given problem. With little or no prior knowledge of the problem at hand, one must determine the network size by trial and error. For a fully connected MLP network no more than three layers are generally used and in most cases only 2. For the number of neurons needed in the hidden layer (of a 3-layer MLP), it has been shown that an upper-bound to implement the training data exactly is of the order of the number of training samples [Hagan *et al.* 1996]. However, in order to achieve generalisation, the number of hidden layer neurons should almost always be substantially less than the number of training samples. For a 3-layer ANN, the number of neurons in the second layer must be greater than one when decision regions are disconnected or meshed and cannot be formed from one convex area. In the worst case the number of second-layer neurons required is equal to the number of disconnected regions in input distributions [Hopfield and Tank 1985], [Zurada 1992].

Radial basis function ANN — A radial basis function (RBF) ANN is a 2-layer ANN whose output nodes form a linear combination of the basis (or kernel) functions computed by the hidden layer neurons. The basis functions in the hidden layer produce a localised response to input stimulus. That is, they produce a significant nonzero response only when the input falls within a small localised region of input (or pattern) space. For this reason this ANN is sometimes referred to as the *localised receptive field* ANN [Haykin 1994].

Of interest here is that the RBF ANN can be used for classification just like the MLP, only in this case a single hidden layer is all that is required to implement the RBF ANN whatever the input dimension whereas for the MLP the question arises as to how many layers are required to solve a given problem (although in many cases a 2-layer MLP seems to be enough). In many cases, however, it has been shown that in order to represent a mapping to some degree of smoothness, the number of RBFs required to span the input space adequately may have to be large [Haykin 1994].

Adaptive resonance theory (1&2) ANN — Adaptive resonance theory (ART) ANNs are unsupervised (self-organized) self-stable types of ANN architecture. An ART ANN will remain responsive (i.e., will adapt) when presented with new inputs and yet preserve its previous learned patterns [Haykin 1994], it does this through possessing a dynamic architecture. The ART ANNs function as follows: First an input pattern is presented to the input on a feed-forward basis which produces a feed-backward pattern of activation of one of the existing node classifiers. The feed-forward and feed-backward patterns are then compared. If a

match occurs (resonance) with specified limits (vigilance), then this pattern is classified as a member of that winning node. If not, the remaining nodes are searched; failure to find an appropriate node results in the opening of a new node for this new pattern. So by the network "growing" in size the ART ANN has the ability to learn new patterns without forgetting the old ones.

ART1 and ART2 differ essentially in the nature of their input patterns. ART1 deals with binary inputs, whereas ART2 can deal with continuous valued inputs as well as binary valued inputs [Hagan *et al.* 1996].

7.4 ARTIFICIAL NEURAL NETWORKS AND THE SPIKE DETECTION PROBLEM

The ANN has already featured in various spike detection systems (as shown in Section 7.2). In most cases, the ANN used has been a MLP ANN trained in a supervised manner by repetitively applying known classes of EEG segments to the ANN until it is considered as being adequately trained. Usually this by way of a 2-layer MLP ANN utilized working on EEG samples of a single channel of EEG. This applies to systems that use raw EEG data as input as well as features extracted from the raw EEG. The action performed by these ANNs could be termed supervised classification learning.

Firstly, this may be due to the large amount of applications and documentation of the use of the MLP. Secondly, the ability of the MLP to generalise when novel data is presented is of significant importance for spike detection, as it is almost impossible to present the ANN with a *complete* training set showing *all* ED/non-ED patterns (or features).

Although, the MLP has been the favoured ANN for the spike detection problem, the need to determine parameters such as the number of hidden-layers, the number of neurons in each hidden-layer, activation types, etc. can make training the MLP a problem. The long training times of the MLP may be an issue, although that can be reduced by using the Levenberg-Marquart algorithm (see Section 4.4.4.5). Variable learning can also be used in order to increase the speed and probability of convergence.

No application of RBF ANNs for EEG spike detection has been reported. The RBF certainly has the advantage over the MLP in that only a single hidden layer is required and as a consequence training time is shorter. However, Haykin [1994] states that the generalisation capabilities of the RBF are poor when compared to a similar MLP ANN implementation and that either a larger, more extensive, input pattern set is needed and/or more neurons (basis functions) are required in the hidden layer for the RBF ANN to perform on a level with that of the MLP ANN. The use of RBF ANNs with *adaptive* basis functions is reported [Haykin 1994] and is said to alleviate

the need to increase the pattern set size or increase the number of hidden neurons, but this introduces further parameters for which adequate parameter values need to be chosen.

The use of the ART2 ANN applied to the long-term monitoring of the EEG is reported by Özdamar *et al.* [1992]. The ANN was trained on samples of ED/non-ED obtained as 20 data points of raw EEG. The quick training time for the ART2 ANN is definitely an advantage (a few iterations as opposed to the thousands of iterations required by, say, an MLP ANN), and the ability of the ANN to recognise novel spike features in the incoming EEG and form new classes is also a desirable characteristic. However, a large number of parameters have to be fine-tuned to suit the application (up to seven). Also, the new 'classes' formed by the ANN on its recognising novel EEG data have to be verified and classified by a human operator. Likewise, a few of the parameters need to be adjusted by the human operator to balance the precision and level of *vigilance* (how alert the system is to novel EEG data). For long-term EEG analysis this is ideal as adjustments can be made over the long period of EEG monitoring time available, but for standard EEG recordings this approach is unsuitable in its current form. Elsewhere, another group of researchers have compared the ART2 ANN with Kohonen's SOFM in detecting brain diseases from the contingent negative variation response in the EEG [Jervis *et al.* 1994]. Their findings show that the SOFM was always more accurate than the ART2 ANN with no false detections taking place, whereas the ART2 ANN misclassified events and proved to be far inferior in accuracy. The inaccuracies were attributed to the sensitivity of ART networks to their parameters and to noise.

Researchers' opinions differ as to whether or not the feature extraction stage is a necessary step in the spike detection problem. A feature extractor stage (generally) reduces the pattern space to smaller feature space, hence ANN architecture would be smaller after feature extraction than it would be without. A smaller ANN usually means an increased learning speed and increased probability of convergence. This applies to both MLP and RBF ANNs. In the case of Özdamar *et al.* [1991], fast wavelet transform coefficients were extracted from the EEG and are used as features to train MLP ANNs. The use of the fast wavelet transform coefficients resulted in no increase in performance (when compared to the raw data case) other than the ease and speed of convergence in the training of the ANN. In contrast, Webber *et al.* [1994] found that parameterized data resulted in improved performance of the 'raw' EEG approach.

From the preceding discussion, it is apparent that in deciding on a system capable of the detection of EDs in the EEG, a small number of important questions need to be answered. Figure 7.3 depicts the questions and some of the possible answers. To summarise, these are:

1. Should the 'raw' EEG be used for classification or should features be extracted

first and the classification performed in the new feature space ?

2. If features *are* to be extracted, what features adequately describe the EDs for classification purposes ?

3. Once the decision regarding 'raw' *vs* features has been made, which ANN classifiers (if any) should be used ?

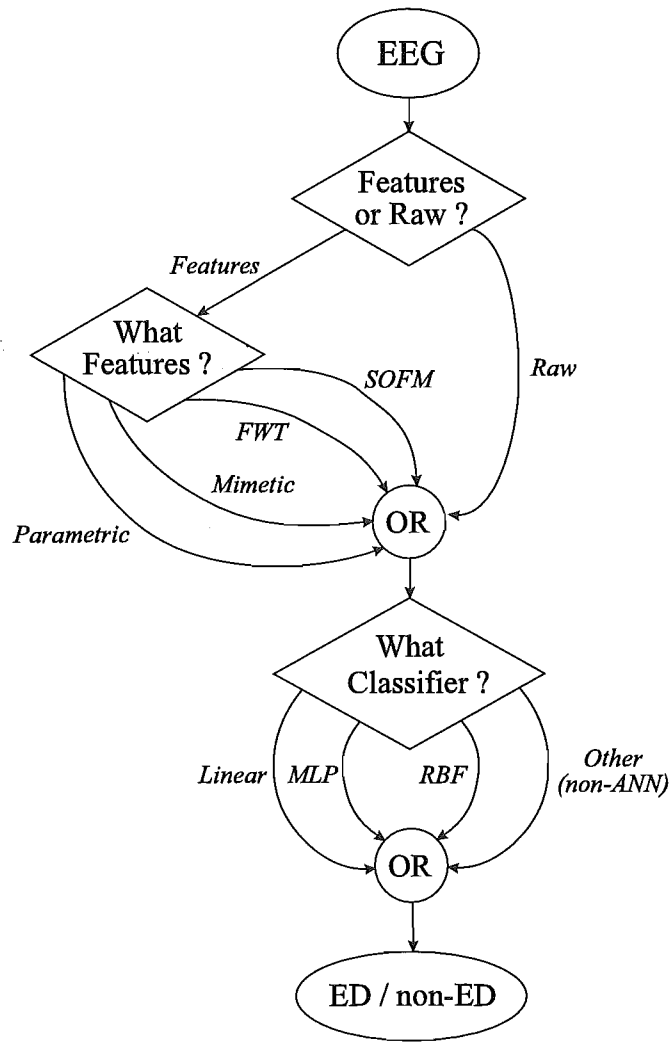


Figure 7.3 Important questions to be asked in choosing the best spike detection criterion.

The next section attempts to answer these questions through the use of the spike detection system which is introduced and elaborated upon further in later chapters.

Once the approach for spike detection has been established, it is important to keep in mind the need to incorporate spatial and temporal information, as described in Section 7.2. At this stage no more mention will be made of spatial and temporal contextual information other than to recognise its importance and to include it in

the overall block diagram of the spike detection system. Utilization of contextual information will be discussed in detail in Chapter 10.

7.5 A NEW SPIKE DETECTION SYSTEM

This section presents a novel approach to the use of ANNs in the spike detection problem. This approach formed the basis of a new spike detection system explained in the following subsections. Furthermore, each stage of the system is discussed in detail in the coming chapters.

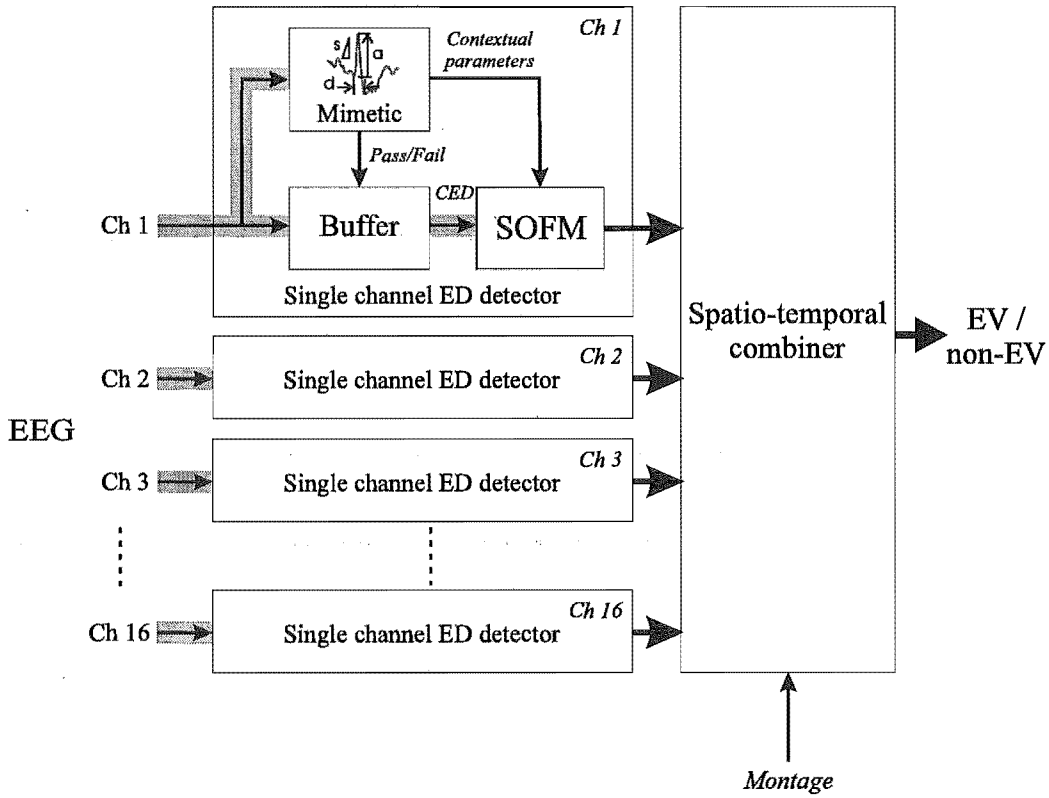


Figure 7.4 The proposed spike detection system.

The spike detection system consists of a multi-stage system at the heart of which is a feature extractor/classifier (Figure 7.4). One stage is made up of two sub-components: a mimetic stage followed by a trained SOFM (see Chapters 5 and 6). The mimetic part extracts and thresholds parameters of the EEG and presents CEDs to the trained SOFM. The SOFM is previously trained on a large training set in a self-organised fashion and results in an ordered set of weight vectors based on the features extracted from the training data, in effect, performing feature extraction on the CEDs. Once the trained SOFM is calibrated and fine-tuned by one of the LVQ algorithms described in Chapters 5 and 6, it then becomes a classifier, assigning class labels to inputs based on

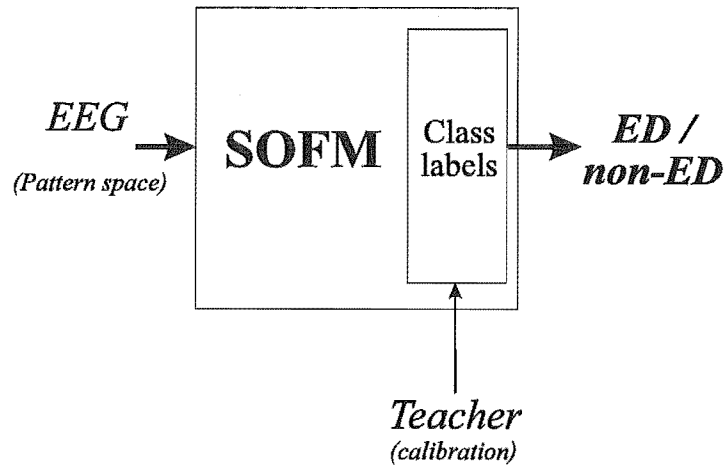


Figure 7.5 The single channel ED detector consisting of the calibrated SOFM followed by 'fine-tuning' with LVQ.

the labels of the closest matching weight vector (Figure 7.5).

Both sub-components form a single channel ED detector and are repeated for each channel in the multichannel EEG. The final stage incorporates the multichannel outputs of the previous stages (i.e., spatial information) along with temporal information to give the final EV/non-EV output of the spike detection system.

Through the use of Kohonen's SOFM, the extraction of features from the raw EEG can be performed without bias, since the SOFM operates in an unsupervised manner representing similar input patterns as topological features mapped out onto a one or two-dimensional feature space.

7.5.1 The inputs to the system

As EDs are said to vary in duration from 20 – 200 ms [Chatrian *et al.* 1974], the ideal inputs to the system would consist of a window of at least 200 ms of 'raw' EEG. The system thus uses an input waveform 200 ms wide with the maximum negative or positive peak $\frac{1}{3}^{\text{rd}}$ of the way across the window.

For a 16 channel system, each input consists of a 200 ms wide waveform obtained from a sliding window across each channel. The reason for going for 'raw' EEG versus parameters extracted from the 'raw' EEG is that the 'raw' EEG contains *all* the information there is to know about the given waveform. Classification based on extracted parameters depends on choosing those parameters that best represent an ED but these are unlikely to contain *all* of the features defining a spike.

The use of 'raw' EEG would, however, have made the system very computationally intensive and hence slow. As one of the aims of the system is to have the system operate

in real-time, an additional stage – i.e., mimetic – was introduced in order to reduce computational time without compromising the system performance.

7.5.2 The mimetic stage

The amount of data presented to the classifier can be significantly reduced by introducing a mimetic stage *before* the classifier stage. The mimetic stage works by extracting a number of parameters from the 'raw' EEG waveforms and passing them through a set of thresholds. Once all thresholds are exceeded it is considered a CED. The *raw* EEG is then passed onto the classifier stage. In this way, the incoming EEG can be screened by the mimetic stage through the use of the parameters, whilst the classifier is still given *all* the information in the form of the 'raw' EEG. By setting the values of the thresholds in the mimetic stage such that the system has a high sensitivity to true EDs (albeit with a poor selectivity) it is hoped that no (or few) EDs are missed at the mimetic stage. In effect the mimetic stage could be called both a 'detector' or a 'classifier' in its own right. The action of forwarding only CEDs which pass the thresholds to the SOFM makes the mimetic stage a 'spike detector', but a detector with an unacceptably large number of false detections due to the low thresholds in use.

Once a CED is found, a number of parameters are calculated which describe the background EEG surrounding the CED. For a 200 ms CED the background EEG of 1.0 s duration is included in the calculations. The parameters measured included values of average amplitude, sharpness and duration of the background EEG. These are calculated in order to place the CED in context as a spike should be *"... clearly visible from the surrounding background EEG."* [Chatrian *et al.* 1974].

These parameters, along with the 'raw' EEG comprising the CED, form the inputs to the next stage in the system. The exact parameters calculated for the mimetic stage are described in more detail in Chapter 9.

7.5.3 The feature extractor/classifier stage

Once a CED is detected it is presented to the next stage which is a combined feature extractor/classifier stage. This stage, along with the mimetic stage, is repeated identically for each channel in the multichannel recording.

7.5.3.1 Feature extractor

A trained SOFM makes up the feature extractor part of the classifier stage. The SOFM, consisting of a square 2D lattice of neurons, was trained using Kohonen's learning rule as described in Chapters 5 and 6. For an $N \times N$ SOFM, N^2 weight

vectors $\mathbf{m}_i$(for $i = 1, 2, \ldots, N^2$) are contained in the SOFM which are of the same order as the input vectors.

A CED is presented to the trained SOFM (along with the contextual information as described in Section 5.6.1) in the form of an input vector. The CED is compared with each weight vector until the closest matching vector is found according to some similarity measure. The closest matching vector $\mathbf{m}_c$ (the 'winning vector') is considered to be best representative of the input data.

As such, the uncalibrated SOFM performs no classification, only feature extraction. However, if the SOFM is *calibrated* as described next, class labels can be assigned to each weight vector $\mathbf{m}_i$ making the SOFM a classifier.

7.5.3.2 Classifier

Calibration of the SOFM is a standard procedure developed by Kohonen (as described in Chapter 5) which assigns a class label to each neuron in the SOFM (and hence to each weight vector $\mathbf{m}_i$). However, in this case the class label assigned to each neuron takes the form of a probability which corresponds to the probability of the input waveform being a true ED if that neuron is declared the 'winner'. So a class label of '0.9' implies a probability of 0.9 that the corresponding weight vector describes a *true* ED whereas a label of '0.1' implies a probability of 0.9 that the weight vector describes a non-ED. The method of assigning probabilities to each neuron during the calibration phase is based on Bayesian statistics and is described in full in Chapter 9.

7.5.4 Enhancing the feature extractor/classifier stage

Once the SOFM is trained and calibrated, it is possible to enhance the performance of the system by 'fine-tuning' the weights of the SOFM through performing LVQ on the trained SOFM as described in Chapter 5. This then becomes supervised training where data (CEDs) with known class labels are used to further adjust the weights of the SOFM such that strong probabilities to true EDs are strengthened.

7.5.5 The multichannel EV detector

So far the system has been developed upon a single channel approach in which the mimetic stage and the feature extractor/classifier stage have been duplicated across all sixteen channels and have all acted on single channel information. The system as described so far would result in sixteen probabilities, each representing the probability of there being a *true* ED at the input of each channel of EEG. The final stage of the system integrates this multichannel information to give the final output of EV/non-EV.

At this stage, spatial information is used to upgrade or downgrade the probability of there being an EV across *all* sixteen channels. The system makes use of the probabilities at each channel of the multichannel recording as well as the polarity of each input waveform (CED) to detect the EV as a whole, i.e., across 16 channels of EEG. As the EEG is recorded using many different *montages*, information regarding the current montage must be also supplied.

Finally, temporal information will be used to upgrade the classification based on the past history of the EEG under review. The actual components of the spatio-temporal stage are explained in greater detail in Chapter 10.

7.6 ENHANCING SPIKES BEFORE SPIKE DETECTION

In the spike detection problem a balance must be obtained between having a high sensitivity and a high selectivity. It is relatively easy to adjust system parameters to obtain performances where *all* EDs are found in a given patient but this would usually be accompanied by an unacceptably large number of false detections. Conversely, it is also relatively easy to achieve a system with very few false detections but then this would usually be accompanied by an unacceptably large number of missed events. Exactly where the balance should be made between sensitivity and selectivity is debatable as well as being dependent on the area of application for automated spike detection.

Many researchers argue that it is better to have a high sensitivity (minimize missed events) and suffer more false detections which can be checked by the EEGer, rather than missing events altogether. Conversely, if we look at the system from the point of view of minimizing the number of false detections then the number of missed events will increase. However, if possible EDs can be enhanced *prior* to the spike detector it should be possible to increase the sensitivity (minimize missed events) while maintaining the selectivity at a satisfactory level.

A spike enhancer would not be a detector but would simply aim to enhance anything vaguely spike-like. In reality this means that real EDs as well as ED-like artifacts and background will be enhanced, i.e., a large number of unwanted waveforms will be enhanced along with real EDs. This is quite acceptable as long as the spike detection system has a high selectivity (i.e., the number of false detections is minimized). Such a spike enhancer is described in the next chapter. It uses the technique of multireference adaptive noise cancelling to enhance spikes in the EEG.

The overall spike detection system can now be expanded so as to incorporate a spike enhancer before the spike detector stage (Figure 7.6).

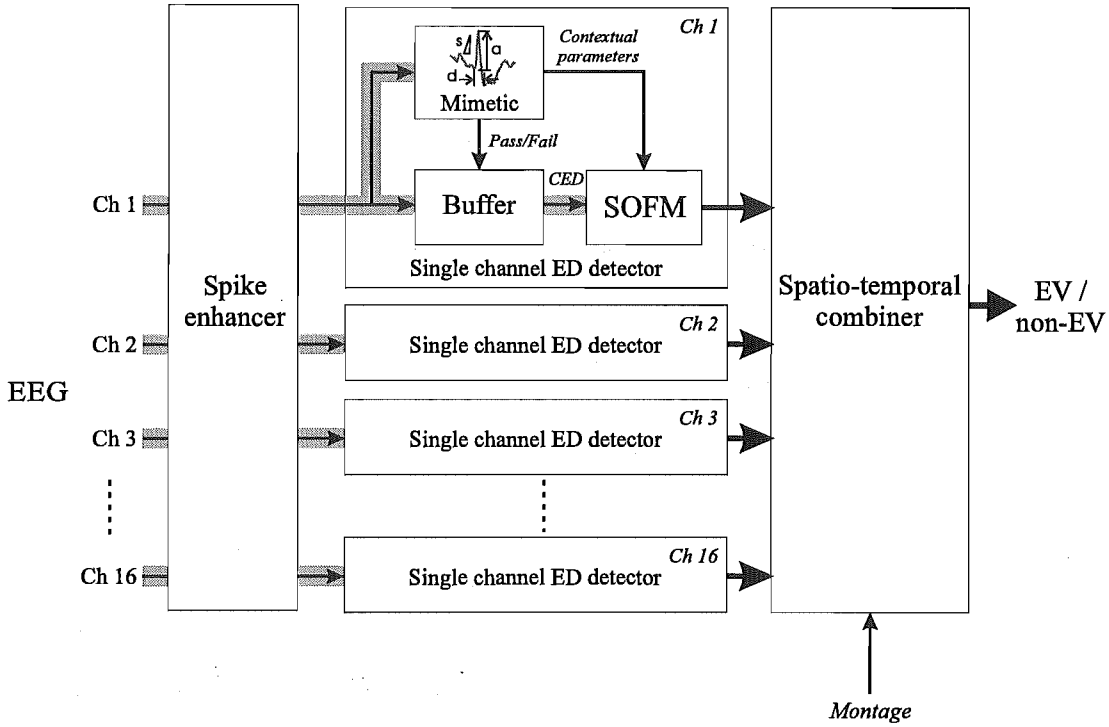


Figure 7.6 The spike detection system, including a spike enhancer.

7.7 SUMMARY

The conceptual framework for a new spike detection system has been developed which makes as much use of information in the EEG as possible whilst attempting to emulate the EEGer's approach to the spike detection problem as much as possible. The system makes particular use of the attributes inherent in ANNs (i.e., nonlinear, adaptive, etc.).

The key factors during the system development which led to the system architecture depicted in Figure 7.6 were:

- Although the mimetic stage extracts parameters for thresholding, once a CED is found the 'raw' EEG is input to the feature extractor/classifier stage in order not to lose important information by basing classification on arbitrarily extracted parameters only.

- Contextual information on the surrounding background EEG is also forwarded to the feature extractor/classifier stage to increase accuracy of classification by the SOFM.

- Since the SOFM is initially formed following a self-organising process, a large amount of training data can be supplied for training purposes which does not have to be seen or graded by expert EEGers. This is a major advantage over

other ANN based systems which use supervised ANNs for classification and hence need large amounts of graded EEG data to form an adequate training set.

- Spatial and temporal cues are incorporated in the system in order to gain the benefits obtainable from this information in a means very similar to that used by an expert EEGer.
- Grading of the overall system output is possible not only through using the spatial and temporal cues present in the EEG but by assigning probabilities to CEDs at the output of each single-channel EEG module rather than straight binary ED/non-ED class outputs.
- A spike enhancer is included in an effort to increase the sensitivity, whilst maintaining a high selectivity, of the overall spike detection system.

Each stage of the spike detection is described in more detail in the following chapters.

Chapter 8

SPIKE ENHANCEMENT

8.1 INTRODUCTION

This chapter introduces a system – called the 'spike enhancer' – designed to enhance the presence of EDs in the EEG. It is the first stage of a multistage system to detect the presence of EDs in the EEG, as described in Chapter 7 (see Figure 8.1).

The EEG can be considered as consisting of an underlying background process which is assumed to be stationary and ergodic, onto which are superimposed transient non-stationarities such as EDs, electrode 'pop', eye-blinks, and muscle artifacts. Of the many methods described in Chapter 7 for the detection of EDs in the EEG, the method of Lopes da Silva *et al.* [1974] best represents this superposition model and has received much attention by workers in the field. The method involves modelling the (stationary) background EEG with an autoregressive prediction filter and detecting transients by examining the prediction error. This must then be followed by a further stage to classify the transients detected as being EDs or artifacts. In the Lopes da Silva *et al.* method the autoregressive filter is calculated from a segment of the background EEG which is assumed to be stationary. Its major drawback is that the stationarity assumption may not always hold true, leading to an unacceptably large number of transients being detected.

Essentially, the spike enhancer processes the EEG by attenuating the background EEG, thus primarily leaving only transients – which are then classified as ED or non-ED by following stages which are described in the next chapters. The ultimate aim of the spike enhancer is to increase the sensitivity of the overall system to candidate EDs (CEDs), while maximizing selectivity (i.e., minimizing the number of CEDs which are *not* epileptiform passed onto the next stage). The spike enhancer makes use of *multireference adaptive noise cancelling* (MRANC) in which the background EEG on nearby channels in the multichannel EEG recording is used to adaptively cancel the background EEG on the channel under investigation. The use of multilayer ANNs to implement the MRANC filters allows the relationship of EEG source to EEG signal to be nonlinear and exhibits improved performance over the linear case. The fact that

the system utilizes filters which are continuously adapting means that any variations in characteristics of the background EEG, such as baseline drift, or changes in level of α-activity, will not affect detection, and the different characteristics of the EEG of different patients will automatically be compensated for within the first few seconds of the detection process (the time taken by the ANN to train to an acceptable level). Adaptive noise cancelling has been applied to enhancing somatosensory evoked potentials [Parsa and Parker 1994] and to cancelling the presence of EOG in the EEG [Sadasivan and Narayana Dutt 1996], although both methods use a linear implementation of the noise cancelling technique.

The first part of this chapter introduces a model which describes the generation of the EEG as recorded at the scalp electrodes. The technique of adaptive noise cancelling is then introduced followed by MRANC which is applied to the model. The MRANC process is implemented by means of a multilayer perceptron ANN. The method has been applied to recorded EEG segments and the performance of the system on predetermined EDs recorded. A comparison is made to the performance of MRANC utilizing a linear model. The advantage of the spatial filtering aspect of MRANC is highlighted when the performance of MRANC is compared to that of the inverse auto-regressive filtering of the EEG, a purely temporal filter. Finally, the effectiveness of the method at enhancing such activity in the EEG is discussed.

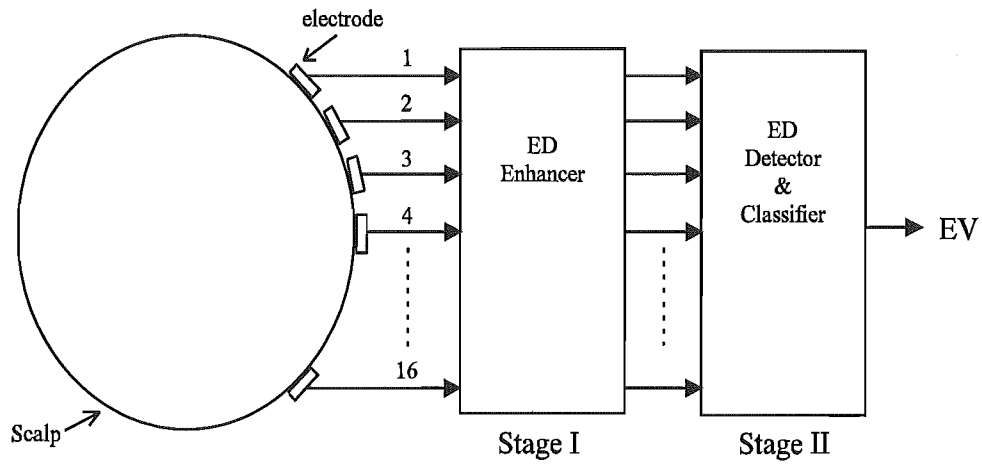


Figure 8.1 The complete spike detection system.

8.2 THE EEG MODEL

There is considerable discussion in the literature on the origin of the EEG. Chapter 2 introduces a number of ideas put forward by various researchers in the field. All researchers agree, however, that the EEG recorded at the scalp represents an averaged

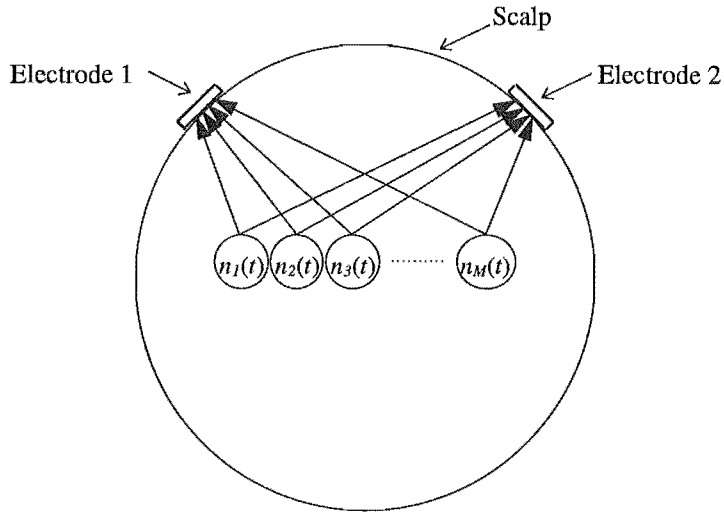


Figure 8.2 A model of the generation of the EEG at the scalp.

picture of the activity of many individual neurons or groups of neurons. The scalp EEG can therefore be modelled as shown in Figure 8.2, where a (large) number of signal generators, $n_i(t)$ (for $i = 1, 2, \ldots, M$), are shown to be generating electrical activity in the brain which is picked up by scalp electrodes after passing through brain tissue, cerebrospinal fluid and the skull. In engineering terms, the signals recorded at each electrode can be defined as the weighted sum of all the generating signals.

Since the number of signal generators, along with the signals they generate, is not known, a simpler model is assumed here. That is, each of the above signal generators can be combined in a certain way to produce a single underlying signal. However, since we are interested in detecting EDs in the EEG, and bearing in mind that the interictal EEG can be considered as consisting of a background process onto which are superimposed EDs, we consider that the individual signal generators can be lumped into two composite generators: the background signal $N(t)$ and the ED signal $S(t)$. The resulting model is shown in Figure 8.3. Thus, the signal recorded at each scalp electrode, $E_j(t)$, (for $j = 1, 2, \ldots, L$), becomes the weighted sum of $N(t)$ and $S(t)$.

The filtering action of the brain can be described in terms of a mathematical model describing the *tissue transfer functions*. These can be represented by means of input-state-output differential equations [Narendra and Parthasarathy 1990], which take the form

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{F}\begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix} \\ \mathbf{E}(t) &= \mathbf{G}\left[\mathbf{x}(t)\right], \end{aligned} \tag{8.1}$$

where

$$\begin{aligned} \mathbf{x}(t) &= [x_1(t), x_2(t), \ldots, x_r(t)]^T, \\ \mathbf{u}(t) &= [N(t), S(t)]^T, \\ \mathbf{E}(t) &= [E_1(t), E_2(t), \ldots, E_L(t)]^T, \end{aligned}$$

with $u_i(t)$ representing the inputs to the tissue transfer functions, $x_i(t)$ the state variables and $E_i(t)$ the outputs from the tissue transfer functions recorded at the scalp electrodes. $\mathbf{F}$ and $\mathbf{G}$ represent unknown functions. These equations describe a system of order r with 2 inputs and L outputs. Equation 8.1 can be re-written as a set of corresponding difference equations, since here we are dealing with discrete-time systems, thus

$$\begin{aligned} \mathbf{x}(k+1) &= \Phi \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix} \\ \mathbf{E}(k) &= \Psi[\mathbf{x}(k)]. \end{aligned} \tag{8.2}$$

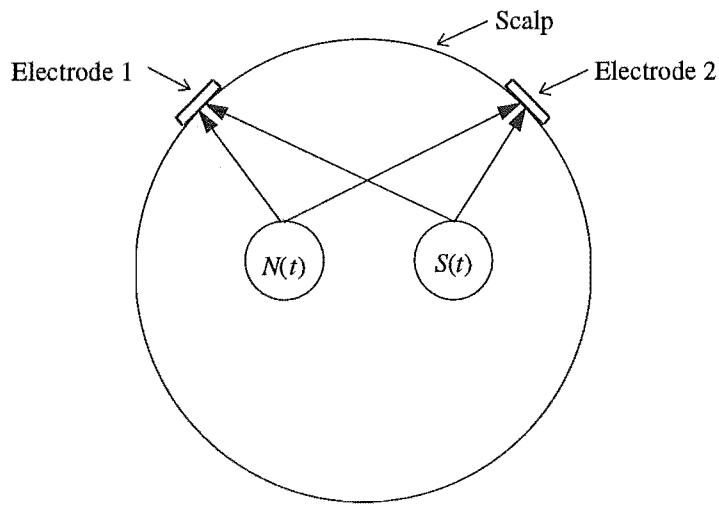


Figure 8.3 The simpler model of the generation of the EEG.

It is not known whether the tissue transfer functions perform a linear or nonlinear operation on the underlying signal generators. For generality, therefore, they are henceforth assumed to be nonlinear (and unknown). Even if they were known, solving the nonlinear equations of Equation 8.2 results in nonlinear algebraic equations for which no simple solution exists. By expanding Equation 8.2 we get an expression for $\mathbf{E}(k)$ —

the signals recorded at the scalp electrodes,

$$\begin{aligned}
\mathbf{E}(0) &= \Psi\left[\mathbf{x}(0)\right] \\
\mathbf{E}(1) &= \Psi\left[\Phi\left[\begin{array}{c}\mathbf{x}(0)\\ \mathbf{u}(0)\end{array}\right]\right] \\
&\vdots \\
\mathbf{E}(k) &= \Psi\left[\Phi\left[\Phi\left[\begin{array}{c}\cdots \quad \left[\Phi\left[\begin{array}{c}\mathbf{x}(0)\\ \mathbf{u}(0)\end{array}\right]\right] \quad \cdots \\ \mathbf{u}(1) \\ \vdots \\ \mathbf{u}(k-1)\end{array}\right]\right]\right]\right].
\end{aligned} \tag{8.3}$$

From Equation 8.3 it can be seen that $\mathbf{E}(k)$ is a nonlinear function of previous values of $\mathbf{u}(k)$. In essence, the model described here shows that the EEG signal recorded at the scalp can be considered as being due to two separate sources consisting of the ED and contaminating background EEG. The tissue transfer functions giving rise to signals as measured at the scalp are unknown and are assumed to be nonlinear. It is the EEG model described by Equation 8.3 which will be used by the noise cancelling procedures (using ANNs) described in the next sections.

8.3 ADAPTIVE NOISE CANCELLING THEORY

When it is required to estimate a signal which is contaminated with additive noise, it is usual to pass the contaminated signal through some fixed filter to attenuate the noise while leaving the signal relatively unchanged. To construct such a fixed filter one must have *a priori* knowledge of the characteristics of the signal and noise. With adaptive filters, however, no *a priori* knowledge of the signal or noise is necessary as the filter 'learns' and adapts its parameters as time progresses.

The concept of adaptive noise cancelling is shown in Figure 8.4. A signal s_0 is received at a sensor along with some (uncorrelated) noise n_0. This noise is unwanted and is contaminating the signal s_0. The noise cancelling scheme requires a separate input which yields a version of the noise n_1 which is correlated to n_0 The combined signal $s_0 + n_0$ forms the primary input to the noise canceller and the noise n_1 forms the reference input. The reference signal is filtered in such a way as to produce a signal y which is as close as possible to n_0. The overall output of the system is given by $z = s_0 + n_0 - y$. The output power can be found by squaring z and taking expectations

such that

$$\begin{aligned} E[z^2] &= E[s_0^2] + E[(n_0 - y)^2] + 2E[s_0(n_0 - y)] \\ &= E[s_0^2] + E[(n_0 - y)^2]. \end{aligned} \tag{8.4}$$

The minimum output power is given by

$$\min E[z^2] = E[s_0^2] + \min E[(n_0 - y)^2]. \tag{8.5}$$

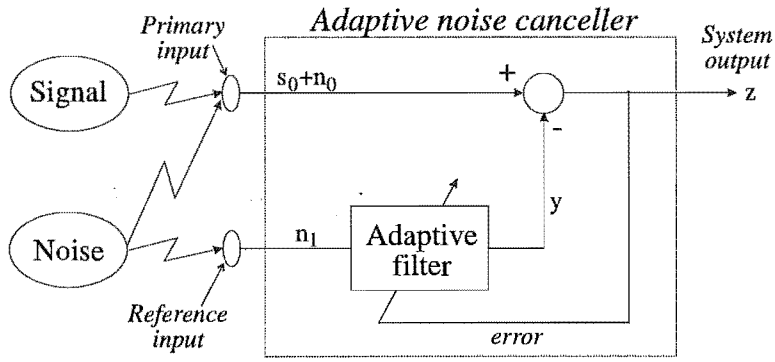


Figure 8.4 Adaptive noise cancelling *without* crosstalk.

The signal power $E[s_0^2]$ is unaffected as the adaptive filter is adjusted to minimize $E[z^2]$. The above equations imply that minimizing the total output power has the effect of causing the output z to be a best least squared estimate of the signal s_0 [Widrow *et al.* 1975], [Widrow and Stearns 1985].

It is possible that the reference input may be contaminated with some signal s_1 (correlated with s_0), as shown in Figure 8.5. This signal *cross-talk* will result in some cancellation of the primary input signal but, if the levels of cross-talk are low, the adaptive noise canceller will still work well, reducing the SNR at the output with minimal distortion of the primary signal s_0 [Widrow and Stearns 1985].

Figure 8.6a shows an adaptive noise cancelling scheme where the signal source and noise source are both uncorrelated white noise sources (each signal is 2000 samples long). The amount of noise contaminating the primary input is governed by the attenuator J and the amount of signal crosstalk by H. For the sake of simplicity all the other paths are directly connected (i.e., no attenuation) and both J and H are zero order filters (i.e., attenuators). The noise cancelling system consists of a single weight and bias initialised to zero. As both J and H are varied, the noise canceller is operated and the adaptive filter characteristics are changed using the LMS algorithm as described by Widrow and Stearns [1985]. Pattern mode training was used with the

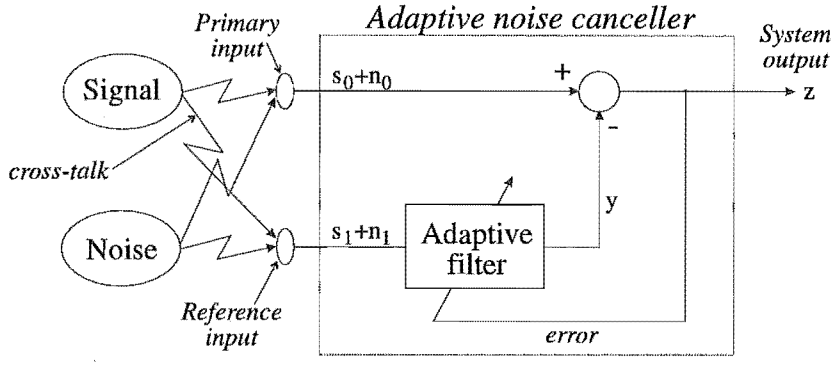


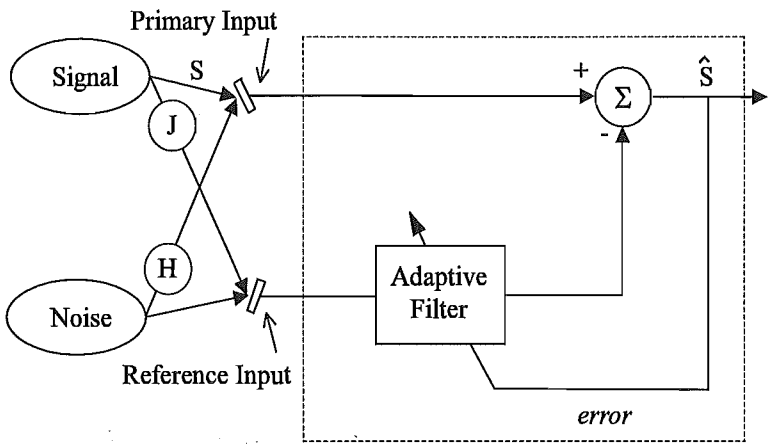
Figure 8.5 Adaptive noise cancelling *with* crosstalk.

LMS algorithm meaning that the adaptive filter weight and bias were changed after each consecutive sample was presented. The algorithm used a fixed learning rate of $\alpha = 0.01$. The mean-squared-error (MSE) between the output of the noise canceller and the signal applied at the primary input, is measured after equilibrium (i.e., after the first 1000 samples) and plotted for each case, as shown in Figure 8.6b. It can be seen that for no (or little) signal cross-talk, the noise canceller works well, resulting in a minimal MSE. As the level of signal cross-talk increases appreciably then so does the MSE indicating that the signal is being distorted as well as the noise being cancelled.

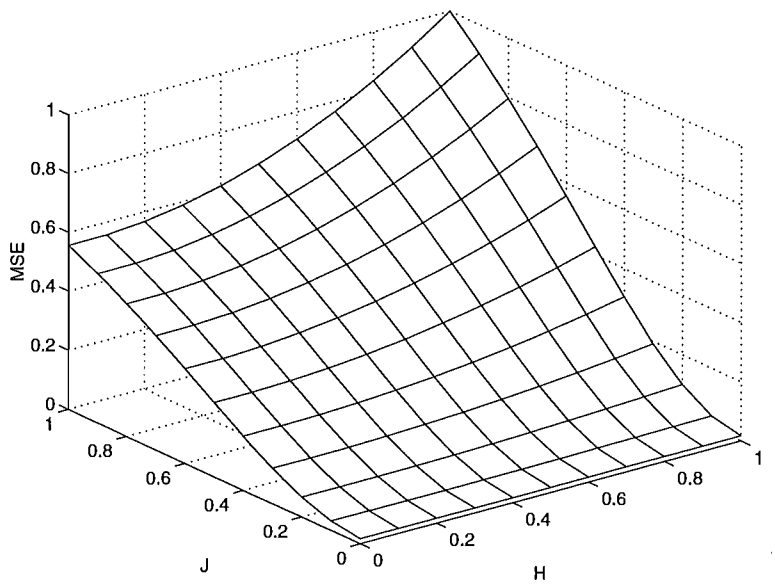
8.4 MRANC FOR THE EEG

The adaptive noise cancelling theory is applied to the spike detection problem in the EEG in the following way. The desired signal $S(k)$ represents the EDs, while the noise source $N(k)$ represents the background EEG. This process is shown in Figure 8.7.

As can be seen in Figure 8.7, the primary input $E_{pri}(k)$ contains a signal s contaminated by noise n_0 from the noise source; n_0 is assumed to be uncorrelated with the signal s. The reference input $E_{ref}(k)$ contains a noise signal n_1 which is uncorrelated with s but correlated with n_0. The adaptive filter adapts its parameters so as to produce an output signal y which is as close as possible to n_0. This output is then subtracted from the primary input, cancelling the noise content n_0 but leaving signal s intact. The filter adaptation is controlled by the error which is the overall system output z. The adaptive filter continuously adjusts so as to minimize the error signal given by z. Any suitable adaptive algorithm which minimizes this error can be used; in particular, the LMS adaptive algorithm [Widrow and Stearns 1985] can be used if the system is assumed to be linear. The LMS algorithm minimizes $E[z^2]$ and in principle the filter converges to the optimal solution and the noise canceller has at its output the signal itself, free of contaminating noise. This corresponds to a filtered version of the ED as recorded at the primary input scalp electrode, free of contaminating background



(a)



(b)

Figure 8.6 Computer simulations of adaptive noise cancelling: (a) A single reference adaptive noise canceller and (b) the MSE given by $E[(\hat{S} - S)^2]$ after noise cancelling using the LMS algorithm for various values of J and H. Both noise and signal source are represented by uncorrelated, white noise signals.

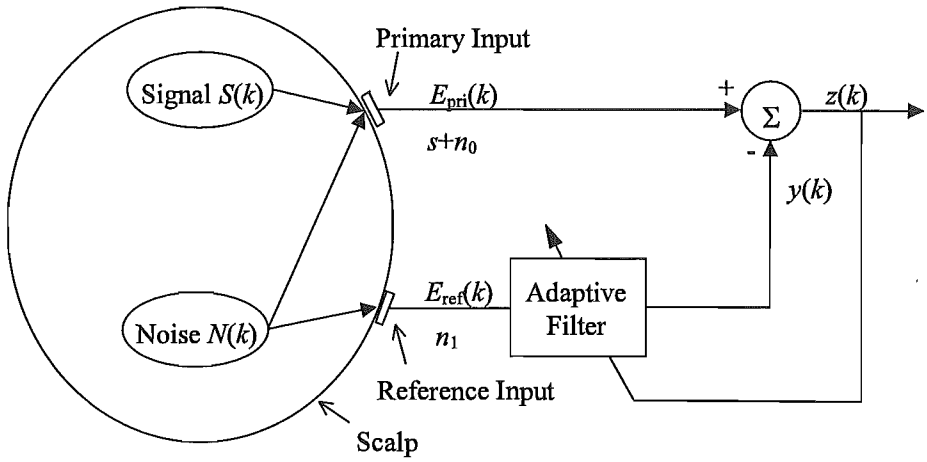


Figure 8.7 Adaptive noise cancelling applied to the EEG.

EEG. The LMS algorithm is employed in the work reported here for comparison with the nonlinear ANN described next.

The example in the previous section (related to Figure 8.5) indicated that some cancellation of the signal in the primary input occurs if signal cross-talk is present. If the signal components present in the reference input are small, then the noise canceller can be shown to work nearly as well as if there were no cross-talk. If the noise canceller is extended to the multichannel case – i.e., MRANC – it can be shown that as the number of reference channels is increased, as in Figure 8.8, the performance is improved, even in the presence of a limited amount of cross-talk on some of the channels [Ferrara and Widrow 1981]. So with MRANC each reference input $E_{ref1}(k), E_{ref2}(k), \ldots, E_{refN}(k)$ contains a version of the noise correlated with the noise-source $N(k)$ and some reference inputs may contain some cross-talk of the signal $S(k)$ itself.

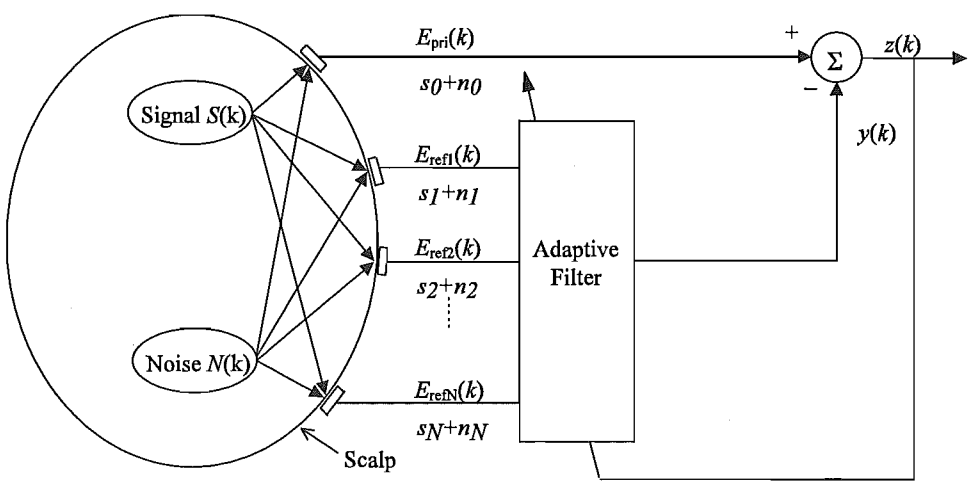


Figure 8.8 Multireference adaptive noise canceller.

As the adaptive filter is attempting to model the unknown and possibly nonlinear transmission of $S(k)$ and $N(k)$ through the tissue transfer functions to the scalp, it must itself exhibit nonlinear behaviour. Equation 8.3 indicates that each reference input $E_{ref1}(k), E_{ref2}(k), \ldots E_{refN}(k)$ is a function of both past and present signal values. This implies that the adaptive filter needs access to past as well as present reference inputs. For practicality, only a finite number p of past values are considered (and implemented by way of tapped delay lines for each reference input as shown in Figure 8.9).

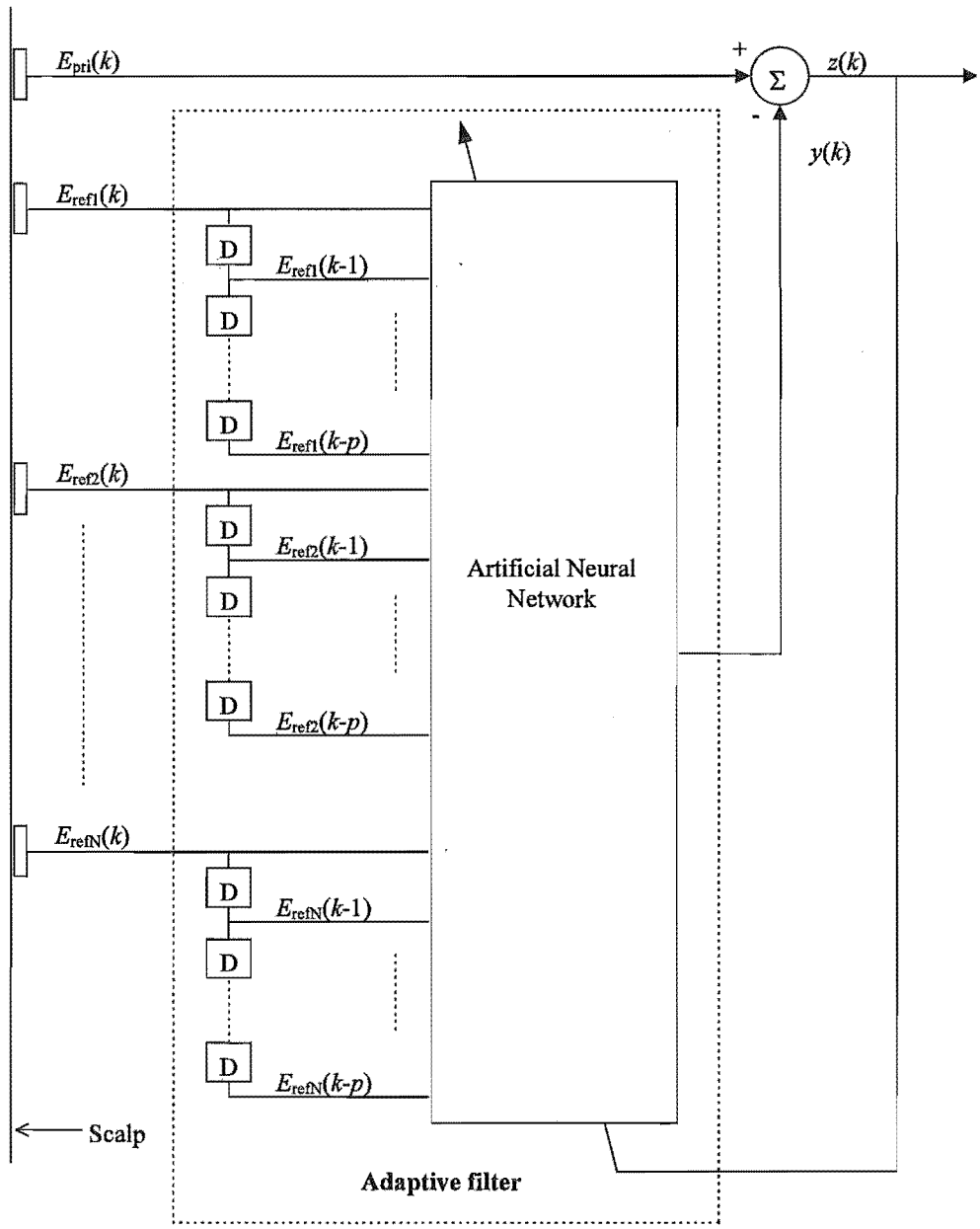


Figure 8.9 The tapped-delay-line in the adaptive filter.

8.4.1 ANN implementation of MRANC

As stated in Chapter 4, a 2-layer perceptron ANN (one hidden layer and one output layer of neurons) can implement any arbitrary nonlinear function, so it is well suited to implement the nonlinear adaptive filter described here. Such an ANN has $N(p+1)$ inputs (where N is the number of reference channels), an arbitrary number H of neurons in the hidden layer and a single neuron in the output layer (refer to Figure 8.9). For this ANN each neuron in the hidden layer contains a log-sigmoidal nonlinear activation function given by

$$\gamma(x) = \left(1/(1+e^{-x})\right), \tag{8.6}$$

and the single output neuron has a linear activation function. The action of the 2-layer ANN is described in the following equations.

For the hidden layer, the net internal activity level $n_j^{(h)}(k)$ for neuron j at sample time k is

$$n_j^{(h)}(k) = \sum_{i=0}^{N(p+1)} w_{ji}^{(h)}(k) a_i^{(in)}(k) \quad \text{for } j = 1, 2, \ldots, H, \tag{8.7}$$

where $a_i^{(in)}(k) = E_{refi}(k)$ is the input signal of neuron i and at time k , and $w_{ji}^{(h)}(k)$ is the weight of neuron j in the hidden layer for the i^{th} input signal. For $i = 0$, we have $a_0^{(in)}(k) = +1$ and $w_{j0}^{(h)}(k) = b_j^{(h)}(k)$, where $b_j^{(h)}(k)$ is the bias applied to neuron j in the hidden layer. The output of the hidden layer is then

$$a_j^{(h)}(k) = \gamma\left(n_j^{(h)}(k)\right) \quad \text{for } j = 1, 2, \ldots, H, \tag{8.8}$$

assuming the log-sigmoid nonlinear activation function given by Equation 8.6. In the same way, the output of the final (output) layer is

$$y(k) = \sum_{i=0}^{H} w_i^{(out)}(k) a_i^{(h)}(k), \tag{8.9}$$

where $w_i^{(out)}(k)$ is the weight of the single linear neuron in the output layer fed by neuron i in the hidden layer. For $i = 0$, we have $a_0^{(h)}(k) = +1$ and $w_0^{(out)}(k) = b^{(out)}(k)$, where $b^{(out)}(k)$ is the bias applied to the neuron in the output layer. The output of the

MRANC then becomes

$$z(k) = E_{pri}(k) - y(k). \tag{8.10}$$

Through the error backpropagation algorithm the weights and biases of the 2-layer perceptron ANN may be adjusted so as to minimize the squared error J given by

$$J = z(k)^2. \tag{8.11}$$

In this case the equations derived in Chapter 4 for a 2-layer perceptron, using error backpropagation with steepest-descent are used. Equation 8.7, Equation 8.8 and Equation 8.9 along with

$$\delta^{(out)}(k) = -z(k) \tag{8.12}$$

and

$$\delta_j^{(h)}(k) = a_j^{(h)}(k)\left(1 - a_j^{(h)}(k)\right)\delta^{(out)}(k)w_j^{(out)}(k), \tag{8.13}$$

are used to give the weight update equations.

Equations 8.7, 8.8 and 8.9 represent the forward computation and Equations 8.12 and 8.13 the backward computation, where $\delta^{(out)}(k)$ and $\delta_j^{(h)}(k)$ are the local gradients for the output layer and neuron j in the hidden layer, respectively. The weights and offsets are changed according to

$$\Delta w_i^{(out)}(k) = -\alpha_i^{(out)}(k)\delta^{(out)}(k)a_i^{(h)}(k) \quad \text{for } i = 0, 1, \ldots, H \tag{8.14}$$

and

$$\Delta w_{ji}^{(h)}(k) = -\alpha_{ji}^{(h)}(k)\delta_j^{(h)}(k)a_i^{(in)}(k) \quad \begin{cases} \text{for } i = 0, 1, \ldots, N(p+1) \\ \text{for } j = 1, 2, \ldots, H \end{cases}, \tag{8.15}$$

where $\alpha_i^{(out)}(k)$ and $\alpha_{ji}^{(h)}(k)$ are the learning rate parameters assigned to weights $w_i^{(out)}(k)$ and $w_{ji}^{(h)}(k)$ respectively. To further optimise the performance of the ANN, an adaptive learning rate is used, the learning rate update rule being defined as follows [Haykin 1994]

(§6.15). For the output layer learning rates

$$\Delta\alpha_i^{(out)}(k+1) = \begin{cases} \kappa & \text{if } S_i^{(out)}(k-1)D_i^{(out)}(k) > 0 \\ -\beta\alpha_i^{(out)}(k) & \text{if } S_i^{(out)}(k-1)D_i^{(out)}(k) < 0 \\ 0 & \text{otherwise} \end{cases}, \tag{8.16}$$

where $D_i^{(out)}(k)$ and $S_i^{(out)}(k)$ are defined respectively as

$$D_i^{(out)}(k) = \delta^{(out)}(k)a_i^{(h)}(k) \tag{8.17}$$

and

$$S_i^{(out)}(k) = (1-\xi)D_i^{(out)}(k-1) + \xi S_i^{(out)}(k-1), \tag{8.18}$$

where κ, β and ξ are control parameters. Likewise, for the learning rates of the hidden layer

$$\Delta\alpha_{ji}^{(h)}(k+1) = \begin{cases} \kappa & \text{if } S_{ji}^{(h)}(k-1)D_{ji}^{(h)}(k) > 0 \\ -\beta\alpha_{ji}^{(h)}(k) & \text{if } S_{ji}^{(h)}(k-1)D_{ji}^{(h)}(k) < 0 \\ 0 & \text{otherwise} \end{cases}, \tag{8.19}$$

where

$$D_{ji}^{(h)}(k) = \delta_j^{(h)}(k)a_{ji}^{(in)}(k) \tag{8.20}$$

and

$$S_{ji}^{(h)}(k) = (1-\xi)D_{ji}^{(h)}(k-1) + \xi S_{ji}^{(h)}(k-1). \tag{8.21}$$

The learning rate adaptation procedure described in Equations 8.16 to 8.21 is known as the "delta-bar-delta" learning rule, described in Section 4.4.4.2 [Jacobs 1988], [Haykin 1994].

8.5 METHODS

A study was performed with EEGs recorded from a selection of patients to test the effectiveness of MRANC for spike enhancement.

8.5.1 Data collection

The EEG was recorded by scalp electrodes placed according to the International 10-20 system [Jasper 1958]. Sixteen channels of EEG were recorded simultaneously both for referential and bipolar montages. The amplified EEG was bandpass filtered between 0.5 and 70 Hz using a five-pole analog Butterworth filter, sampled at 200 Hz and digitized to 12 bits. All data were stored for later off-line processing.

8.5.2 Performance index

For the sake of providing a means of measuring the performance of the system, in the following the EDs are termed the signal and the background EEG is termed the unwanted noise contaminating the signal. The signal-to-noise ratio (SNR) is then defined as the ratio of the peak-to-peak value of the ED to the root-mean-square value of the background EEG for a number of samples on either side of the ED, excluding the ED itself (see Figure 8.10).

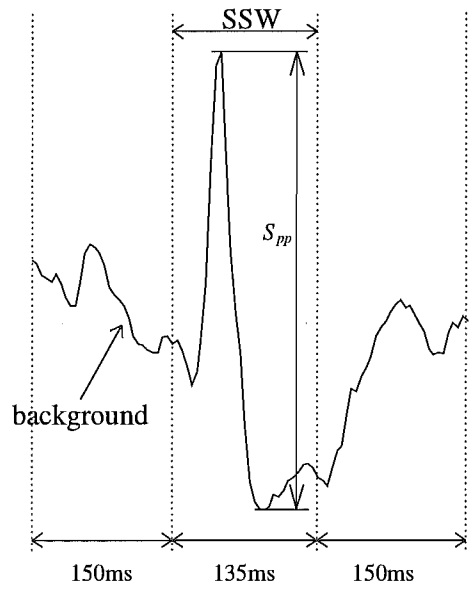


Figure 8.10 The SNR is defined as the ratio of the peak-to-peak amplitude of the ED to the RMS value of 30 samples on either side of the ED.

The ED is initially identified by the location of its maximum negative peak (the negative direction, by convention, is upwards in the EEG). The expected duration of ED's is 70–200 ms [Chatrian *et al.* 1974]; a typical duration of 135 ms is assumed, which corresponds to 27 samples at 200 samples per second. The minimum sample within the range ±14 samples from the negative peak is chosen to be the positive peak of the ED and the peak-to-peak value S_{pp} calculated accordingly. The sample midway between the negative and positive peaks is assumed to be the centre of the ED. Finally,

30 samples (150 ms) on either side of the 27-sample wide ED are chosen to describe the background EEG in the vicinity and its RMS value B_{RMS} calculated. The SNR is calculated by

$$\text{SNR} = \frac{S_{pp}}{B_{RMS}}. \tag{8.22}$$

Using this SNR calculation the primary performance index used is the percentage increase in SNR defined as

$$\Delta\text{SNR} = \frac{\text{SNR}_{new} - \text{SNR}_{old}}{\text{SNR}_{old}} \times 100\%, \tag{8.23}$$

where subscripts "old" and "new" refer to before and after filtering respectively.

8.5.3 Selection of reference channels

The channel containing the highest amplitude EDs, generally being closest to the epileptogenic focus, was made the primary channel. The reference channels were then grouped, as shown in Figure 8.11, as follows: group A comprised of the 3 channels closest to the primary channel, group B the 4 channels furthest from the primary channel and group C all channels other than the primary channel.

MRANC was then performed on each segment using each reference group singly, or (in one case) a combination of 2 reference groups. The SNR of the ED at the output of the MRANC filter was recorded and the percentage increase in SNR calculated.

8.5.4 The ANN parameters

8.5.4.1 Nonlinear implementation

The MRANC system was implemented by means of a 2-layer perceptron ANN as described in Section 8.3. Experiments were performed to determine the number of reference channels N, the number of delays to be considered for each reference channel p, and the number of neurons H in the hidden layer of the ANN.

To determine N it was necessary to determine which channels were to be used as reference channels for the MRANC process. In order to reduce the cross-talk between the primary input channel and the reference channels, it is preferable to choose reference channels as far as possible from the primary input channel. Conversely, the more distant a reference channel lies from the primary input channel, the less correlated the background EEG (or noise) becomes with the primary channel and hence the more

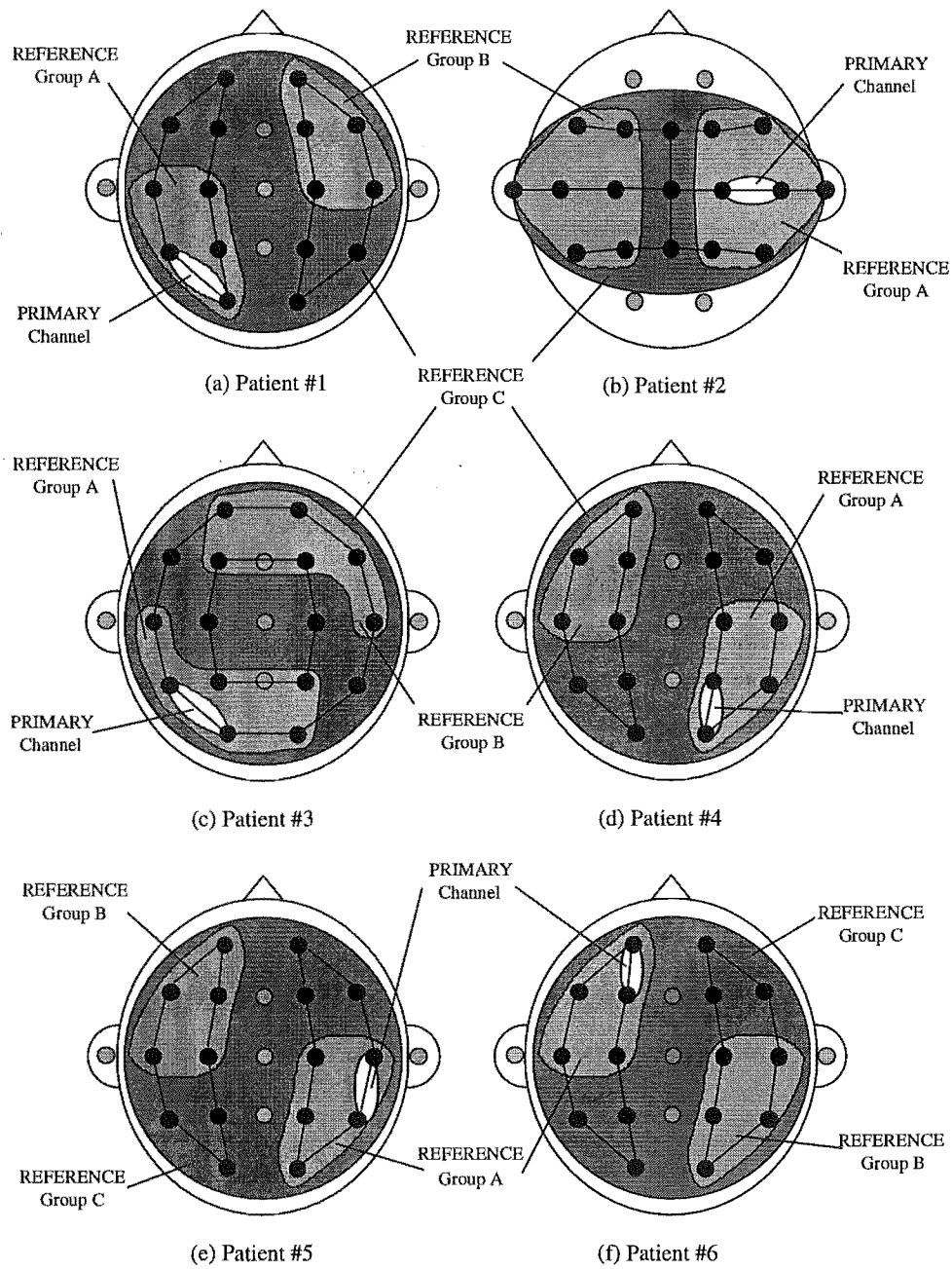


Figure 8.11 The reference groups used for each of the 6 test patients.

MRANC performance deteriorates. To determine the optimal combination of reference channels, the reference channels were put into 3 groups, N was varied for each group, and a number of tests carried out for each case. In the same manner, a number of tests were carried out with H set first at 2 and then at 5, 10 and 20. Preliminary testing indicated that system performance was optimal for $p = 2$ (i.e., no significant improvements in performance for $p > 2$) and, hence, this was used for all subsequent tests.

8.5.4.2 Linear implementation

For comparison, the percentage increase in SNR of MRANC performed on the same samples utilizing an adaptive linear combiner filter adapted by means of the LMS algorithm (as described in Section 4.3) was included. The filter consisted of a weighted linear combination of the inputs sampled at time k and the p consecutive past samples for a pth order filter. The filter consisted of a single layer (i.e., $H = 0$).

8.5.5 Subjects

The system was tested on the EEGs of six patients recorded using the procedure indicated in Section 8.5.1. Although the data recorded included referential montages, all referential montages were converted to longitudinal bipolar montage before being stored and subsequently used for testing. Segments of EEG were chosen from each patient, containing 10 EDs identified by an electroencephalographer (G.Carroll) who had access to the full multichannel EEG (i.e., could rate EDs based on spatial and temporal contextual information). To ensure that the ANN had adequately converged, care was taken that the first ED did not occur within the first 4 s of the recorded segment. The SNR of each ED was calculated in the original recording. So as to test the system on a range of different EDs, the EEGs chosen included both focal EDs and generalized EDs. Table 8.1 summarises the EEG characteristics of each patient.

Patient	Montage	Dur. (s)	ED distribution	ED classification		
				Def.	Prob.	Poss.
Patient #1	Longitudinal	24	Focal *(t5)*	3	2	5
Patient #2	Transverse	20	Focal *(t4)*	0	4	6
Patient #3	Circumferential	20	Focal *(o1)*	8	1	1
Patient #4	Longitudinal	20	Focal *(c4–p4)*	0	3	7
Patient #5	Longitudinal	20	Multifocal *(t4, c4–p4, c3)*	0	0	10
Patient #6	Longitudinal	20	Generalized	0	0	10

Table 8.1 The characteristics of the EEG segments used to test MRANC performance.

8.5.6 Autoregressive prediction

As mentioned in the introduction to this chapter, the autoregressive (AR) prediction method as described by Lopes da Silva *et al.* [1974] and Lopes da Silva *et al.* [1977] has received much attention in the literature. That method was therefore applied to the EEGs of the same six patients for comparison with MRANC.

The Lopes da Silva method is based on the assumption that an EEG segment can be described by means of a linear difference equation with constant coefficients. Thus the EEG is considered as being the output of an AR-filter having an input of white noise (normally distributed). Passing the EEG through the inverse of the estimated AR-filter should therefore result in normally distributed (white) noise, the output of the inverse AR-filter being called the prediction error. By examining the statistical properties of the prediction error, it can be said that at any point in time at which the estimated noise deviates from a normal distribution (at a certain probability level), a non-stationarity is present at the input. In this case, such a non-stationarity is taken to be an ED (although it could well be artifact, as discussed later).

This single channel approach was applied to the primary channel of each patient, after having first estimated the coefficients of the AR model by applying Durbin's algorithm (see Makhoul [1975]) to the first 800 samples (4 seconds). The order of the estimated AR-filter was set at $p = 15$, corresponding to the optimum value calculated by Lopes da Silva *et al.* [1974]. The SNR of the known EDs was measured at the output of the inverse AR-filter in the manner described in Section 8.5.2 and the percentage increase in performance calculated. Also, a detection function $d(k)$ was calculated following Lopes da Silva *et al.* [1974]

$$d(k) = \sum_{m=k-2}^{k+2} \left[\frac{\hat{e}(m)}{\hat{\sigma}}\right]^2, \tag{8.24}$$

where $\hat{e}(k)$ is the prediction error of the inverse AR-filter and $\hat{\sigma}^2$ is the variance of the prediction error. For a normally distributed $\hat{e}(k)$, $d(k)$ would have a chi-squared distribution with 5 degrees of freedom [Lopes da Silva *et al.* 1974]. A threshold D was set for $d(k)$ on the basis that $\mathrm{P}(d(k) > D) < 0.001$; from tabulated values of chi-squared distribution, $D = 20.5$.

8.5.7 Transfer function analysis

In order to estimate the transfer function of the MRANC filter, a number of MATLAB Signal Identification toolbox functions were used on the segments of EEG. Segments of EEG 200 samples wide were obtained at various points throughout the EEG recording for both the unfiltered EEG (pre-MRANC) and the MRANC filtered EEG.

In addition, the result of simple high-pass filtering each primary channel was obtained by presenting the primary EEG channel of each patient in Table 8.1 to both Butterworth and Bessel high-pass filters (HPFs). The filter order was varied between 0 and 15, whilst the cut-off frequency was varied between 1 Hz and 40 Hz. The performance in terms of SNR improvement was recorded in each case.

8.6 RESULTS

Table 8.2 lists the average performance (measured by the percentage increase in SNR) of the MRANC for each patient and group of reference channels, with the number of neurons in the hidden layer varying from 2 to 20. Results for both linear and nonlinear implementations of MRANC are listed. Table 8.3 shows the average performance over all six patients with each group of reference channels.

		Average % increase in SNR			
Patient	**Filter type**	**Group A**	**Group B**	**Groups A+B**	**Group C**
Patient #1	Linear	90.7	73.0	76.1	60.9
	Nonlinear ($H = 2$)	155.0	109.0	104.1	114.6
	Nonlinear ($H = 5$)	93.0	97.4	108.9	114.7
	Nonlinear ($H = 10$)	129.0	73.9	102.0	119.7
	Nonlinear ($H = 20$)	147.0	76.1	78.8	117.6
Patient #2	Linear	61.8	107.4	66.0	93.6
	Nonlinear ($H = 2$)	101.4	115.4	121.2	127.1
	Nonlinear ($H = 5$)	114.3	116.0	114.9	118.6
	Nonlinear ($H = 10$)	129.7	82.2	121.5	120.8
	Nonlinear ($H = 20$)	105.6	114.0	108.3	129.8
Patient #3	Linear	9.9	39.9	10.2	13.1
	Nonlinear ($H = 2$)	62.9	44.3	49.1	52.6
	Nonlinear ($H = 5$)	52.8	40.7	42.8	54.1
	Nonlinear ($H = 10$)	33.4	51.3	46.0	55.6
	Nonlinear ($H = 20$)	33.0	41.1	30.7	48.3
Patient #4	Linear	71.9	134.1	86.4	114.4
	Nonlinear ($H = 2$)	158.9	195.2	212.7	214.2
	Nonlinear ($H = 5$)	221.1	197.4	195.3	208.6
	Nonlinear ($H = 10$)	207.3	209.2	198.9	201.9
	Nonlinear ($H = 20$)	165.1	181.1	165.1	160.1
Patient #5	Linear	113.5	98.9	22.8	95.5
	Nonlinear ($H = 2$)	134.0	138.5	133.0	125.2
	Nonlinear ($H = 5$)	134.2	125.3	131.6	125.6
	Nonlinear ($H = 10$)	115.6	128.3	150.1	162.4
	Nonlinear ($H = 20$)	153.5	136.4	114.8	175.9
Patient #6	Linear	52.4	16.7	28.5	49.6
	Nonlinear ($H = 2$)	52.5	81.1	76.8	61.5
	Nonlinear ($H = 5$)	78.3	73.3	64.3	74.4
	Nonlinear ($H = 10$)	86.1	141.2	84.6	67.8
	Nonlinear ($H = 20$)	39.7	92.0	71.7	98.5

Table 8.2 Performance of linear and nonlinear MRANC ($p = 2$). The average was taken over all EDs identified in the EEG segment from each patient (at least 10).

On average, MRANC achieved an increase is SNR on all patients and for both

Filter type	Group A	Group B	Groups A+B	Group C
Linear	66.6	78.3	65.5	75.5
Nonlinear ($H = 2$)	110.9	114.9	115.9	113.0
Nonlinear ($H = 5$)	115.7	108.5	109.7	116.4
Nonlinear ($H = 10$)	117.2	114.4	116.8	121.2
Nonlinear ($H = 20$)	119.8	106.3	106.5	120.8

Table 8.3 Average % increase in SNR over the six patients.

linear and nonlinear configurations. In virtually every ED tested over the 6 patients, the nonlinear MRANC configuration resulted in a significant improvement in performance over the linear configuration. On average, increasing the number of neurons in the hidden layer above 10 resulted in no real improvement in performance. These results also show that on average as more channels are included in the reference groups, the performance increases slightly.

Of particular interest are the results for patient #6; the EDs of this patient were of a generalized nature, meaning that EDs were evident on all channels and did not have any primary focus. This means that the level of cross-talk of the signal on the primary channel to the reference channels was particularly high, and yet the system still managed to enhance the performance, albeit at a lower level than the others.

Figure 8.12 depicts a particular example of three EDs in the EEG segment of patient #1. Figure 8.12a shows the original signal with the SNR for each ED as indicated. Figure 8.12b shows the signal obtained after MRANC for a linear filter adapted by means of the LMS algorithm, whereas Figure 8.12c shows the signal obtained after MRANC for the nonlinear case. The increase in SNR achieved by both the linear and the nonlinear MRANC is indicated by the values in brackets.

Figure 8.13 shows a plot of the original SNR versus the new SNR due to nonlinear MRANC filtering (refs. group C, $H = 10$, $p = 2$) for 91 EDs drawn from the six patients with a further 11 'ED-like' background EEG artifacts also included. The EDs were classified as possibles, probables and definites by an EEGer who had access to the full EEG recordings and was thus able to use spatial and temporal contextual information to help in grading each ED. The line indicates equality between new SNR and original. From the graph it can be seen that only 5 EDs resulted in a filtered signal with SNR reduced, while over 40% resulted in an increase of SNR of greater than 100%. Those EDs marked as 'definites' by the EEGer all had a large SNR to begin with and hence are concentrated in the upper RH portion of the graph. Conversely, those EDs marked as 'possibles' are mostly concentrated in the lower LH portion of the graph. The few EDs marked as 'probables' seem to be evenly distributed over the graph. On average there was an increase in the SNR for 'possibles' whereas the background artifacts, which had a low initial SNR to begin with, resulted in a decrease in the SNR.

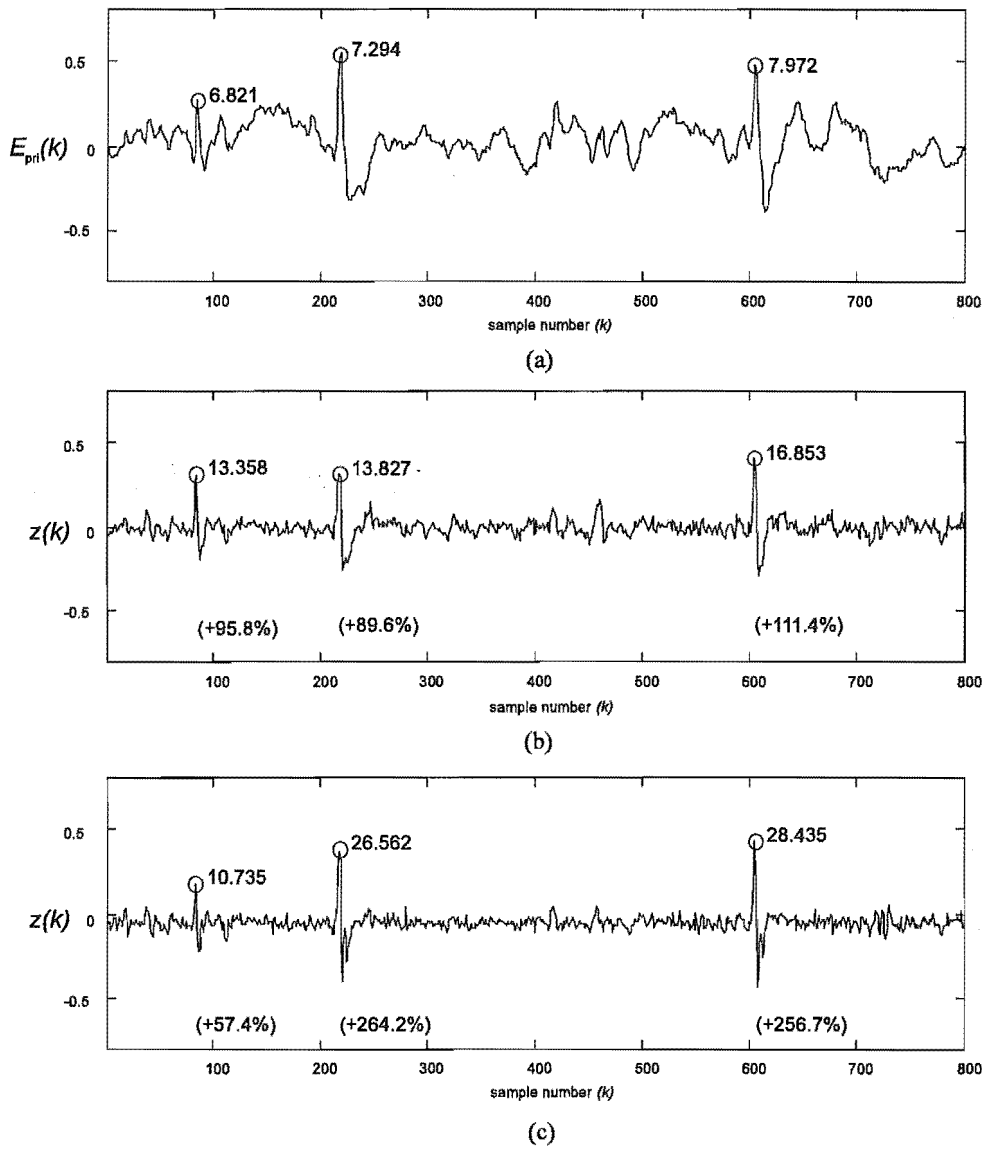


Figure 8.12 Results of processing an 800 sample (4 sec) interval of signal for Patient #1 with three EDs. Recorded with bipolar montage, primary signal recorded at *t5–o1* and references at *c3–p3*, *p3–o1* and *t3–t5*. (a) The primary input to the MRANC filter — the SNRs of EDs are indicated. (b) The output of the linear MRANC filter and (c) the nonlinear MRANC filter ($p = 2, H = 2$). The percentage increase in SNR is indicated by the values in brackets.

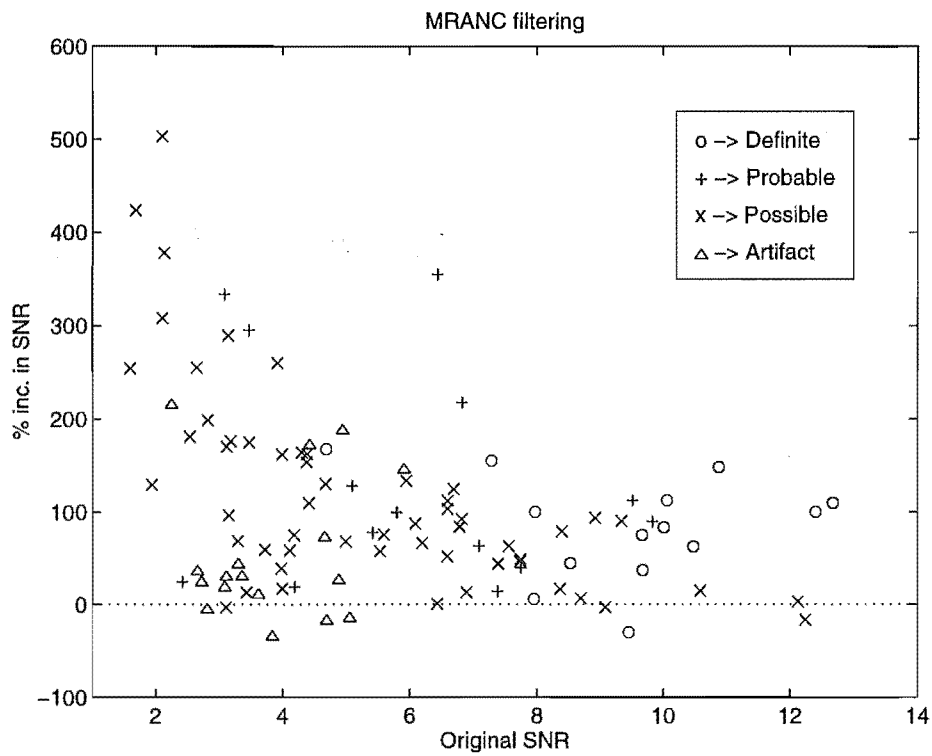


Figure 8.13 A plot of the % increase in SNR after nonlinear MRANC filtering (refs. group C, $H = 10$, $p = 2$) *vs* the original SNR for 91 EDs recorded from the six patients (31 in addition to those in Table 8.1). A further 11 'ED-like' background EEG artifacts are also included.

Table 8.4 compares the performance obtained for each patient utilizing nonlinear MRANC (using reference group C, $H = 10$ and $p = 2$) with the performance obtained by the inverse AR-filtering technique. MRANC was superior to the inverse AR-filter output by 18%.

Patient	**Nonlinear MRANC (Ref. group C, $H = 10, p = 2$)**	**Inverse AR-Filter ($p = 15$)**
Patient #1	119.7	57.5
Patient #2	120.8	83.4
Patient #3	55.6	46.2
Patient #4	201.9	220.8
Patient #5	162.4	106.6
Patient #6	67.8	105.4
Average	121.4	103.3

Table 8.4 Comparison of average % increase in SNR between nonlinear MRANC and inverse AR filtering.

Figure 8.14 depicts an EEG segment on which the following tests were performed: first, an AR-predictor model was fitted for the signal in Figure 8.14a ($p = 15$) yielding the prediction error of Figure 8.14b. Next a detection function is calculated from the prediction error with 5 degrees of freedom yielding the trace in Figure 8.14c. A threshold is set at a probability level of 0.001 for detecting the presence of transients in the original EEG segment. Next, MRANC is applied to the original (raw) EEG segment (refs. group C, $H = 10$, $p = 2$) to give the trace in Figure 8.14d. Then the whole process of estimating an AR-predictor model is repeated for the MRANC filtered EEG ($p = 15$), which then produces the prediction error given in Figure 8.14e and the detection function of Figure 8.14f. The SNR of the known ED is marked and recorded both for the inverse AR-filtered EEG and the MRANC filtered EEG, along with the percentage increase in SNR in each case.

When looking at the detection function of Figure 8.14c due to inverse AR-filtering of the 'raw' EEG, there are 5 points where the detection function exceeds the threshold leading to 4 false detections. The detection function derived from the MRANC filtered EEG shows a less 'noisy' waveform with the threshold being exceeded at only on point which coincides with the presence of the known ED leading to no false detections. The MRANC filtered EEG of Figure 8.14d leads to a 48.9% increase in SNR for the marked ED, with increases seen for both instances of the prediction error as seen in Figure 8.14b and Figure 8.14e.

The MRANC filter can be considered to converge to a HPF whose characteristics vary both at different times within the same EEG and between EEGs of different patients. Figure 8.15 shows the frequency response of the MRANC filter (with $H = 10$ and utilizing reference group C) at different instances in time through the EEG segment

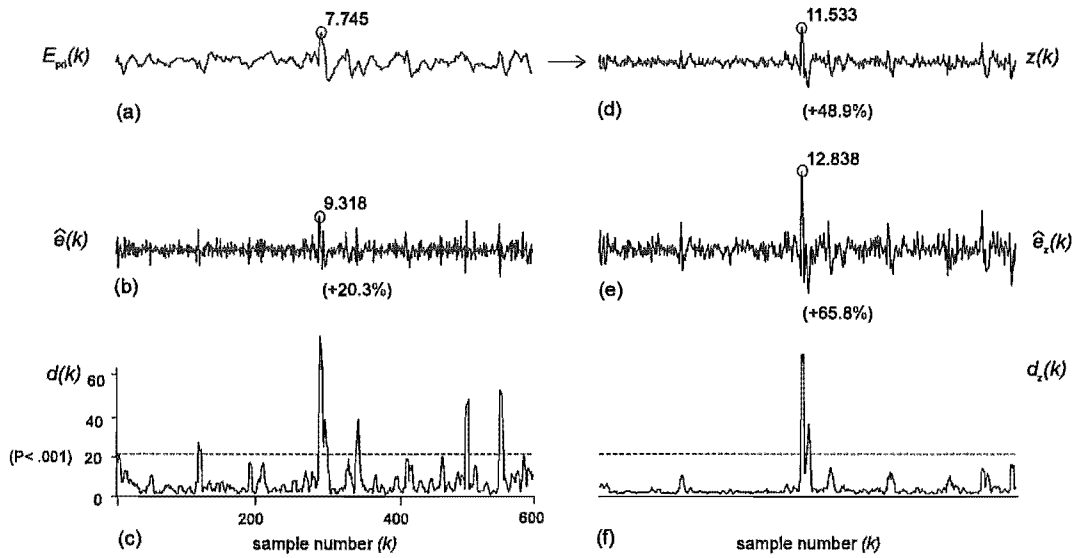


Figure 8.14 The application of nonlinear MRANC and inverse AR-filtering to a 600 sample (3 sec) segment of EEG. (a) The original EEG segment. (b) The prediction error resulting from inverse AR-filtering the 'raw' EEG ($p = 15$) and (c) the detection function calculated from the prediction error with 5 degrees of freedom. (d) The MRANC filtered version (nonlinear) of the EEG (ref. grp. C, $H = 10$, $p = 2$) and (e) the corresponding prediction error due to inverse AR-filtering the MRANC filtered EEG ($p = 15$). (f) The detection function, with 5 degrees of freedom, formed from the prediction error of (e).

of Figure 8.12. The figure highlights the HPF nature and the variability with time of the filter characteristics.

On passing the EEG segments through a Butterworth filter, an average cut-off frequency of 26 Hz was optimum for a filter of order 2. This optimum was, however, accompanied by moderate distortion of the ED shape in addition to the desired attenuation of the background EEG. Increasing the order of the filter grossly increased the distortion at almost all cut-off frequencies. At order 1, the distortion was considerably less but performance was then less than that of MRANC. Employing a Bessel filter resulted in only minimal increase in performance at lower cut-off frequencies and with marked distortion of the ED as the cut-off frequency was increased and the filter order increased above 2.

8.7 DISCUSSION

The system presented here is based on the assumption that the EEG consists of an underlying background process onto which transients are superimposed. MRANC allows these two processes to be separated without the need of prior information about the background process or the transients. An additional stage is then required to classify the enhanced transients into EDs and other events but it is considered that the classi-

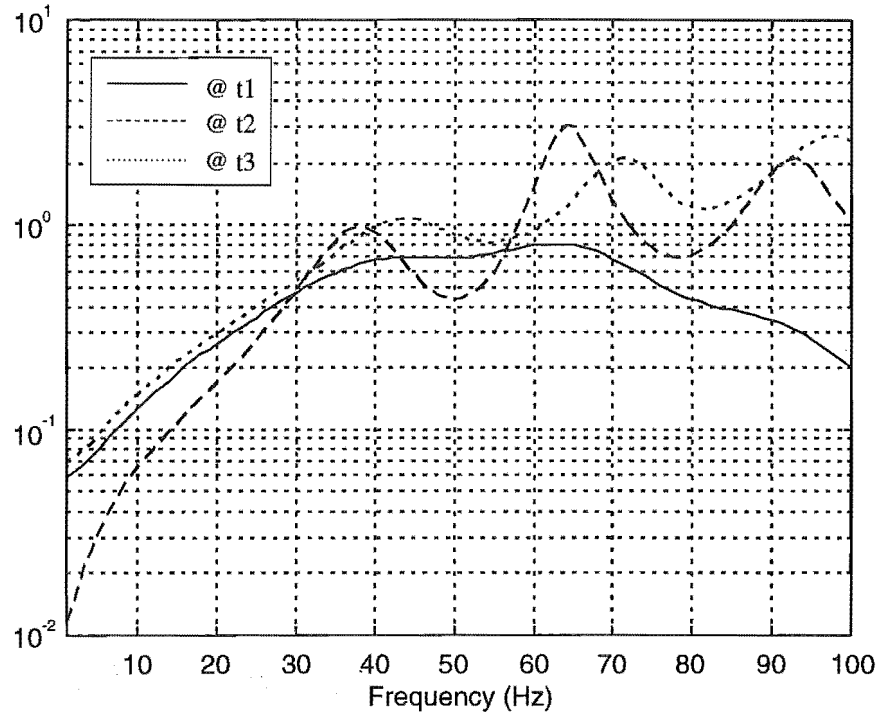


Figure 8.15 Amplitude response of nonlinear MRANC filter (reference group C, $H = 10$, $p = 2$) at times t1=500 ms after ED #1, t2=900 ms after ED #2 & t3=200 ms after ED #3 (see Figure 8.12).

fication process will be more accurate by the previous attenuation of the background EEG. Implementing MRANC by means of an ANN has been shown to yield considerably better results than a comparable linear implementation (LMS). This lends weight to the suggestion that the relationship between the signal source and recorded signals may be nonlinear (cf. Section 8.1). The adaptive nature of the MRANC filter also allows for variations in the background EEGs of different patients, as well as variations within an EEG, to be accommodated.

The presence of signal cross-talk between the primary and reference channels is a significant factor affecting the performance of MRANC. In the EEG case, maximum cross-talk occurs in the case of generalized ED distribution as, for example, in the case of patient #6. Nevertheless, although MRANC did not perform as well for patient #6 as for the other patients, a substantial improvement in SNR was still achieved. Signal cross-talk does, however, introduce distortion in the primary signal resulting in EDs with changed morphologies. In the tests described here no measure of distortion was calculated although conceptually this could be tackled by calculating a value for the normalized mean squared difference between the portion of the original signal representing the ED and the corresponding MRANC filtered signal.

Initially, it was thought that the choice of reference channels would prove the most important factor in the application of MRANC to the EEG. For the most part, this turned out not to be the case. Although increases in performance were seen as

more reference channels were added, these were slight. Tests conducted using the different reference groupings indicate no clear justification for choosing one grouping over another. The performance was marginally, but not consistently, better when the reference channels consist of a balance of channels neighbouring the primary channel and the channels furthest away from the primary channel (groups A+B). Presumably this works due to the balance obtained by the neighbouring reference channels, which have background EEG highly correlated with the primary but a high level of cross-talk, and the remote reference channels, which have a less correlated background EEG but even less cross-talk. However, designating all channels other than the primary channel as reference channels (i.e., reference group C) confers a practical advantage in that it eliminates the need of arbitrary selection of reference groups dependent on primary channel and montage for a particular EEG segment.

The performance of the ANN implemented MRANC was particularly impressive on those EDs with a small initial SNR (cf. Figure 8.13), meaning that their presence in the background EEG was not that well defined. It is believed that this enhancement of the EDs with small initial SNR will contribute in increasing the selectivity of the overall spike detection system.

Overall, MRANC (with $H = 10$, utilizing reference group C) performs better than the inverse AR-filtering method. The fundamental difference between the two methods is that the inverse AR-filtering method utilises purely temporal information and relies on the non-stationary properties of EDs to enhance their presence in the otherwise stationary background EEG. In contrast, MRANC utilises spatial as well as temporal information (but particularly the former) to enhance the EDs at the primary channel, with no prior knowledge of 'signal' or 'noise' characteristics required. Also, only MRANC (as implemented) is adaptive, allowing the filter characteristics to change with time as the background EEG is not strictly stationary.

Having established the essential differences between the MRANC and AR methods, it is reasonable to conjecture whether there may be an advantage in combining the two. The results presented in Figure 8.14 suggest that the combination may be useful. MRANC has the ability to improve the SNR of EDs in the EEG resulting primarily from their spatial distribution. AR filtering, on the other hand, is successful at detecting purely temporal variations in a single signal, provided it has an estimate of the underlying stationary process. The improvement due to MRANC can also be seen when forming the detection function mentioned earlier, both for the 'raw' EEG segment and for the MRANC filtered EEG segment. The enhancing of EDs by MRANC means that the detection functions result in more 'clear-cut' decisions as to the detection of an ED at a given probability level and thus reducing the possibilities of numerous false detections.

Examination of the MRANC's spectral characteristics and of the performance of

'equivalent' fixed HPFs has confirmed that MRANC comes to represent a HPF. The HPF is, however, one which has been optimized for a specific patient or EEG and, due to its adaptive nature, one whose characteristics change in time to optimally accommodate changes *within* an EEG.

8.8 SUMMARY

In this chapter the possibility of enhancing the sensitivity of a spike detection system through the use of MRANC has been discussed. This is done by enhancing the presence of spikes in the EEG by adaptively cancelling the surrounding background EEG. MRANC has been implemented for both linear and nonlinear ANNs in order to measure the effect of the noise cancelling process in each case. Nonlinear MRANC always performed much better than its linear counterpart. Comparing MRANC to AR-modeling techniques shows that MRANC results in improved performance. Although MRANC also enhances artifacts, it does so to a lesser extent than for EDs. The MRANC technique results in a HPF which is capable of changing its characteristics with time, thus following any changes in the EEG characteristics in a given patient and will adapt to the 'optimum' filter settings for each new EEG analysed. Although the data of only six patients have been analyzed, it is evident that MRANC does indeed enhance the presence of focal activity in the EEG and the use of nonlinear ANNs in the application of MRANC further enhances the process.

Chapter 9

SPIKE DETECTION WITH THE SELF-ORGANISING FEATURE MAP

9.1 INTRODUCTION

The self-organising feature map has already been discussed in detail in Chapters 5 and 6. This chapter deals with applying the SOFM to the spike detection problem. After first introducing the proposed scheme, the chapter provides further insight into the two stages that make up this part of the spike detection system. The *mimetic* stage screens the incoming EEG before presenting candidate waveforms to a trained SOFM stage. The methods used for training and testing the system are then discussed along with results obtained after the testing process.

The unsupervised learning methods used by the SOFM for training are used extensively to form a realistic representation of the problem. This is done by using large numbers of unlabelled inputs followed by calibration using a small quantity of labelled inputs. However, spikes in the interictal EEG are nearly always considerably less abundant than non-epileptiform counterparts in the background EEG. Also spike characteristics vary from one patient to the next and within the same EEG recording. Further compounding the problem is the fact that spikes can be very easily confused with artifact in many cases. This all means that putting together a subset of spike 'exemplars' is no easy task. The fact that the SOFM classifier only needs a *small* subset of the training data to be labelled means that only those spikes which EEGers label with a high level of confidence can be used for calibration, any others are used for training only without any need for a class label.

9.2 THE PROPOSED SYSTEM

The proposed system is depicted in Figure 9.1. The outputs from this stage are positive real numbers within the range 0 to 1 giving the probability that there is an ED on each particular channel (16 channels total). The final stage uses these values to indicate the

presence of an epileptiform *event* (EV) across *all* 16 channels of EEG. The final stage uses spatial clues obtained from the outputs of the first stage and is described in more detail in the next chapter.

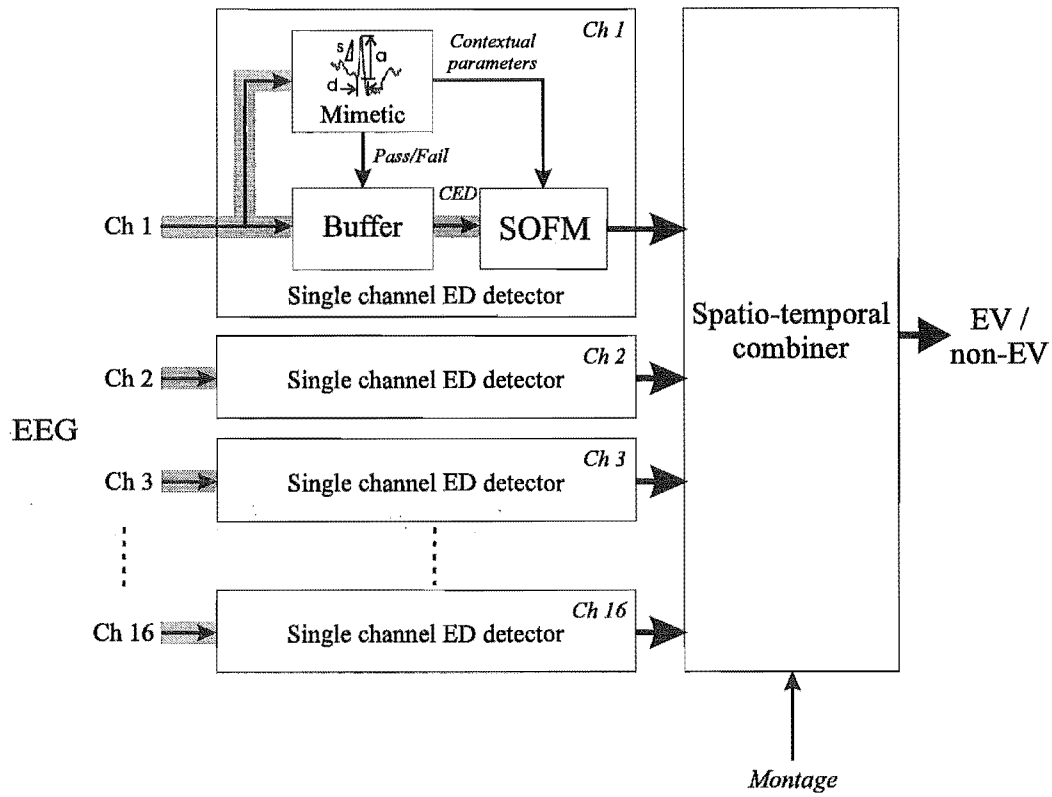


Figure 9.1 The proposed multi-stage spike detection system.

9.3 DATA COLLECTION

The EEG was recorded by scalp electrodes placed according to the International 10-20 system [Jasper 1958]. Sixteen channels of EEG were recorded simultaneously for bipolar montages only. The amplified EEG was bandpass filtered between 0.5 and 70 Hz using a five-pole analogue Butterworth filter, sampled at 200 Hz and digitized to 12 bits. All data were stored for later off-line processing.

Four types of bipolar montages were used which were: longitudinal, transverse, longitudinal-transverse and circumferential (more information on these montages can be obtained in Appendix A). All EEG recordings were made while the patient was awake but resting and included periods of eyes-open, eyes-closed, photic stimulation and hyperventilation. Three recording protocols were used, depending on the age of patients which were: (a) BABY, (b) UNDER5 and (c) OVER5 (more information on the recording protocols can be found in Appendix A). Although some of the record-

ing protocols include referential montages, all referential montages were converted to bipolar longitudinal before recordings were stored.

9.4 THE MIMETIC STAGE

The purpose of the mimetic stage is to screen the incoming EEG across all channels and perform a data reduction exercise by presenting the SOFM with candidate epileptiform discharges (CEDs) which have passed the screening process. As the purpose of this stage is for data reduction, it is imperative that the mimetic stage detect as many of the EDs as possible whilst rejecting most 'obvious' non-ED waveforms. The design is based on the mimetic system derived by Dingle [1992], Gotman and Gloor [1976] and Gotman *et al.* [1978].

Since it is generally accepted that EDs are defined due to their being "... clearly visible from the surrounding background EEG ..." [Chatrian *et al.* 1974], the parameters that must be examined for a CED in the incoming EEG must include parameters which describe the state of adjacent background activity. A CED is obtained by first detecting a vertex in the EEG followed by calculations of a number of parameters around this vertex and measures of adjacent background activity (i.e., contextual parameters). All parameters are then thresholded and all waves (a wave is centred around each vertex) which exceed the thresholds are considered. The mimetic process can be broken down into a number of steps as described in the following subsections.

9.4.1 Scaling the incoming EEG

The EEG amplitude varies greatly between individuals and is particularly related to the individual's age. For example, the EEGs of babies are of a higher amplitude than adults. The amplitude of the recorded EEG can be adjusted by varying amplifier gain settings at record time by the recording technician but this makes for a very subjective process. For the mimetic stage to work well it is imperative that the incoming EEG be objectively normalized. In this system the *global EEG amplitude* is estimated from the first 60 s of multichannel EEG recording. Care is taken to automatically eliminate artifacts (EMG mostly) which would otherwise result in a poor estimate of the global amplitude. This global amplitude is used to scale the RMS amplitude of the EEG to an amplitude of 8.4 μV (the average RMS amplitude calculated for 10 EEGs by Dingle [1992]). Artifacts (such as EMG) are automatically detected when the average amplitude over 1.0 s on any channel is more than 2.5 times the 60 s average on that channel. If an artifact is detected in such a manner on one channel, then all the data across *all* channels is removed and the global amplitude recalculated.

9.4.2 Locating the primary vertex

Once a gain value is calculated and the incoming EEG is normalized, all channels are scanned for a positive or negative vertex using a simple peak detection algorithm. Once a vertex is found, a number of parameters are extracted from around the vertex. The parameters extracted relate to the amplitude, duration, slope and sharpness of the waveform under consideration (see Figure 9.2). These parameters are considered to be the most important parameters to consider when distinguishing between EDs and non-EDs ([Ktonas and Smith 1974], [Gotman and Gloor 1976], [Gotman *et al.* 1978], [Gotman 1980], [Dingle 1992], [Webber *et al.* 1994]).

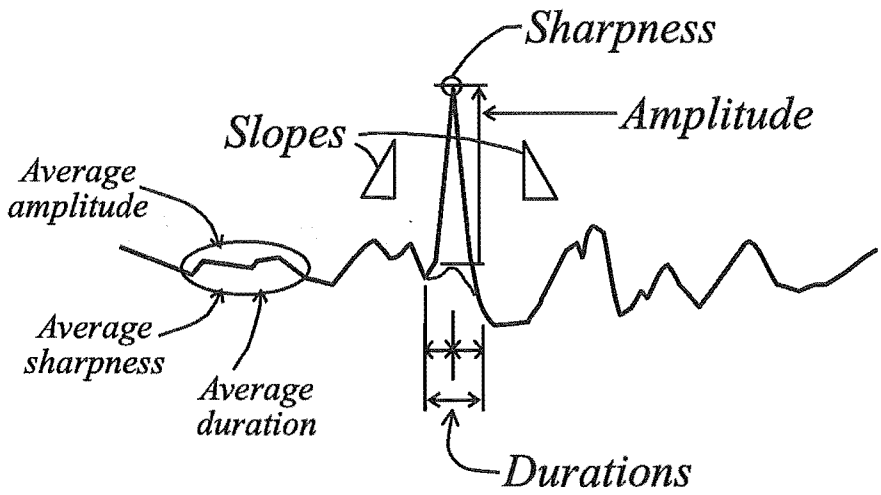


Figure 9.2 The parameters extracted by the mimetic stage.

Amplitude (A_p)**:** This is defined as the difference between the peak value and the floating mean, where the floating mean is calculated as the average value of the EEG over 75 ms centred on the peak. The reason for the floating mean, as opposed to a fixed baseline, is so that the amplitude measured is due to the CED alone and not due to the CED being superimposed on, say, a large amplitude slow wave.

Durations (D_1, D_2, D_3)**:** These are measured such that D_1 and D_2 represent the durations of each half-wave before and after the vertex. D_3 represents the sum of D_1 and D_2. The half-wave durations are measured from the vertex to the point where either slope changes rapidly. This means that the duration is measured at the point where there is more than a 60% drop in slope or a change in direction of the slope.

Slopes (S_1, S_2)**:** Slope measurements are obtained from the pre-vertex and post-vertex slopes. For waveforms of short durations (i.e., $D_1 < 20$ ms or $D_2 < 20$ ms) the

peak-to-peak slope is calculated, otherwise a least-squared estimate is obtained based on 4 samples (excluding the peak sample).

Sharpness (S_p): The sharpness of a wave is defined as the sum of the magnitudes of the pre-vertex and post-vertex slopes.

Once the parameters are extracted they are passed through a set of thresholds and once a waveform is found which exceeds all the thresholds it is flagged as *passed thresholds* (all others being flagged *failed thresholds*). The values used for the thresholds are given in Table 9.1 and are based on values determined by discriminant analysis by Dingle [1992]. In this case only one set of thresholds are used (as opposed to the two sets defined by Dingle [1992]) this is because here the thresholds have been "loosened" in order for the mimetic stage to be more sensitive to EDs.

	Thresholds	
Parameter	Min	Max
A_p	16.8 μV	-
D_1, D_2	10 ms	150 ms
D_3	20 ms	250 ms
S_p	1.26 μV/ms	-

Table 9.1 The thresholds used by the mimetic stage for screening the incoming EEG (after the EEG has been scaled to an RMS amplitude of 8.4 μV).

The mimetic stage acts on all channels independently, however, once a waveform is found which exceeds all thresholds the vertex of that waveform is called the *primary vertex*. At that point all CEDs on the remaining channels (within 50 ms of the primary vertex) are grouped so that together they make up a candidate epileptiform event (CEV) which is passed on to the next stage.

In keeping with the definition that an ED must be clearly distinguishable from surrounding background EEG, a number of contextual parameters are extracted from the 1.0 s segment of EEG about the vertex of each CED. The values extracted are as follows:

Average amplitude ($\hat{A}$): This is calculated as the RMS difference between the actual EEG and a floating mean calculated over a 15 sample (75 ms) window.

Average duration ($\hat{D}$): The average peak-to-peak duration of the half-waves (half-waves with a peak ampitude of less than 4.2 μV are ignored).

Average slope ($\hat{S}$): The average magnitude of the slope between consecutive samples.

The measure of precisely 1.0 s of background EEG was chosen such that the presence of an ED does not dominate the measurements. On the other hand, a segment of a longer duration would possibly allow bursts of EMG or α-waves to dominate.

Once the primary vertex has been found, and the contextual information extracted for each CED, the following information is put forward to the next stage for each channel of EEG:

1. **CED:** The CED for each channel is made up of a window of 'raw' EEG. As EDs are said to vary in duration from 70 – 200 ms, the ideal inputs to the system would consist of a window of at least 200 ms of 'raw' EEG – with data sampled at 200 Hz this translates to 40 samples of EEG. An important characteristic of an ED is the slow wave that generally follows the vertex, for a 200 ms waveform the vertex occurs roughly during the first third of the waveform and the slow wave (if present) in the remaining two thirds.

   Thus, a 41 sample window of EEG is used leading to a 205 ms window such that the maximum vertex is placed at the 14^{th} sample across the window (at approximately one third the way across) as shown in Figure 9.3.

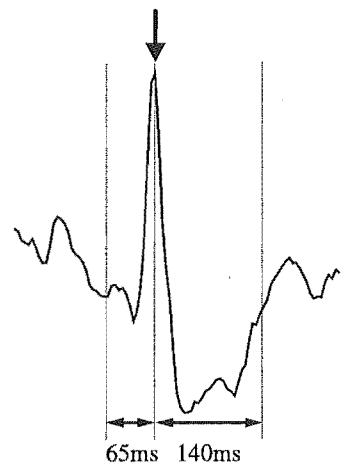


Figure 9.3 Windowing the ED. The maximum positive (or negative) vertex is placed one third of the way across the 205 ms window.

2. **Contextual information:** The three contextual parameters ($\hat{A}, \hat{D}$ and $\hat{S}$) for each CED.

3. **Passed/Failed thresholds flag:** A flag is set for a CED which passes the thresholds and reset otherwise.

9.5 THE SOFM STAGE

This stage consists of a SOFM which is trained, calibrated and then further fine-tuned using LVQ techniques as described in Chapters 5 and 6. The input to the SOFM is a vector made up of the 'raw' EEG CED, contextual parameters and the passed/failed thresholds flag output by the mimetic stage. The trained and calibrated SOFM in turn outputs a single value between 0 and 1 which represents the probability of there being an ED at that channel. The SOFM is duplicated for all channels of the EEG such that the overall output of the SOFM stage is a set of 16 probabilities – to this point there is no interaction between channels (other than in mimetic stage where are all channels are forwarded on finding a primary vertex/CED).

The SOFM was trained using a large set of epileptiform EEG data. A subset of the EEG data was used for calibration of the SOFM where the probability value of being a true ED was assigned to each neuron according to a novel system involving Bayesian statistics. LVQ2 was performed on the trained SOFM for 'fine-tuning' purposes.

9.5.1 EEG data

The EEGs of 35 patients were obtained. All EEGs had been previously seen independently by at least two EEGers (in some cases 3) and been graded as containing definite epileptiform events. The average EEG length was 24.4 minutes and the ages of the patients varied from 7 months to 71 years old (average age of 19). Table 9.2 shows the characteristics of the data used which totals over 14 hours of 16-channel EEG data. More than 2,585 definite EVs and 511 questionable EVs were identified by the EEGers.

The data also included a variety of background activities (e.g., alpha, delta, etc.) and most EEGs contained significant amounts of artifact, particularly: eye-blinks, electrode movement, and muscle artifact. Artifacts were especially prominent during the periods of hyperventilation and photic stimulation. All the data recorded was for routine clinical use and no segments of EEG were rejected because of excessive artifact or 'noisy' background activity.

The data was then passed through the mimetic stage and a large number of corresponding CEDs were collected. Considerably more CEDs failed the thresholds than passed (75% vs 25% respectively). Assuming that the values of the thresholds were optimally set such that *no* EDs were missed, then the CEDs which passed the thresholds contained the true EDs and well as many more false EDs. This means that of the large number of CEDs available for training the SOFM only a small minority had the potential of being true ED waveforms. If this was the case then only a small portion of the SOFM would be representative of true EDs after training was complete - the actual size of the corresponding portion of SOFM would depend on the relevant proportions of ED to non-ED in the CED data. In an effort to balance the data in favour of ED

EEG	Age*	Duration	EEGers	Epileptiform Events (EVs)		
				Definite	Quest.	Total
1	9	27m 7s	2	102	11	113
2	30	25m 2s	2	>200	0	>200
3	49	25m 21s	2	1	7	8
4	2	15m 42s	2	>200	0	>200
5	4	15m 26s	2	49	0	49
6	58	25m 46s	2	0	4	4
7	16	26m 27s	2	57	41	98
8	20	33m 20s	2	1	4	5
9	67	27m 50s	2	>200	0	>200
10	6	26m 8s	2	2	7	9
11	5	26m 24s	2	201	0	201
12	25	26m 34s	2	0	9	9
13	6	28m 40s	2	2	0	2
14	11	34m 2s	2	230	0	230
15	7m	14m 8s	3	0	17	17
16	15	29m 12s	2	246	24	270
17	15	22m 24s	2	28	159	187
18	7	21m 4s	2	3	12	15
19	7	20m 25s	2	>200	0	>200
20	18	25m 51s	2	52	3	55
21	13	23m 52s	2	>200	0	>200
22	29	27m 10s	2	0	10	10
23	9	23m 52s	2	0	11	11
24	16	25m 57s	2	1	4	5
25	46	26m 15s	2	333	4	337
26	17	25m 42s	2	95	0	95
27	71	18m 49s	2	17	40	57
28	12	31m 25s	2	8	5	13
29	8	26m 43s	3	3	32	35
30	7	25m 11s	2	7	41	48
31	4	14m 58s	2	12	17	29
32	10m	12m 12s	2	13	32	45
33	32	24m 55s	2	6	4	10
34	12	26m 19s	3	61	9	70
35	17	25m 26s	2	55	4	59
Totals		14h 15m 39s		>2,585	511	>3,096

Table 9.2 The training set comprising 35 EEGs with in excess of 2,585 definite EVs and 511 Questionable EVs. EEGs which contained '>200 EVs' had excessive amounts of EVs which were not individually graded by the EEGers. (*m=month).

EEG	Age	Duration	EEGers	Epileptiform Events (EVs)		
				Definite	Quest.	Total
27	71	18m 49s	2	17	40	57
28	12	31m 25s	2	8	5	13
29	8	26m 43s	3	3	32	35
30	7	25m 11s	2	7	41	48
31	4	14m 58s	2	12	17	29
32	10m	12m 12s	2	13	32	45
33	32	24m 55s	2	6	4	10
34	12	26m 19s	3	61	9	70
35	17	25m 26s	2	55	4	59
Totals		3h 25m 58s		182	184	366

Table 9.3 The calibration set comprising 9 EEGs with a total of 182 definite EVs and 184 questionable EVs.

data, a number of *failed threshold* CEDs were removed from the training set such that the number of passed threshold CEDs was twice that of the failed thresholds CEDs.

Another method of obtaining equal representation in the SOFM output for data which is under-represented in the training set is to use the so-called *conscience* method. This method involves keeping track of the *frequency* with which each neuron wins during training. Once the number of 'wins' of a neuron exceeds a certain threshold that neuron is taken out of the competition, allowing other neurons to win instead — hence giving the neurons a 'conscience'. The conscience method is described in more detail in DeSieno [1988], and was used successfully by Roberts and Tarassenko [1992] in analysing sleep EEG. However preliminary testing on the spike detection problem indicated that the conscience method is only partially successful when the imbalance between classes is relatively small (say, 4:1) and has no effect when the imbalance is greater – as is the case for the CED data in the training set. The conscience method was not used further in this study.

From the training set, 9 EEGs were chosen for calibrating the SOFM. These EEGs had an average length of 22.9 minutes and were recorded from both children and adults (Table 9.3 shows the calibration data). The EEGers graded 182 definite EVs and 184 questionable EVs. In this case EEGers graded individual *events* (i.e., spanning at least 2 channels or more) as epileptiform or not. Where a graded event coincided with any CED (as found on any channel by the mimetic stage) which passed all the thresholds, the CED was labelled epileptiform, otherwise it was labelled non-epileptiform.

As the calibration data was graded by more than one EEGer, the final grading assigned to each waveform was based upon a consensus amongst the 2 (sometimes 3) EEGers. Table 9.4 shows the final label assigned to waveforms after considering the various possible combinations of labels assigned by different EEGers. Waveforms which had widely varying labels amongst the EEGers were removed from the calibration set (but still remained part of the training set). This resulted in a calibration set made up

of 5,846 CEDs, 4,574 of which passed the thresholds and 1,272 of which failed. 1,333 CEDs were assigned true-ED labels (based on the consensus discussed above) and 4,513 were assigned non-ED (200 were rejected from the calibration set due to large disagreement amongst EEGers). The calibration set data was then used to calibrate the SOFM after training was complete. The same data was also used for fine-tuning using the LVQ2 algorithm.

Final grading		
D	**Q**	**N**
DDD	DQQ	NNN
DDQ	DQN*	NN
DDN*	DNN*	
DD	QQQ	
	QQN	
	QNN*	
	DQ	
	DN*	
	QQ	
	QN*	

Table 9.4 The final grading of events after grading by 2 or 3 EEGers. D represents definite EDs, Q questionable EDs and N non-EDs. The combinations marked with an asterisk are rejected when setting up the calibration set.

9.5.2 Training the SOFM

The SOFM was trained by presenting the training set data in random order to the SOFM whilst adapting according to the SOFM training algorithm as described in Chapter 5. The SOFM training parameters were set as shown in Table 9.5 (for further explanation see Chapter 6). The SOFM size was varied from a [10 × 10] to a [20 × 20] SOFM in order to assess the performance of the system as a function of the SOFM size.

Parameter		Value
SOFM size	$S \times S$	$S = 10, 12, 14, 16, 18, 20$
Initial learning rate	α_o	1.00
Final learning rate	$\alpha_{\min}$	0.01
Initial neighbourhood size	N_o	$S - 1$
Final neighbourhood size	$N_{\min}$	1
Neighbourhood taper	-	Quadratic taper
Learning rate, Neighbourhood } size decay	-	Linear

Table 9.5 The parameters used to train the SOFMs. Multiple values of SOFM size are used in order to assess the performance of the system as a function of the SOFM size.

The training data was presented randomly to the SOFM during training in such a way that *all* the data had been presented at least *twice* during the crucial ordering stage. The ordering stage was set at $\frac{1}{8}^{\text{th}}$ of the complete training period (i.e., $\omega = 0.125$), which means that the entire training set was presented 16 times to the SOFM during training.

9.5.3 Calibrating the SOFM

Once training was complete, the trained SOFM needed to be calibrated. During calibration a label was assigned to each neuron in the SOFM according to which category or class that neuron (or rather its corresponding weight vector $\mathbf{m}_i$) best represented.

A requirement is that once a CED (along with contextual information and a pass/fail flag) is presented to the SOFM *after* training is complete, the SOFM responds with a value indicating the *probability* to true ED. This meant that, as a result of calibration, a probability level needed to be assigned to each neuron. In operation, once a CED is presented to the SOFM, the probability assigned to the 'winning' neuron (i.e., the neuron with the closest matching $\mathbf{m}_i$) is taken to be the probability of the input CED being a true ED.

9.5.3.1 Labelling by maximum voting

The method suggested by [Kohonen 1990] for labelling the neurons during calibration is based upon a majority voting scheme and is not intended to be used to indicate probabilities, as was required in this case. Kohonen's maximum voting scheme of assigning labels is described next.

If, for a two class system, $\mathbf{m}_c$ is the weight vector of the winning neuron, then c is the index of the winning neuron. If the class of the input happened to be class 'a' then the count for class 'a' is incremented for neuron c. This is repeated for all the neurons in the SOFM while all the data in the calibration set is presented to the SOFM. Once the entire calibration set has been presented to the SOFM, the final class label assigned to each neuron is simply based upon the class count with the greatest number of 'hits', that is, for neuron i

$$\text{assign class 'a'} \quad \text{if } s_i^a > s_i^b, \tag{9.1}$$

$$\text{assign class 'b'} \quad \text{if } s_i^a < s_i^b, \tag{9.2}$$

where s_i^a represents the number of times neuron i identified class label 'a'. If $s_i^a = s_i^b$ then neuron i is considered 'undecided' about which class to represent. Depending on the particular application, it might be enough to default to a given class label or

else it might be worth assigning no label to such a neuron and removing it (and its corresponding weight vector) from the competition entirely.

9.5.3.2 The Bayesian approach to labelling

The drawback with the majority voting scheme is that the class labels are not assigned in a *probabilistic* manner. It is possible to assign *confidence levels* to each neuron (as seen in Chapter 6) based on the *success rate* for each neuron, given by

$$\theta_i^a = \frac{s_i^a}{n_i} \tag{9.3}$$

where θ_i^a is the success rate of neuron i for class a, s_i^a is the number of times neuron i identified class label 'a' and n_i is the total number of times neuron i was the 'winner'. However, this is not a true probabilty value.

For the EEG case, each neuron is required to represent a probability value ranging from 0.0 (non-ED) to 1.0 (ED). In this case the success rate for neuron i is given by

$$\theta_i = \frac{s_i}{n_i}, \tag{9.4}$$

where the number of successes (s_i) here implies the number of times input waveforms were labelled ED (as opposed to non-ED) by the EEGers.

Using the Bayesian statistical approach, it is possible to assign a probability to each neuron. The probability obtained for each neuron using this approach results from a weighted mix of *prior* probabilities and the data itself, the so-called *posterior* probability [Schmitt 1969], [Bernardo and Smith 1994].

The *prior* distribution represents our uncertain knowledge of the success rate θ_i for an individual neuron. As no *a priori* knowledge of the success rate is assumed, the prior distribution is 'flat', meaning that any success rate can be applied to each neuron. This lack of prior knowledge can be represented by a Beta probability density function which is described by

$$\pi(\theta) \sim \mathrm{Be}(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \tag{9.5}$$

where the prior distribution of the success rate θ is given by $\pi(\theta)$, Γ represents the gamma function and for a uniform prior distribution the parameters α & $\beta = 1$. Next, a likelihood function is needed which describes the probability of an input CED being 'successfully' graded ED by an EEGer given the success rate over n trials. The data is Binomial so the likelihood function for the data (conditional on the success rate and

the number of trials) is given by the Binomial probability density function

$$f(s|\theta) \sim \text{Bi}(s|\theta, n) = {}^{n}C_{s}\theta^{s}(1-\theta)^{n-s}. \tag{9.6}$$

Bayes' Theorem states that the *posterior* probability distribution of the success rate given the data is given by

$$\pi(\theta|s) = \frac{f(s|\theta)\pi(\theta)}{f(s)}. \tag{9.7}$$

This means that the posterior probability is proportional to a weighted mix of the likelihood function for the data and the prior probabilities. Applying Bayes' Theorem (equation 9.7) to the prior (Beta) distribution given by Equation 9.5 and the Binomial likelihood function of Equation 9.6 and simplifying, results in the following posterior distribution

$$\pi(\theta|s) = \frac{\Gamma(n+\alpha+\beta)}{\Gamma(s+\alpha)\Gamma(n+\beta-s)}\theta^{s+\alpha-1}(1-\theta)^{n-s+\beta-1}, \tag{9.8}$$

which is a Beta posterior distribution given by

$$\pi(\theta|s) \sim \text{Be}(\theta|\alpha+s, \beta+n+s). \tag{9.9}$$

For a uniform prior distribution (assumed) $\alpha = \beta = 1$ and so the Beta prior distribution is reduced to

$$\pi(\theta|s) \sim \text{Be}(\theta|s+1, n+s+1). \tag{9.10}$$

Taking the mean value of the distribution given by Equation 9.10 gives an estimate of the success rate θ. The mean for a Beta distribution, (with uniform prior distribution) is given by

$$E[\theta|s] = \frac{s+1}{n+2}, \tag{9.11}$$

for s successes in n trials. This means that using the Bayesian approach a better estimate of the probability of a CED being a true ED for neuron i can be found by

$$\psi_i = \frac{s_i+1}{n_i+2}. \tag{9.12}$$

Consider, for example, two neurons i and j. If neuron i was declared the 'winner' 5 times, 4 of which were for true ED waveforms, then this gives a success rate of 0.8 but a probability of 0.71 using the Bayesian probabilities described above. If neuron j was declared the 'winner' 50 times, 40 times out of which for true ED waveforms, then the success rate is 0.8 (as for neuron i) but the probability now becomes 0.79, reflecting the larger number 'wins' for neuron j. In a similar manner, if neuron i was declared winner for true EDs all 5 times and neuron j declared winner for true EDs all 50 times, both would have a success rate of 1.0. However, with the Bayesian probabilities, neuron i would be assigned a probability of 0.86 whereas neuron j a probability of 0.98.

9.6 PERFORMANCE INDICES

The ultimate measure of success in the spike detection system is how many EVs the system detects. There is however always the possibility of false detections of EVs, as well as missed EVs. The performance measures adopted in the spike detector case are the sensitivity and selectivity of the system to EVs. The sensitivity is essentially a measure of how many events are *missed* by the system. Whereas the selectivity is a measure of how many 'extra' events were picked up (i.e., how many events were detected which were *not* marked epileptiform by the EEGer(s)). As in Chapter 6, the sensitivity and selectivity are given by

$$\text{sensitivity} = \frac{\text{correct detections}}{\text{total number of true events}} \times 100\%, \tag{9.13}$$

and

$$\text{selectivity} = \frac{\text{correct detections}}{\text{total number of detections}} \times 100\%. \tag{9.14}$$

9.7 MULTICHANNEL ANALYSIS

As described in Section 9.5 the SOFM stage consists of 16 *identically* trained SOFMs each which assign a probability of being a true ED to each waveform as presented to it by the mimetic stage. This means that the probabilities assigned by the SOFMs are related to *single channel data.* In contrast, when an EEGer 'grades' an EEG the EEGer invariably marks *events* which may appear across more than one channel – as opposed to marking each individual discharge that appears on each channel. For the sake of analysing the performance of the SOFM stage without the final stage (which is described in the next chapter), an epileptiform event (EV) is considered to be detected if the probability on *any* channel exceeds a certain threshold level d_t.

This means that the performance measures now become dependent on the threshold d_t, such that

$$\text{sensitivity}(d_t) = \frac{\text{correct detections}(d_t)}{\text{total number of true events}} \times 100\%, \tag{9.15}$$

and

$$\text{selectivity}(d_t) = \frac{\text{correct detections}(d_t)}{\text{total number of detections}(d_t)} \times 100\%. \tag{9.16}$$

where d_t describes a threshold level between 0 and 1.

The most desirable performance is to have 100% sensitivity and 100% selectivity, i.e., *no* missed events and *no* false detections. What generally happens, however, is that one measure performs well but only at the cost of poor performance on the other. A balance between the two measures is a good compromise. For the SOFM stage described here, three threshold levels were tried to examine the performance over the range of probabilities at the outputs. The three thresholds were; $d_t = 0.1$ (low d_t), $d_t = 0.50$ (mid d_t) and $d_t = 0.85$ (high d_t).

9.8 RESULTS

The SOFM was trained using the parameters and training data set described in Section 9.5.1. The trained SOFM was then calibrated using the Bayesian technique described in Section 9.5.3.2. This was followed by 'fine-tuning' using the LVQ2 technique. The whole process was repeated for varying SOFM sizes ($S = 10, 12, 14, 16, 18, 20$ for a $[S \times S]$ SOFM). A separate test set consisting of 6 EEGs was used to assess the performance of the combined mimetic/SOFM stage (see Table 9.6) at the three threshold levels.

EEG	Age	Duration	EEGers	Epileptiform Events (EVs)		
				Definite	Quest.	Total
1	71	25m 8s	2	17	17	34
2	3	16m 3s	2	8	17	25
3	31	24m 59s	2	2	19	21
4	11	23m 24s	2	2	1	3
5	5	27m 24s	3	9	0	9
6	24	26m 17s	2	25	16	41
Totals		2h 23m 15s		63	70	133

Table 9.6 The test set comprising 6 EEGs with a total of 63 definite EVs and 70 questionable EVs.

Figure 9.4 shows a plot of the mean change in Euclidean distance $\delta(k)$ (as described in Chapter 6) throughout the entire training process for the various SOFM sizes. As in the computer simulations of Chapter 6 the ordering phase is clearly visible where the mean changes in Euclidean distance of the $\mathbf{m}_i$ are larger until the changes become much smaller at the fine-adjustment phase. As the SOFM size is changed from $[10 \times 10]$ to $[20 \times 20]$, the weights become more 'settled' during the fine-adjustment phase.

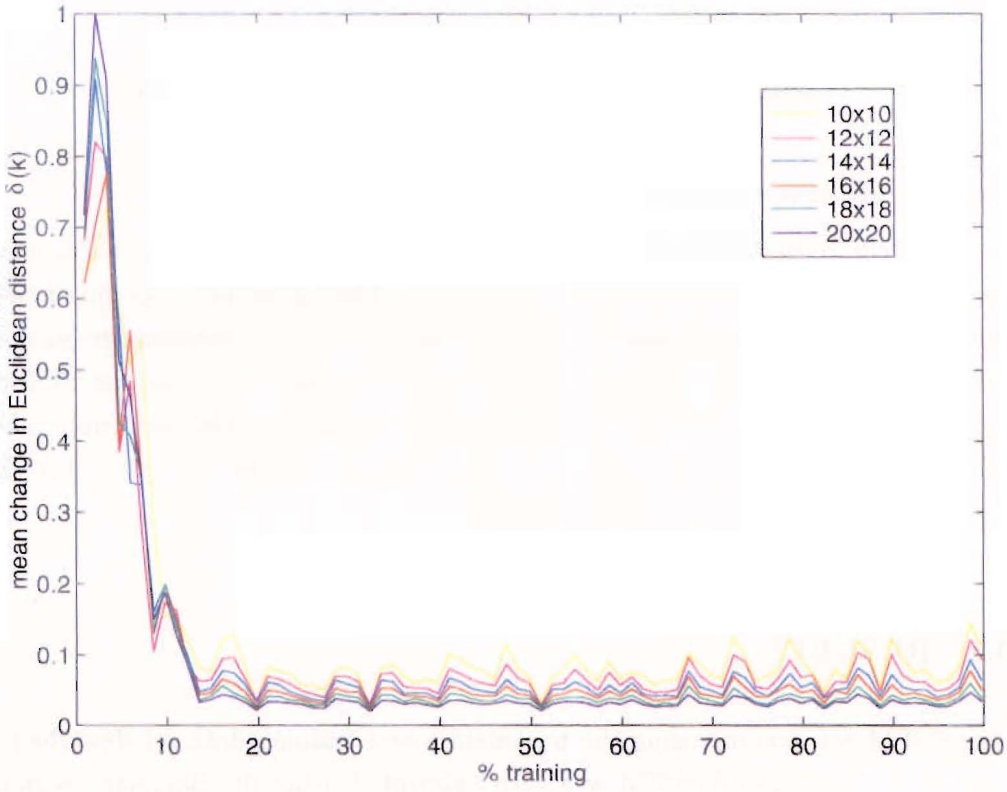


Figure 9.4 The mean change in Euclidean distance of the weights of the SOFM during training for SOFM sizes from $[10 \times 10]$ to $[20 \times 20]$.

Figure 9.5a depicts the weight vectors of the $[20 \times 20]$ SOFM after training and calibration (before LVQ2). The topological ordering of the 'ED-like' waveforms is quite apparent. Note that only the 'raw' EEG portion of the weight vectors is shown, the contextual parameters and passed/failed thresholds flag are not shown. Those waveforms that resulted in a probability greater than 0.5 after calibration are indicated in red whilst all others are in blue. It can be seen that a portion of the SOFM has come to represent *failed threshold* CEDs while the remaining portions depict *passed thresholds* CEDs. It can be said that *all* weight vectors describe ED waveforms but at different probabilities when using the technique described in Section 9.5.3.2.

Figure 9.5b depicts the weight vectors of the $[20 \times 20]$ SOFM after training and calibration, and after fine-tuning with LVQ2. Note how the LVQ algorithms do not preserve the topological ordering of the 'ED-like' waveforms.

Figure 9.6a shows the probabilities assigned to each neuron in the trained [20 × 20] SOFM after straight SOFM training. A probability of 1.0 is shown in red and a probability of 0.0 in blue, all other shades represent probabilities in between. Note that with the Bayesian based estimate of the probabilities, the probabilities of 1.0 and 0.0 can never actually be achieved. From Figure 9.6a it can be seen that there are a number of 'clusters' of neurons assigned high probabilities; these correspond to those weight vectors most representative of the EDs in the training set. Figure 9.6b shows the probabilities assigned to the SOFM after fine-tuning with LVQ2. When compared to Figure 9.6a it can be seen than in this case the clusters have been 'tightened' resulting in greater probability levels for neurons within those clusters representative of EDs. At the same time, the probabilities assigned to neurons less representative of EDs have been weakened as a result of the fine-tuning process.

Table 9.7 shows the performance of the SOFM stage for the [10 × 10] SOFM on the novel test data described earlier. The sensitivity and the selectivity are calculated as described in Equations 9.15 and 9.16 for various values of d_t. The values in brackets indicate the *total* number of events (i.e., definite + questionable) found by the EEGer and the system respectively. From the table it can be seen that at low d_t the system has a relatively high sensitivity (76%) but an extremely low selectivity (2%) overall. At mid d_t the selectivity improves noticeably (15%) but is accompanied by a substantial drop in sensitivity (34%). At high d_t the selectivity is at its best (22%) and sensitivity at its worst (17%).

Sensitivity/Selectivity (%)							
		low d_t ($d_t = 0.10$)		mid d_t ($d_t = 0.50$)		high d_t ($d_t = 0.85$)	
Patient	EEGer	Sen/Sel	System	Sen/Sel	System	Sen/Sel	System
1	(34)	68/2	(1185)	32/26	(43)	6/100	(2)
2	(25)	76/3	(590)	16/7	(54)	4/4	(25)
3	(21)	86/3	(601)	43/27	(33)	10/33	(6)
4	(3)	100/0	(682)	100/3	(92)	67/7	(28)
5	(9)	89/1	(1173)	78/13	(54)	56/25	(20)
6	(41)	73/4	(799)	27/33	(33)	27/50	(22)
Total	(133)	76/2	(5030)	34/15	(309)	17/22	(103)

Table 9.7 The sensistivities and selectivities of the [10 × 10] SOFM to definite & questionable EVs for each patient after fine-tuning with LVQ2. The performance is measured based on the criterion where the probability of *at least* one channel must exceed the threshold d_t for a detection to take place. Values in brackets indicate the number of events marked by the EEGer or detected by the system.

The effect on the performance due to varying the size of the SOFM can be seen in Table 9.8. In this table the overall performance across all the 6 patients (for definite + questionable events) is assessed as the SOFM size varies. Once more, the trend seen in Table 9.7 for the [10 × 10] SOFM is repeated for the various SOFM sizes in Table 9.8. A low d_t results in a relatively high sensitivity and a low selectivity whilst a high d_t

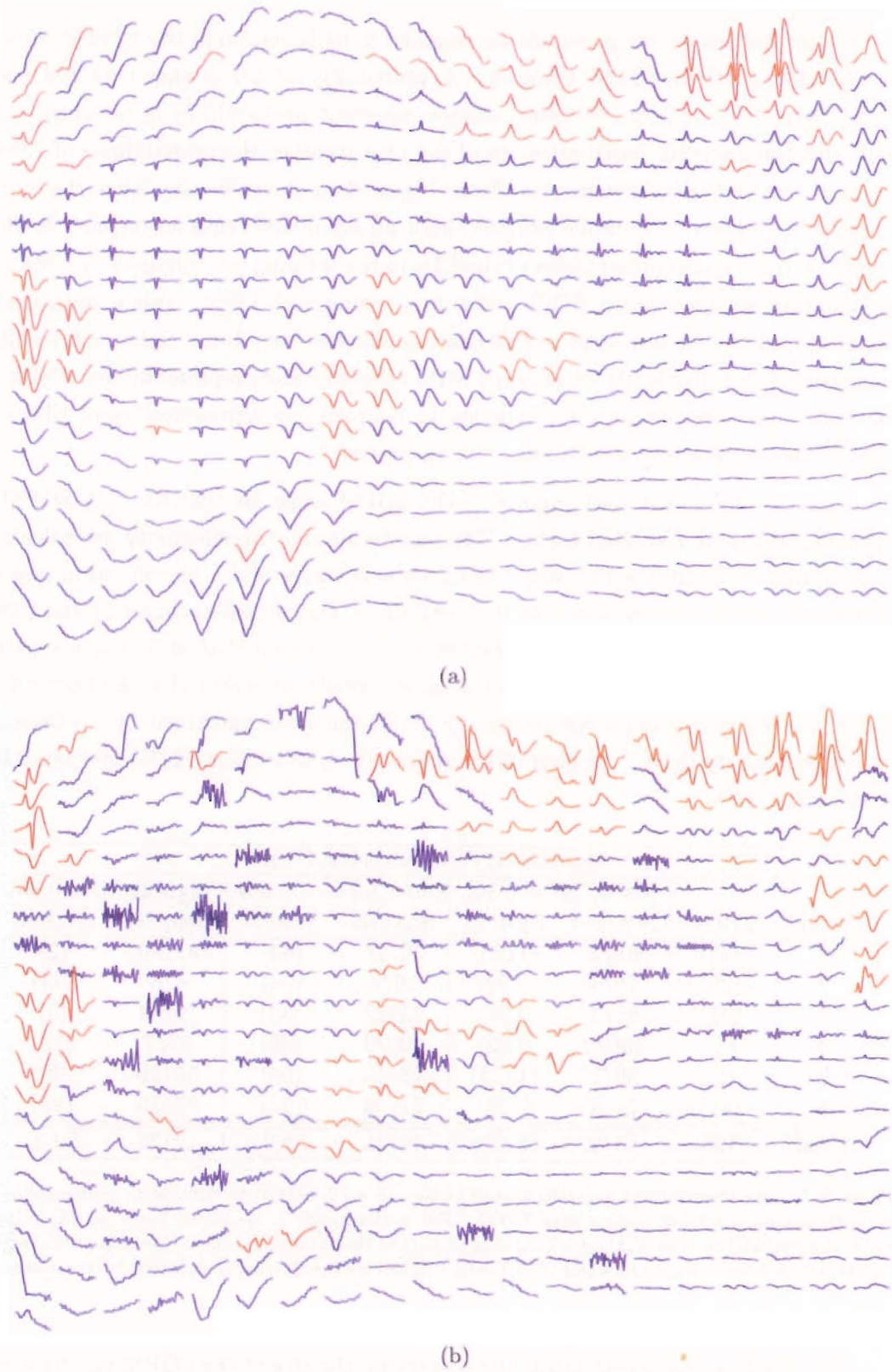
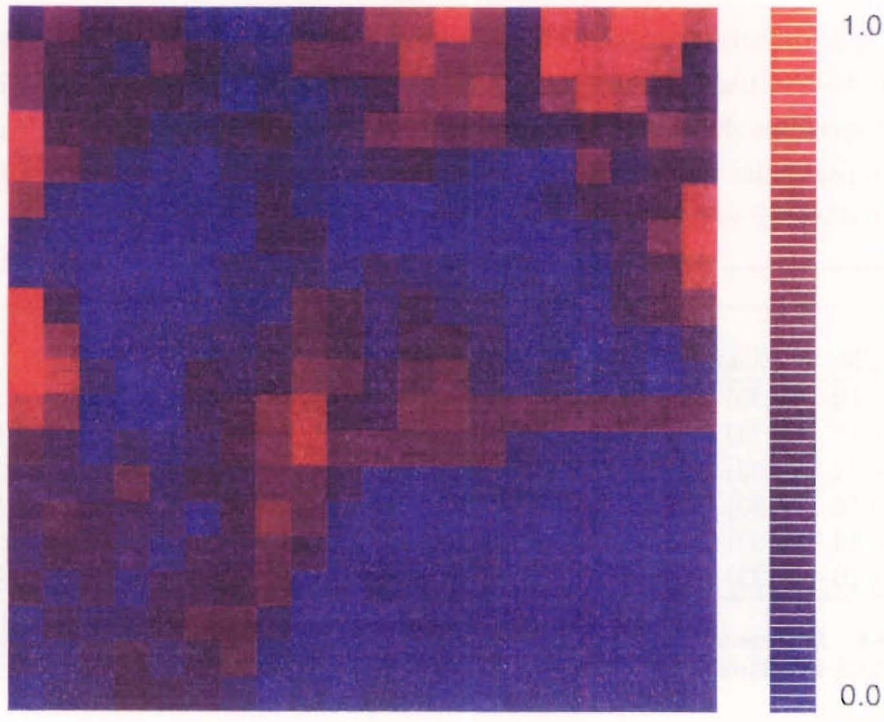
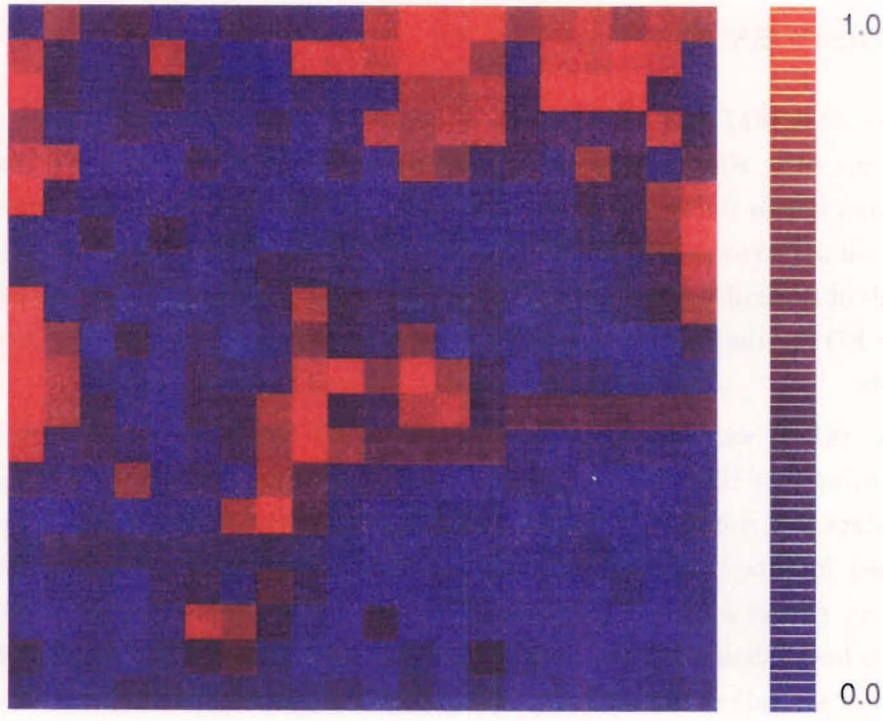


Figure 9.5 The weight vectors of the $[20 \times 20]$ SOFM (a) after training and calibration and (b) after 'fine-tuning' with LVQ2 (Note that the LVQ algorithms no longer preserve the topological ordering inherent with the SOFM training algorithm). Weight vectors which have been assigned a probability of greater than 0.5 are indicated in red whilst all others are in blue.



(a)



(b)

Figure 9.6 The probabilities assigned to each neuron in the $[20 \times 20]$ SOFM after (a) straight SOFM training (see figure 9.5a) and (b) after fine-tuning with LVQ2 (see figure 9.5b). The fine-tuning process of LVQ2 results in enhanced probability levels within clusters representative of EDs. (The largest probability is shown in red and the smallest in blue.)

results in a low sensitivity and a slightly raised level of selectivity. At mid d_t a balance between both extremes is obtained. As the SOFM size is increased, the performance in general increases slightly at the lower threshold levels, although no clear trend can be seen. In particular the performances at high d_t for the SOFM sizes [12 × 12], [14 × 14] and [16 × 16] are quite low.

Sensitivity/Selectivity (%)							
		low d_t ($d_t = 0.10$)		mid d_t ($d_t = 0.50$)		high d_t ($d_t = 0.85$)	
SOFM	EEGer	Sen/Sel	System	Sen/Sel	System	Sen/Sel	System
10 × 10	(133)	76/2	(5030)	34/15	(309)	17/22	(103)
12 × 12	(133)	70/2	(3834)	37/13	(365)	6/7	(108)
14 × 14	(133)	74/2	(4196)	35/13	(348)	5/5	(129)
16 × 16	(133)	77/3	(4039)	41/13	(422)	3/4	(114)
18 × 18	(133)	81/3	(3631)	41/14	(387)	22/18	(159)
20 × 20	(133)	77/3	(3374)	45/15	(412)	24/20	(158)

Table 9.8 The sensitivities and selectivities of the SOFMs across the whole test set (after fine-tuning with LVQ2), at different SOFM sizes – from [10 × 10] to [20 × 20].

9.9 DISCUSSION

The overall SOFM stage of the spike detection system has been designed to extract waveforms from single channel EEG (following the screening action of the mimetic stage) and assign to these waveforms (along with their contextual parameters) a probability value corresponding to their being a true ED. This is carried out across all 16 channels of the multichannel recording so that the SOFM stage outputs 16 probabilities to true ED for the waveforms (CEDs) centred within a 50 ms window across all the channels.

The SOFM was trained using a large data set consisting of a mix of waveforms representing true EDs, artifact, etc. Once the SOFM had been trained, a small subset of the data was used to calibrate the SOFM and further fine-tune it using LVQ2 (as described in Chapters 5 and 6). The calibration and fine-tuning processes are by their very nature a form of *supervised* training. This means that the calibration set needs to have labels assigned to each waveform with a high degree of confidence. The method of assigning a label based on a consensus of 2 or more EEGers, as described in Section 9.5.1, was devised such that only waveforms with a strong agreement between the EEGers were used for the calibration/fine-tuning process.

When observing the mean change in Euclidean distance between the weights of the SOFM (Figure 9.4) it can be seen that during the fine-adjustment phase the weights of the SOFM appear 'settled' with not much change between the [20 × 20] SOFM and the [10 × 10] SOFM. Table 9.8 suggests there is little to be gained by using one SOFM

size over another. On average, the [20 × 20] SOFM performed slightly better for each d_t value, but is not much better than, say, the [10 × 10] SOFM. The reason for the decrease in sensitivity and selectivity for the maps of size S=12, 14 and 16 in particular, is unclear.

When assessing the performance of this SOFM stage, it is not surprising that the selectivity will be very low (i.e., there are many false detections). As has already been stated, the EEGer relies on spatial (as well as temporal) clues to reliably detect an EV. Waveforms on a single channel may be assigned high probabilities (due to their 'spike-like' nature) but then be rejected at a later stage because there are no spatial clues indicating the presence of an EV. Figure 9.7 depicts two examples of waveforms assigned probabilities based on single channel information only. Figure 9.7a depicts a CED assigned a high probability (0.9) which was later seen to form part of an EV (by the final stage of the spike detector), whilst Figure 9.7b shows a 'spike-like' waveform assigned a high probability (0.81) but was rejected as artifact (electrode artifact) by the later stage.

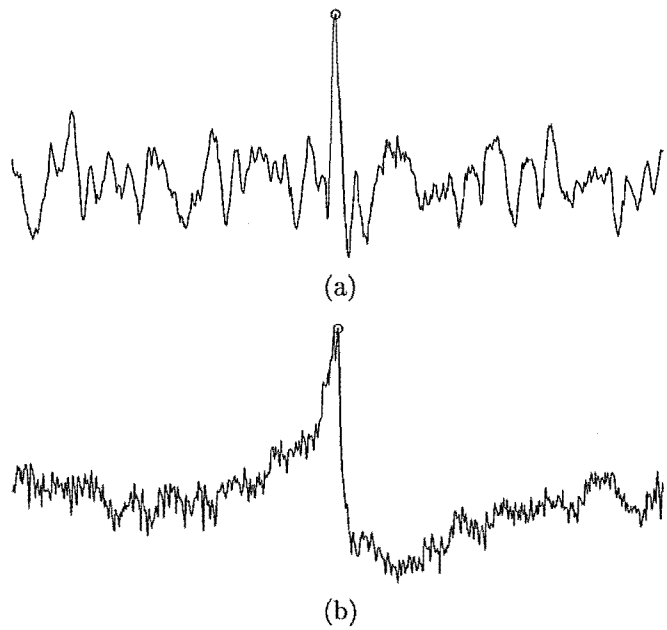


Figure 9.7 High probabilities assigned to two CEDs based on single channel analysis. (a) A probability of 0.9 was assigned to this CED which later turned out to form part of an EV (following spatial analysis). (b) A probability of 0.81 assigned to a CED which was later rejected as electrode artifact.

What may seem surprising is that the sensitivity is relatively low, even for low d_t. Sometimes individual EDs which make up EVs may be rather poorly defined (being buried in noisy background EEG, for example) when viewed singly. However, such cases could, when viewed spatially, exhibit all the characteristics of an EV with a given focus, albeit with small probabilities. This means that such occurrences have the possibility of being picked up spatially when they would otherwise be missed on a single channel

basis. Figure 9.8 depicts examples of waveforms assigned relatively low probabilities on a single channel basis but later detected as part of an EV through spatial analysis.

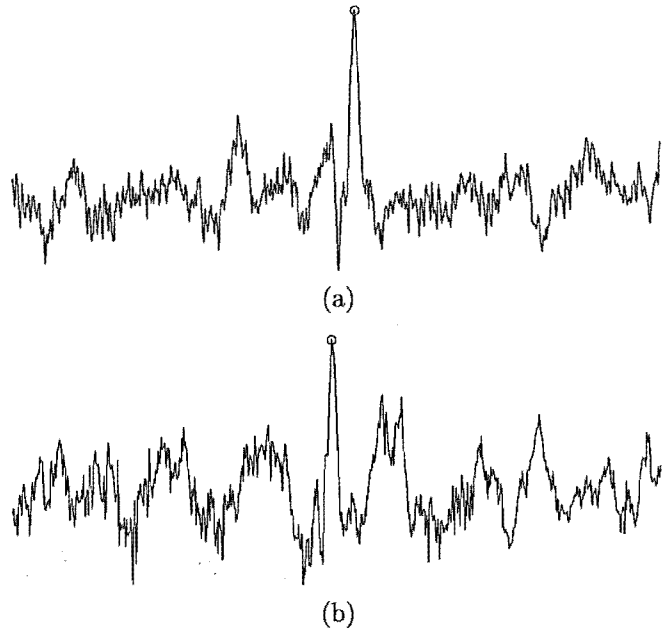


Figure 9.8 Low probabilities were assigned to these two CEDs based on single channel analysis alone. Both later turned out to form part of an EV (following spatial analysis). (a) probability: 0.10 (b) probability: 0.20.

9.10 SUMMARY

The self-organising ability of the SOFM to form an unbiased representation of the underlying features in the training data is drawn upon heavily in this stage of the spike detection system. Once the feature map is formed a small subset of the training set (the calibration set) is used to fine-tune the SOFM weights using the LVQ2 technique. The calibration set is made up of CEDs which are well labelled according to a concensus based on 2 (sometimes 3) EEGers. A probability (of it describing an ED) is assigned to each weight vector according to a modified calibration technique using Bayesian probabilities.

The mimetic stage performs a screening of the incoming EEG reducing the amount of data forwarded to the SOFM to just 'spike-like' data. The 'raw' EEG waveform is used as input rather than the parameters extracted by the mimetic stage. In addition three items of contextual information are appended to the waveform.

The mean change in Euclidean distances for the various SOFM sizes has been used to investigate what size of SOFM is required. The mean change in the Euclidean distance for each SOFM size tried shows only a slightly more 'settled' map as the SOFM size is increased. The performance varies only slightly, on average, with the larger of

the SOFMs, i.e., the [20 × 20] SOFM, performing only marginally better than for the other SOFM sizes. These findings indicate that there is no particular preference for one SOFM size over another at this stage.

The mimetic and SOFM stages combined in essence perform two main tasks. The first is to perform data reduction on the EEG with a high sensistivity (mimetic stage) and increase the selectivity somewhat (SOFM stage). The second task is to assign probability values to CEDs in each channel.

Relatively low sensitivity and selectivity was obtained on test data. This is a strong indication that multichannel (spatial) analysis is required for performing spike detection on the EEG. In the next chapter the test data described here is further analysed with spatial variation taken into account.

Chapter 10

SPATIO-TEMPORAL ASPECTS

10.1 INTRODUCTION

Throughout this thesis it has been repeatedly emphasized that EEGers use both spatial and temporal cues when looking for EDs in the EEG. The system proposed in Chapter 7 describes a final stage which utilises spatial and temporal clues in order to make the final EV/non-EV decision based on information received from the previous stage. This chapter presents this final stage which involves the use of *fuzzy logic* to implement spatial reasoning in the spike detection system.

Alternatives to the use of fuzzy logic exist. It would be possible to implement the spatial analysis of this final stage by means of an ANN (a multi-layer perceptron ANN for example). Using such an ANN would require a training data-set which is representative of the many different spatial distributions of EVs in the EEG. The training set would need to be labelled accurately and be truly representative of *all* the various spatial distributions possible. Such a data-set may be difficult to assemble and may suffer from biases on the part of the EEGers. The use of fuzzy logic as described in this chapter, however, allows the capture of the essence of the *reasoning* of an EEGer when performing spatial analysis, whilst retaining the ability to generalize to novel situations as they arise. Section 12.2 makes further comment on alternative implementations of the spatial analysis.

The performance of the overall, spike detection system, including the final stage, is assessed using the test patients introduced in the previous chapter. Finally, the concept of utilising temporal information to further enhance system performance is introduced.

10.2 THE SPATIAL-COMBINER

The final stage of the spike detection system combines the outputs of the SOFM stage in such a way as to confirm the presence of an EV across two or more channels of the EEG and, hence, report the detection of an EV. If the spatial pattern across the

outputs of the previous stage is inconsistent with the presence of an EV, it is rejected. This stage is dubbed the *spatial-combiner*.

In essence, the spatial-combiner uses a number of rules which specify allowable combinations of individual (i.e., single-channel) EDs across channels to detect the presence of an EV. The spatial-combiner works on a 4 channel bipolar electrode chain basis (see Section 2.3.2.2), where the incoming (bipolar) EEG is examined based on identical subsystems which group 4 channel bipolar chains together according to the current bipolar montage in use. Electrode chains with more than 4 channels (at most 6 channels) are split into 2 overlapping 4 channel chains, and shorter electrode chains are padded with 'nulls' to make up a 4 channel chain. The combiner relies on two pieces of information for each bipolar chain: (a) the probability assigned to each CED on each channel by the SOFM stage and (b) the polarity of each CED within the bipolar chain. Figure 10.1 depicts the spatial-combiner in relation to the overall spike detection system system, showing the 4 channel electrode chain subsystems described above.

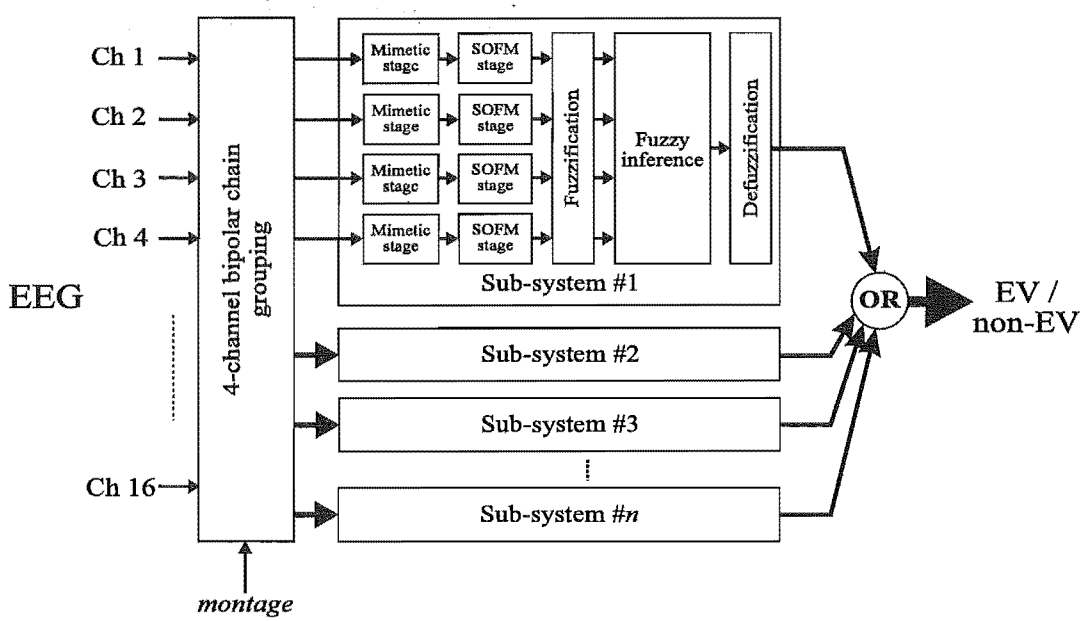


Figure 10.1 The spatial-combiner in relation to the complete spike detection system.

The rules mentioned earlier are drawn-up based on pre-defined knowledge of how an EV will manifest itself across a bipolar electrode chain and is described in more detail in Section 10.5. Each rule covers the possibility of a focal event at points along the bipolar electrode chain. As the probabilities assigned to each CED (by the SOFM stage) on each channel can take any value from 0 to 1, it would take a great many rules to cover every combination of polarity and probability value for each possible EV focus along a 4 channel chain. This problem is simplified by introducing fuzzy logic. Through the use of fuzzy logic, the generation of the spatial rules becomes easier as no explicit

mathematical models are needed that describe the underlying process. The fuzzy logic theory underlying the spatial-combiner and the implementation of the spatial-combiner itself is described in greater detail in the following sections.

10.3 FUZZY LOGIC THEORY

Fuzzy logic was first developed by Zadeh [1965] and is based on a mathematical theory which combines elements of multi-valued logic, probability theory, and artificial intelligence. Much research has taken place since Zadeh's seminal paper and today fuzzy logic has found itself in many areas of application, including expert systems, decision making, information processing, pattern recognition and process control.

Fuzzy logic simulates aspects of human thinking by incorporating the imprecision inherent in all physical systems [Klir and Folger 1988]. For example, as humans we prefer to use terms like "sunny" or "cloudy" to describe the weather rather than in exact percentages of cloud cover. In the former, some of the precise information is sacrificed in favour of a vague but more robust summary, whereas the latter is more accurate but less useful to us. However, even terms like "sunny" and "cloudy" have limited scope as, for example, at what point does the weather stop from being "sunny" and become "cloudy"? This implies a degree of overlap between the two 'sets', and is in fact precisely the concept of fuzzy sets as introduced by Zadeh. The fuzzy set forms, in essence, a generalization of the classical or *crisp* set.

10.3.1 Crisp sets and fuzzy sets

In a crisp set a sharp unambiguous distinction exists between the members and nonmembers of the class or category represented by the crisp set. Members 'unequivocally belong to a set' whereas nonmembers 'unequivocally do not'. However, in many of the categories employed in life, the distinctions employed are not so abrupt and their boundaries seem vague and the transition from member to nonmember seems rather gradual. So a fuzzy set introduces vagueness (with the aim of reducing complexity) by eliminating the sharp boundary dividing members of the class from nonmembers. Mathematically a fuzzy set can be defined by assigning to each of its members its *degree of membership* in the fuzzy set. Thus, individuals may belong to a particular fuzzy set to a greater or lesser degree based on their degree of membership. In the following, degrees of membership are assigned the variable μ_a (for fuzzy set 'a') and are represented by real-number values ranging in the interval between 0 and 1. As full membership and full nonmembership in the fuzzy set can still be indicated by 1 and 0 respectively, the crisp set can be considered to be a restricted case of the more general fuzzy set for which only these two grades of membership are allowed.

As stated above, a fuzzy set is defined by a membership function which usually encompass the entire range of input values of a given set, assigning to each a grade of membership. The shape of the membership function can be any shape particular to the application; in engineering applications it is usually one of trapezoidal, sigmoidal or Gaussian (trapezoidal is usually preferred because of its ease of computation). The *linguistic variables* (or *fuzzy variables*) used to describe fuzzy sets for engineering applications usually involve terms such as 'hot' or 'cold', 'high' or 'low', etc. Figure 10.2a depicts an example of three overlapping fuzzy sets with trapezoidal membership functions which describe *body temperature* in terms of fuzzy memberships. Each membership function describes the sets 'low', 'normal' and 'high'. An input at t_1 has a membership described by $\mu_n(t_1) = 0.55$, $\mu_h(t_1) = 0.9$, and $\mu_l(t_1) = 0$.

10.3.2 Fuzzy logic

Classical logic (i.e., two-valued logic) is based upon the basic assumption that every proposition is either true or false. Fuzzy-logic is a generalisation of *n-valued* logic ($n \geq 2$) which has a set of T_n *truth values* – as opposed to the two values for classical logic [Klir and Folger 1988].

Using Boolean algebra in classical logic, a logical expression can be broken down into three main logical operations: **AND**, **OR** and **NOT**. These operations can be extended for n-valued logic so that each operation becomes

$$\begin{aligned} a \textbf{ AND } b &= \min(a, b), \\ a \textbf{ OR } b &= \max(a, b), \\ \bar{a} &= 1 - a. \end{aligned} \tag{10.1}$$

It now becomes possible to perform logical operations on the membership values of a number of fuzzy variables. So for the membership functions defined in Figure 10.2a, the operation described in

$$t_1 \text{ is 'normal' } \textbf{AND } t_2 \text{ is 'high'},$$

reduces to

$$\begin{aligned} \mu_n(t_1) \textbf{ AND } \mu_h(t_2) &= \min\left(\mu_n(t_1), \mu_h(t_2)\right), \\ &= \min\left(0.55, 0.4\right), \\ &= 0.4. \end{aligned}$$

With the fuzzy logic operators described in Equation 10.1 it becomes possible to define *fuzzy rules* which produce an output conditional on some logical operation on

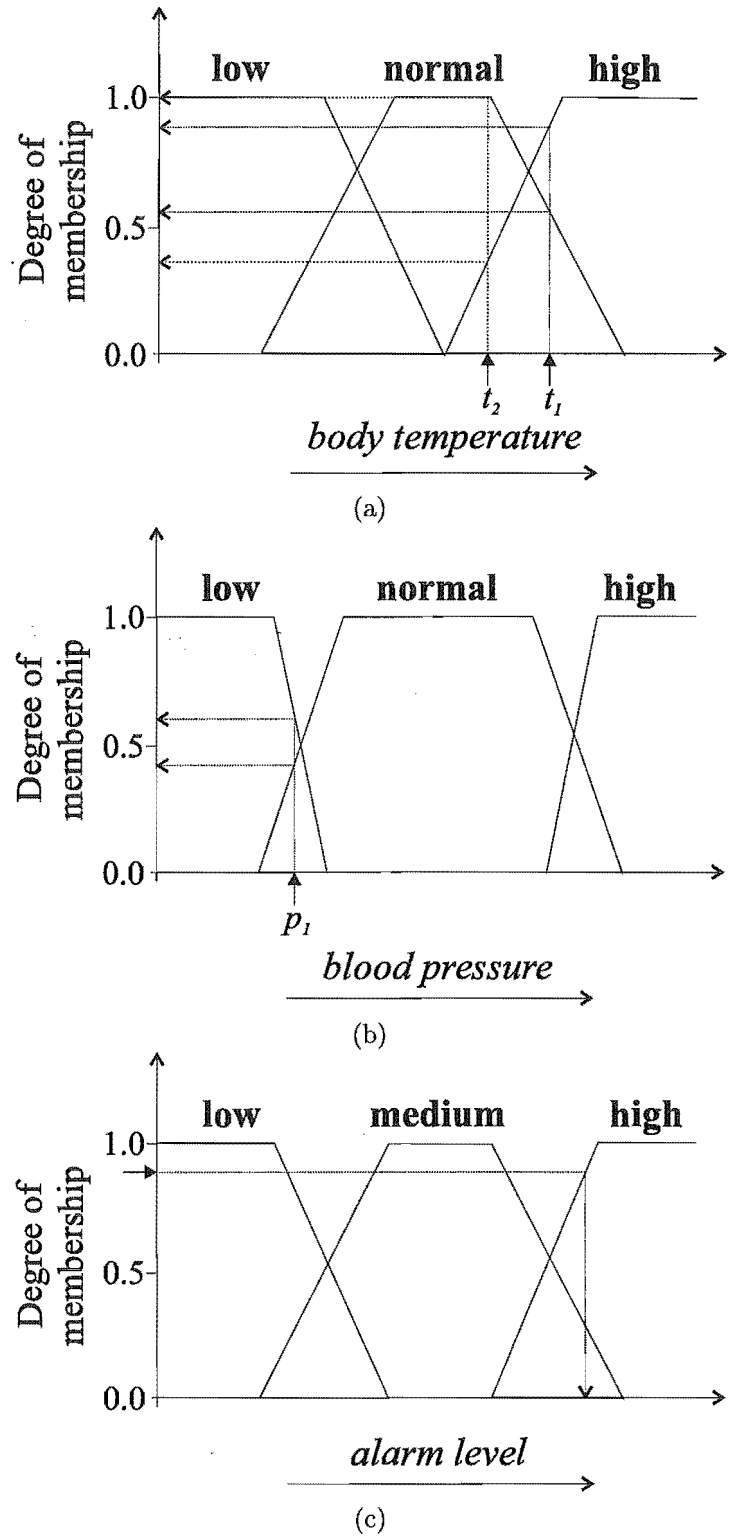


Figure 10.2 The fuzzy sets 'low', 'normal' and 'high' used to assign membership values to the inputs; (a) 'body temperature' and (b) 'blood pressure'. The fuzzy sets 'low', 'medium' and 'high' are used to assign membership values to the output (c) 'alarm level'.

the fuzzy variables in use.

10.3.3 Fuzzy rule-base

Rules are normally defined in the usual "**IF** condition **THEN** outcome" form, where the outcome of a rule depends on one or more conditions being met. For classical logic the conditions would consist of two-valued (crisp) conditions. Fuzzy logic offers the opportunity to perform some function based on a condition (or conditions) which allow degrees of membership [Zimmermann 1986]. An example of a fuzzy rule for a patient monitoring system, say, could be as follows

IF body temperature is 'high' **OR** blood pressure is 'low' **THEN** alarm level is 'high'.

Consider the fuzzy sets and fuzzy variables defined in Figure 10.2. If a patient's temperature and pressure were t_1 and p_1 respectively, the conditional part of this rule can be re-written as

$$\max(\mu_h(t_1), \mu_l(p_1)).$$

For the example given this results in an overall membership value of $\mu_h(t_1)$ (=0.9). The outcome of the above-mentioned rule is that for the output variable "alarm level" a certain degree of membership is defined in the fuzzy set "high". The degree of membership is given by the outcome of the conditional part (0.9).

When inputs are applied to the multiple rules of the fuzzy rule-base, all rules will respond with different membership values (albeit most may be 0). It then becomes necessary to combine the outcomes of all the rules and produce a single 'crisp' outcome for the input conditions given, this process is known as *defuzzification.* Many techniques are available for the defuzzification process but, in general, the two most common are to use the *composite moment* (i.e., centroid) or the *composite maximum.* The centroid takes the centre of gravity of the final fuzzy space and produces a result that is sensitive to all the rules, whereas the composite maximum produces a result that is sensitive to the single rule that has the highest outcome. Generally, process control applications use the centroid method as the results tend to move smoothly across the control surface, while information based applications tend to use the composite maximum. In the following the method of composite maximum is used in the defuzzification process.

The block diagram of Figure 10.3 depicts the fuzzy logic topology. The inputs are first fuzzified, inference then takes place on the fuzzy rule base, followed by defuzzification where the fuzzy outcome of the rules is converted to a crisp output.

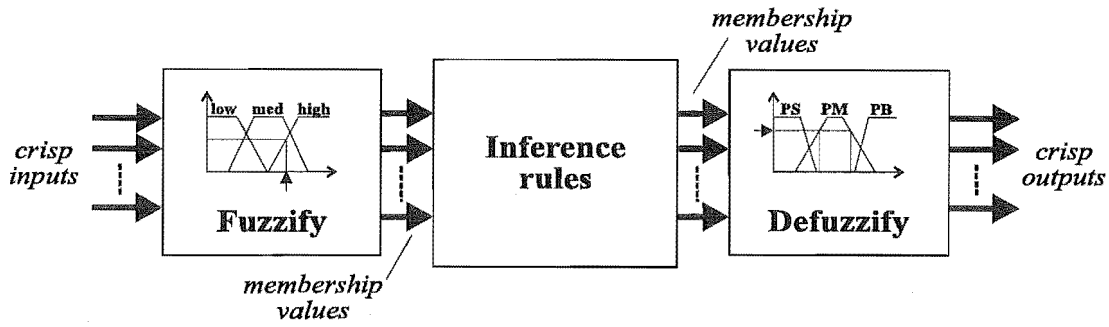


Figure 10.3 A block diagram depicting the fuzzy logic topology.

10.4 FUZZY LOGIC FOR THE SPIKE DETECTION SYSTEM

In practice, a precise model of a biological system may not be known or it may be too difficult to model. In those cases, fuzzy logic may be an appropriate tool for modeling and controlling the biological system, since our knowledge and experience are directly contained and represented in the fuzzy logic model without the need for explicit mathematical models. The application of fuzzy logic in biological systems ranges from control of blood pressure during anesthesia [Meier *et al.* 1992], the automated delivery of muscle relaxants [Mason *et al.* 1994], the production of medical expert systems [Hudson and Cohen 1994] and implementing intelligent alarms [Becker *et al.* 1994].

The spatial-combiner discussed in Section 10.2 makes use of the degree of uncertainty inherent in a fuzzy logic system by defining a fuzzy rule base to cover the possibility of detecting EVs across a 4 channel bipolar chain of electrodes given a set of single channel probabilities.

10.4.1 Fuzzification

The crisp inputs to the spatial-combiner are the probabilities of true ED as output by each SOFM of the previous stage. The crisp inputs for each channel are fuzzified by using the fuzzy sets defined in Figure 10.4. For each channel, the crisp inputs can range between 0 and 1 with a positive or negative polarity (depending on the polarity A_p of the vertex of each CED as detected by the mimetic stage in Section 9.4). A waveform with a negative polarity is taken to be a waveform with a peak deflection upwards in line with current practice in analysing the EEG. The fuzzy sets are defined to be: NB (negative big), NS (negative small), ZE (zero), PS (positive small) and PB (positive big). Trapezoidal membership functions are used because of their ease of implementation.

As there are now only 5 fuzzy variables, for a 4 channel electrode chain there are

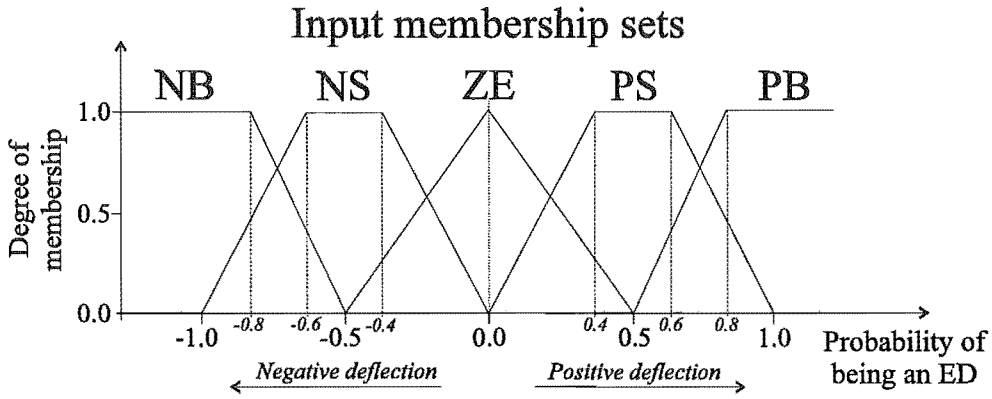


Figure 10.4 The fuzzy sets used to define the inputs to the spatial-combiner.

a maximum of $5^4 = 625$ possible rules which cover all the possible combinations of inputs. A large number of these rules are meaningless, however, and are, therefore, not used. The rules to be used are determined according to our expectation of the outcome given certain input conditions and are described more fully in Section 10.5.

10.4.2 Inference

Once the SOFM stage assigns a probability to each CED, the inputs are split into 4 channel groups according to the current montage in use. For each group, the inputs are fuzzified and presented to each rule in the fuzzy rule-base in turn. Each rule performs an **AND** function on the four inputs and results in a single output (i.e., a membership value and a corresponding fuzzy output variable). Each rule defines a 'possible' EV.

As there are more than one sub-system the overall output of the spatial-combiner is taken to be the subsystem with the largest output, that is, the outputs of each sub-system are **OR**ed together as depicted in Figure 10.1.

10.4.3 Defuzzification

Each sub-system produces a single output which is defuzzified using the fuzzy sets described in Figure 10.5. The 4 fuzzy sets are defined to be: ZE (zero), POS (possible), PRO (probable) and DEF (definite) and cover a crisp-valued range between 0 and 1 inclusive. Once more trapezoidal membership functions are used because of their ease of implementation. The method of composite maximum is adopted for the defuzzification process in this system. Using this method, the rule most representative of an 'allowable' EV distribution across the 4 inputs contributes to the fuzzy output set label and the membership value.

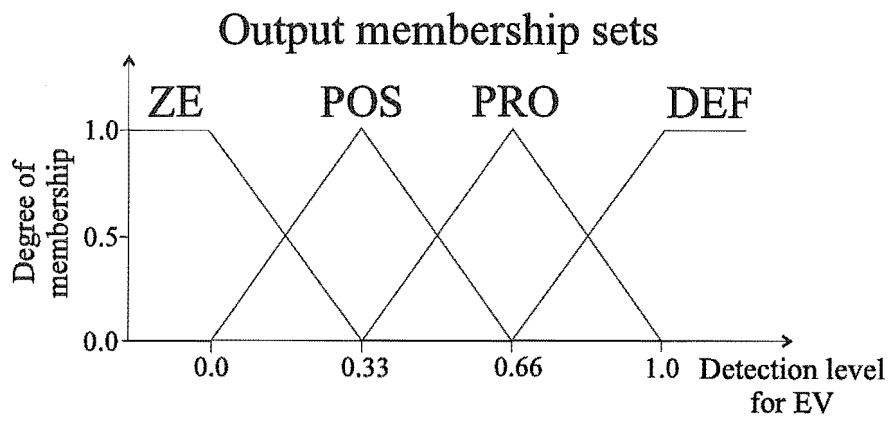


Figure 10.5 The fuzzy sets used to define the output of each sub-system in the spatial-combiner.

It is possible for an artifact or some other non-ED waveform to elicit high probabilities on certain channels but which do not exhibit the right spatial characteristics to be graded an EV. As the rules only define possible outcomes for *true* EDs, when presented with data which doesn't exhibit valid spatial characteristics each rule will result in a low membership value for the output variable.

In order to obtain a fuzzy output of ZE the systems works on the assumption that if the membership value of the best matching rule is less than 0.5 then the outcome of the whole fuzzy inference process is assigned an output of fuzzy variable ZE with a membership value of one minus the membership value of the best matching rule. In effect this represents the one rule which performs a **NOT** operation on each of the previous rules.

Once the best responding rule is established, the resulting membership value gives one or two crisp values at the points where the membership value intersects the corresponding fuzzy membership function. If there are two crisp values, the final crisp value output is taken to be the mean of these two values.

Both the the fuzzy output set label and the crisp value derived thereof are preserved at the output of the spatial-combiner, the crisp value or *detection level* being represented by D_{sys}. With the availability of both the fuzzy label and the crisp output, a detection of an EV can be made (a) if the crisp output D_{sys} exceeds a given threshold or (b) if the fuzzy output is either of POS, PRO or DEF. The spike detection system implemented uses the latter method, although both values are available on request.

10.5 DERIVING FUZZY SPATIAL RULES

It is possible to calculate the potential that would be generated at each scalp electrode by a dipole placed within a realistic head model. The dipole orientation and location could be varied within the model to give different voltage readings at each scalp electrode position. In this way it is possible to model the effect of a focal voltage as measured at the scalp across a bipolar chain of electrodes. As an EV generally gives rise to a region of negative potential measured at the scalp (relative to some indifferent reference on the body), it then becomes possible to model the effect of having such a focal discharge at various positions on the scalp. This can then be used to map out the different voltages (as measured along the bipolar chain of electrodes) due to such a focus, and derive a set of rules describing the state of each electrode for a given discharge.

The difficulty with the procedure just described is that it requires precise information regarding dipole orientation, location and intensity for each dipole position and would require an unacceptably large number of rules to adequately cover all the possible combinations that can be measured. Also, although an EV must, due to volume conduction, appear at all electrodes across the scalp, in many cases is only significant on a few of the channels (generally those closest to the focus); that is, on most channels the discharge is entirely masked by higher amplitude background EEG or artifact. This introduces a factor of uncertainty into the system and it is for this reason that EVs are represented by probabilities on each individual channel thus far.

By introducing fuzzy logic to the spike detection system, rather than having to map precise values at each electrode for each particular focus, a number of fuzzy sets can be defined which encompass the range of probabilities at each scalp electrode. Through the use of the fuzzy sets it then becomes possible to model the effect that an ED would have when present at various points along a given bipolar chain of electrodes. The fuzzy sets remove the need for measuring precise values at each electrode and allow for the creation of a set of rules based on our knowledge of what would constitute an "allowable" event along a given bipolar chain.

The set of rules governing "allowable" events are generated according to the following process:

1. A radially oriented dipole is modelled at a given depth beneath the surface of a 4 channel bipolar electrode chain.

2. The dipole is moved such that its associated peak negative potential is alternately just beneath an electrode and mid-way between two electrodes at the same fixed depth beneath the surface. This gives rise to 11 possible locations for the peak along the bipolar chain.

3. At each position of the dipole, the probability of the presence of an ED on each *channel* is assigned, based on the assumption that channels experiencing the highest magnitude voltage will have the greatest probability and those experiencing the lowest magnitude (commonly those furthest away) will have the least probability, with a linear interpolation for the probabilities of the channels in between. The probabilities at each channel are calculated in terms of the fuzzy input variables NB, NS, ZE, PS and PB.

4. The process is repeated for several depths of the dipole beneath the surface. The maximum depth is assumed to be such that all channels in the bipolar chain are affected and the minimum depth such that only the channel closest to the peak negative voltage at the surface is affected.

Figure 10.6a & Figure 10.6b depict examples of the effect of having the dipole deep below the surface, while Figure 10.6c & Figure 10.6d depict the dipole nearer to the surface.

Fuzzy input	**Weight**
PB/NB	0.25
PS/NS	0.15
ZE	0

(a)

Fuzzy output	**Thresholds (w_T)**
DEF	$w_T \geq 0.80$
PRO	$0.5 \leq w_T < 0.80$
POS	$0.15 \leq w_T < 0.5$
ZE	$w_T < 0.15$

(b)

Table 10.1 The weightings (a) and thresholds (b) used in assigning a fuzzy output variable to each fuzzy spatial rule derived.

Each set of probabilities calculated (in terms of the fuzzy variables) for each location of the dipole, determines a fuzzy rule for the location of an ED focused at the point of maximum negative voltage at the surface. This results in 127 distinct fuzzy rules (out of a maximum of 625 rules) describing allowable combinations of fuzzy variables for the 4 inputs of each sub-system. Each rule derived in this manner is assigned an outcome of either DEF, PRO or POS. The actual fuzzy output label assigned to each particular rule was arbitrarily based on the number of PB/NB and PS/NS variables assigned to each particular rule. For example, a rule involving 4 PB/NB variables was assigned DEF, a rule with 4 PS/NS was assigned PRO and a rule with 3 or less PS/NS was assigned the label POS. This was done by arbitrarily assigning a weight to each fuzzy input variable and adding up the total weights for each fuzzy rule derived. The fuzzy output variable assigned to each rule was then based on the sum of these weights exceeding a number of thresholds. The weights assigned to the fuzzy input variables are given in Table 10.1a and the thresholds for each fuzzy output variable in Table 10.1b. For example, rule 1 in the following rules is assigned the fuzzy output variable of PRO based on the sum

of the weights of the input variables (0.25+0+0.25+0.25=0.75) exceeding 0.5 but less than 0.8. A complete list of the fuzzy rules derived can be found in Appendix C.

The following gives three examples of rules obtained using the method just described:

1.	**IF**	CH1 is PB	**AND**
		CH2 is ZE	**AND**
		CH3 is NB	**AND**
		CH4 is NB	
	THEN	OP is PRO,	
2.	**IF**	CH1 is ZE	**AND**
		CH2 is PS	**AND**
		CH3 is PB	**AND**
		CH4 is NB	
	THEN	OP is PRO,	
3.	**IF**	CH1 is NB	**AND**
		CH2 is NS	**AND**
		CH3 is ZE	**AND**
		CH4 is ZE	
	THEN	OP is POS,	

where CH1, CH2, CH3 and CH3 represent the input to channels 1, 2, 3 and 4 respectively and OP the output from each sub-system. For rule 1 a 'strong' focus is assumed between electrodes 2 and 3 (Figure 10.6a), in rule 2 a focus at electrode 4 (Figure 10.6b) and in rule 3 a 'weak' focus at electrode 1 (Figure 10.6c).

The underlying assumption when deriving the fuzzy spatial rules as shown in this section is that a focal EV is detected along the bipolar chain of electrodes. Generalized EVs show no distinct focus, however, across a number of 4 channel bipolar chains, generalized activity will be manifest as a focal event at the end (or beginning) of each chain. This means that the rules derived using the above method will also be able to detect generalized activity.

10.6 FURTHER SPATIAL CONSIDERATIONS

In order to avoid falsely detecting artifacts as EVs, individual outputs from the SOFM stage are eliminated if they are due to muscle contraction, eye-blinks or electrode

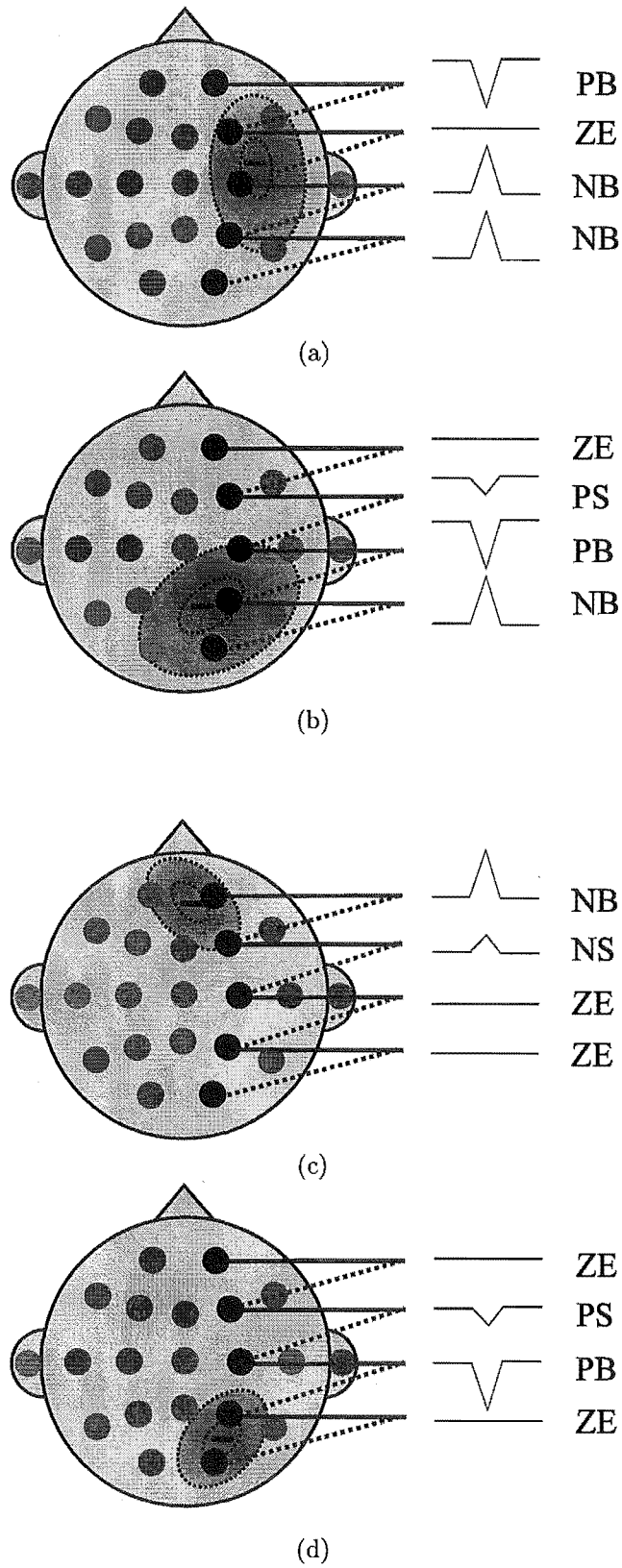


Figure 10.6 The method used to assign fuzzy variables to each channel when defining the fuzzy rules for each sub-system of the spatial-combiner. (a) & (b) depict the dipole deep below the surface at different positions along the bipolar chain and (c) & (d) depict the dipole nearer to the surface.

movement. Elimination takes place by assigning a probability value of 0 on the channel to be eliminated. The detection of one of the above-mentioned occurrences is made through the use of the contextual information presented with the CED for each channel and is based on work by Dingle [1992]. Each artifact is tackled in the following way:

1. Bursts of muscle activity on any channel are detected when (a) the average background duration is short (< 20 ms, corresponding to the high frequency waveforms characteristic of EMG) and (b) the RMS background amplitude is large (>12.6 μV).

2. Eye-blinks are detected when the floating-mean falls significantly below the baseline (< -80 μV) on a *frontal* channel.

3. Electrode movement is detected on a channel when the floating mean reaches a maximum of at least 100 μV above the baseline.

10.7 METHODS

An experimental study was performed to assess the performance of the spatial-combiner as part of the overall spike detection system.

10.7.1 Subjects

The same 6 EEGs used to test the SOFM stage in the previous chapter were used. The details of the 6 EEGs are reproduced here in Table 10.2. The EEGs were recorded using the method described in Section 9.3 and contained only bipolar montages (all referential montages being transformed to longitudinal bipolar montage).

				Epileptiform Events (EVs)		
EEG	Age	Duration	EEGers	Definite	Quest.	Total
1	71	25m 8s	2	17	17	34
2	3	16m 3s	2	8	17	25
3	31	24m 59s	2	2	19	21
4	11	23m 24s	2	2	1	3
5	5	27m 24s	3	9	0	9
6	24	26m 17s	2	25	16	41
Totals		2h 23m 15s		63	70	133

Table 10.2 The test set comprising 6 EEGs with a total of 63 definite EVs and 70 questionable EVs.

10.7.2 Selecting the CEDs

During the development of the system, the mimetic stage was initially set up such that a CED was declared once the first vertex was found which exceeded all thresholds, this being called the primary vertex. This was repeated on all channels for vertices within 50 ms of the primary vertex. However, this approach frequently caused the mimetic stage to form CEDs around an inappropriate vertex for a known ED (particularly with a bi-phasic ED). This caused two problems: (a) the CED was windowed around the wrong vertex and so was presented to the SOFM shifted in time, and (b) on most occasions the probability resulting from the presentation of the CED to the SOFM was coupled to the wrong polarity. The latter was because the polarity assigned to the probability value output from the SOFM was that of the (incorrect) vertex of the CED, A_p. When looked at in a spatial context, the resultant probabilities and polarities across channels did not conform to any of the derived rules and, consequently, the entire event was rejected. A similar problem was reported in the mimetic stage of Webber *et al.* [1994].

To correct this problem a system was devised along similar lines to that of Webber *et al.* [1994]. For each channel, if there was more than one CED present within a window of 100 ms, a score was calculated for each CED; one point was assigned to the CED with the greatest peak amplitude and one point to the CED with the sharpest vertex. This was performed for all channels of the EEG if more than one CED was present. A final score was assigned to each CED based on the distance of the vertex from the mean location of the vertices calculated from all of the CEDs for a given bipolar chain. First, the mean vertex location was calculated for each channel using each CED. The mean for each channel was then used to find the global mean location of the vertex across the 4 channels forming a bipolar chain. This was done in order to obtain an estimate of the location of a focus across the channels. Once this was done, the CED with the closest vertex to the global mean vertex position was assigned a further point. The CED with the greatest number of points on each channel was the CED chosen to be presented to the SOFMs.

This *ad hoc* method helped to alleviate the initial problem through a more informed choice of CED for each channel.

10.7.3 Grouping the inputs to the sub-systems

The outputs from the SOFM stage needed to be grouped into 4 channel bipolar chains (depending on the montage in use). Once grouped each group was presented to a sub-system which in turn output a fuzzy output value and crisp value. The overall system output was taken to be the maximum output of all the sub-systems. Each sub-system performed exactly the same operation on the four probability values input to it.

The 4 channel bipolar chains were grouped according to the montage in use as

depicted in Figure 10.7 (see Figure 2.1 for position and labels of electrodes). For the longitudinal and longitudinal-transverse montages, the sub-systems are naturally grouped into 4 sub-systems of 4 channels of EEG each (Figure 10.7a and Figure 10.7b). The transverse montage is divided into 5 sub-systems (Figure 10.7c). The 6 channel electrode chain *a*1-*t*3-*c*3-*cz*-*c*4-*t*4-*a*2 is divided into two overlapping 4 channel chains such that *a*1-*t*3-*c*3-*cz*-*c*4 are input to one sub-system and *c*3-*cz*-*c*4-*t*4-*a*2 are input to another sub-system. The 2 channel electrode chain of *fz*-*cz*-*pz* is converted to a 4 channel chain by padding the last two channels with 0 probabilities for every value entered on the two 'real' channels. The circumferential montage is divided into 6 sub-systems (Figure 10.7d). Two non-overlapping 4 channel chains were formed with *fp*2-*f*8-*t*4-*t*6-*o*2 and *fp*1-*f*7-*t*1-*t*5-*o*1 and two overlapping 4 channel chains with *f*3-*f*4-*c*4-*p*4-*p*3 and *f*4-*f*3-*c*3-*p*3-*p*4. Finally two 3 channel bipolar chains were padded with 'null' channels (i.e., 0 probability channels) for the chains *f*7-*fp*1-*fp*2-*f*8 and *t*5-*o*1-*o*2-*t*6.

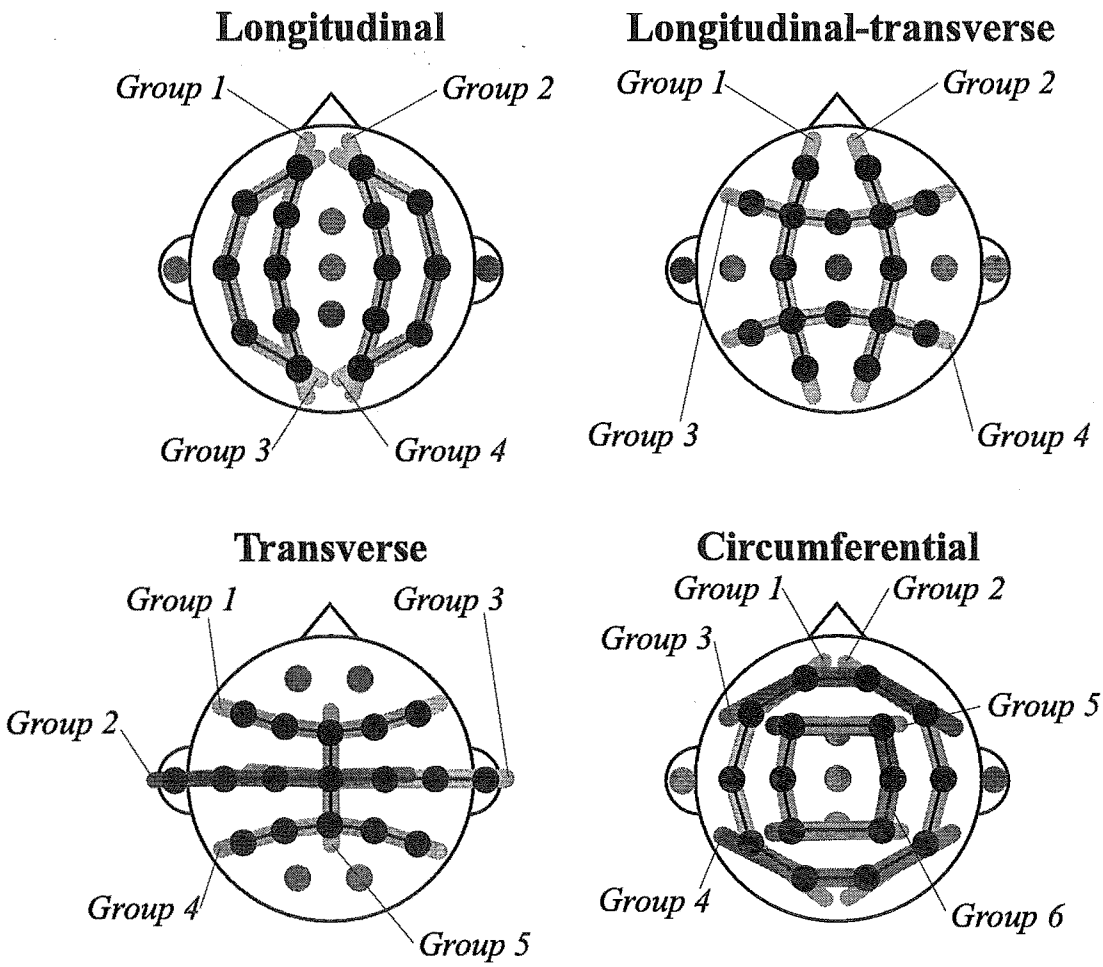


Figure 10.7 The grouping into 4 channel bipolar chains for each sub-system in the spatial-combiner for (a) longitudinal montage, (b) longitudinal-transverse montage, (c) transverse montage and (d) circumferential montage.

10.8 RESULTS

Table 10.3 shows performance of the spike-detection system including the spatial-combiner for the 6 test patients with a [20 × 20] SOFM in the SOFM stage. The table depicts the performance of the system for detecting definite plus questionable events. The total sensitivity is given by 59% and the selectivity by 32%. The low selectivity reflects the relatively large number of false detections made by the system. Both the missed events and the false detections seem to be evenly distributed throughout the test data and cannot be identified as being mainly due to one poorly performing EEG.

Sensitivity & Selectivity (%)						
		Detections				
Patient	EEGer	Correct	False	Total	Sensitivity	Selectivity
1	34	19	9	28	56	68
2	25	14	43	57	56	25
3	21	12	43	55	57	22
4	3	2	20	22	67	9
5	9	7	35	42	78	17
6	41	24	17	41	59	59
Total	133	78	167	245	59	32

Table 10.3 The sensitivities and selectivities of the spike-detection system to definite & questionable EVs combined for each patient in the test set. The [20 × 20] SOFM after fine-tuning with LVQ2 was used in the SOFM stage. The performance is measured based on the criterion that a detection takes place when the fuzzy output of any sub-system in the spatial-combiner is either POS, PRO or DEF.

Table 10.4 shows the performance of the complete system on the 6 test patients for different sizes of the SOFM, from a [10 × 10] to a [20 × 20] SOFM. The number of correct detections in general increases as the SOFM size is increased leading to better sensitivities for the larger SOFMs. The total number of detections shows no particular trend as the SOFM size is varied and this is reflected in the values for selectivity. On the whole the [20 × 20] SOFM resulted in the best selectivity, although all were rather low. On average, the trend for the average performance (i.e., mean of sensitivity and selectivity) is such that increases are seen as the SOFM size is increased, making the system using the [20 × 20] SOFM the best performed system.

Table 10.5 shows the number of missed events over the 6 patients for the [20 × 20] SOFM. Overall, there were 55 missed events out of the 133 events found by the EEGers. The table breaks these into 3 classes of missed events for each patient. Overall, 5 events were completely missed by the system, this being due to the mimetic stage failing to pick up the constituent CEDs and hence not forwarding them to the later stages. A further 15 events were picked up by the mimetic stage and forwarded to the SOFM stage but the constituent CEDs were assigned low probabilities resulting in the event

Sensitivity & Selectivity (%)						
		Detections				
SOFM	EEGer	Correct	False	Total	Sensitivity	Selectivity
[10 × 10]	133	64	185	249	48	26
[12 × 12]	133	49	124	173	37	28
[14 × 14]	133	58	128	186	44	31
[16 × 16]	133	66	190	256	50	26
[18 × 18]	133	71	245	316	53	22
[20 × 20]	133	78	167	245	59	32

Table 10.4 The sensitivities and selectivities of the spike-detection system to definite & questionable EVs across the whole test set whilst varying the SOFM size (after fine-tuning with LVQ2). The performance is measured based on the criterion that a detection takes place when the fuzzy output of any sub-system in the spatial-combiner is either POS, PRO or DEF.

being rejected by the spatial-combiner. These missed events were evenly distributed across the 6 test patients. Importantly, 35 events were missed due to 'wrong polarity'. That is, the constituent CEDs for these events were picked up by the mimetic stage, assigned strong probabilities by the SOFM stage but were rejected by the spatial-combiner because spatially they did not conform to any of the 'allowable' rules. On closer examination this was found, in every case, to be due to the wrong polarity being coupled to the CED probability due to the problems discussed in Section 10.7.2 (despite the *ad hoc* method used to attempt to overcome the problems). Had the individual CED polarities of these 35 events been correctly assigned, the overall sensitivity of the system would rise to 85% as opposed to the current 59% (for the [20 × 20] SOFM).

Missed EVs				
EEG	Missed entirely	Low probabilities	Wrong polarity	Total
1	0	3	12	15
2	1	4	6	11
3	0	4	5	9
4	0	0	1	1
5	0	1	1	2
6	4	3	10	17
Totals	5	15	35	55

Table 10.5 The missed EVs for the 6 test patients with the SOFM size at [20 × 20].

Table 10.6 shows a breakdown of the false detections made by the system for the 6 test set patients using the [20 × 20] SOFM. It is these 167 false detections which result in the relatively low selectivity (32%) of the system. Five of the false detections were due to eye-blink artifact and were all recorded from patient 3. Muscle artifact accounts for 26 false detections, most of which were recorded from patients 1 and 2. The major contributors to the false detections were electrode artifacts (66) and sharp background

(70). Electrode artifacts include both electrode movement and electrode 'pop', although the majority of the cases were due to electrode 'pop'. In each case, when viewed on a single channel basis, the CEDs were remarkably 'spike-like' and thus were assigned relatively high probabilities (i.e., $0.6 \sim 0.7$) by the SOFM stage. The electrode 'pop' artifacts, in particular, exhibited strong focal characteristics and this, coupled with the relatively high individual CED probabilities, caused the spatial-combiner to accept the events as epileptiform. The majority of the electrode artifacts were due to patients 2 and 5. The sharp background artifacts were mainly due to patient 3 (36 events) and consist of 'spike-like' background activity which happened to exhibit acceptable spatial characteristics and, hence, be detected as EVs by the spatial-combiner.

False EV detections					
EEG	Eye blink	Muscle artifact	Electrode artifact	Sharp background	Total
1	0	7	1	1	9
2	0	11	25	7	43
3	5	0	2	36	43
4	0	1	11	8	20
5	0	4	25	6	35
6	0	3	2	12	17
Totals	5	26	66	70	167

Table 10.6 The falsely detected EVs for the 6 test patients with the SOFM size at [20 × 20].

10.9 DISCUSSION

On the whole, the overall spike-detection system had a reasonable sensitivity (59%) but a rather poor selectivity (32%). Following closer examination of the missed events and false detections which resulted in these measures of performance, a number of points can be made.

Missed events were primarily due to the wrong polarity being assigned to CEDs, despite their being assigned strong probabilities by the SOFM stage. This occurred despite the measures taken as described in Section 10.7.2. Figure 10.8 depicts three instances of a 4 channel sub-system of the spatial-combiner acting on CEDs. Figure 10.8a shows a true EV where the CEDs presented to the SOFMs were assigned high probabilities and coupled to the appropriate polarity (based on the vertex of each CED) resulting in the detection of an EV. Figure 10.8b and Figure 10.8c depict other examples of true EVs presented to a sub-system. In each case, the CEDs were assigned good probabilities by the SOFMs but were coupled to the wrong polarity resulting in the EVs being rejected by the spatial-combiner. The figures show that the vertices chosen for each CED resulted in the wrong polarity being coupled to each single channel

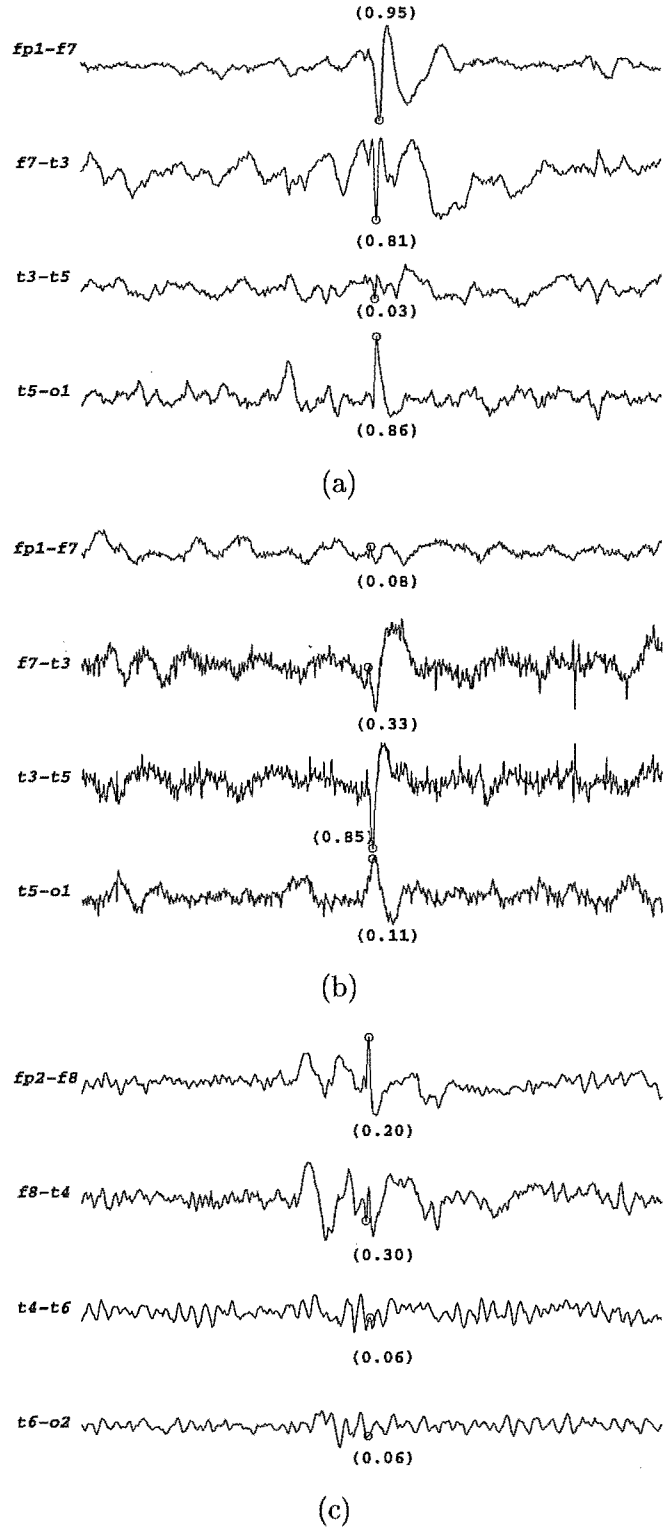


Figure 10.8 (a) An example of an event correctly detected as DEF following spatial reasoning, in contrast to (b) & (c) which depict two events rejected following spatial reasoning due to wrongly assigned polarities of the vertex in each case. (The values in brackets indicate the probability value assigned to each CED by the SOFM stage and the circle indicates the vertex of each CED in question).

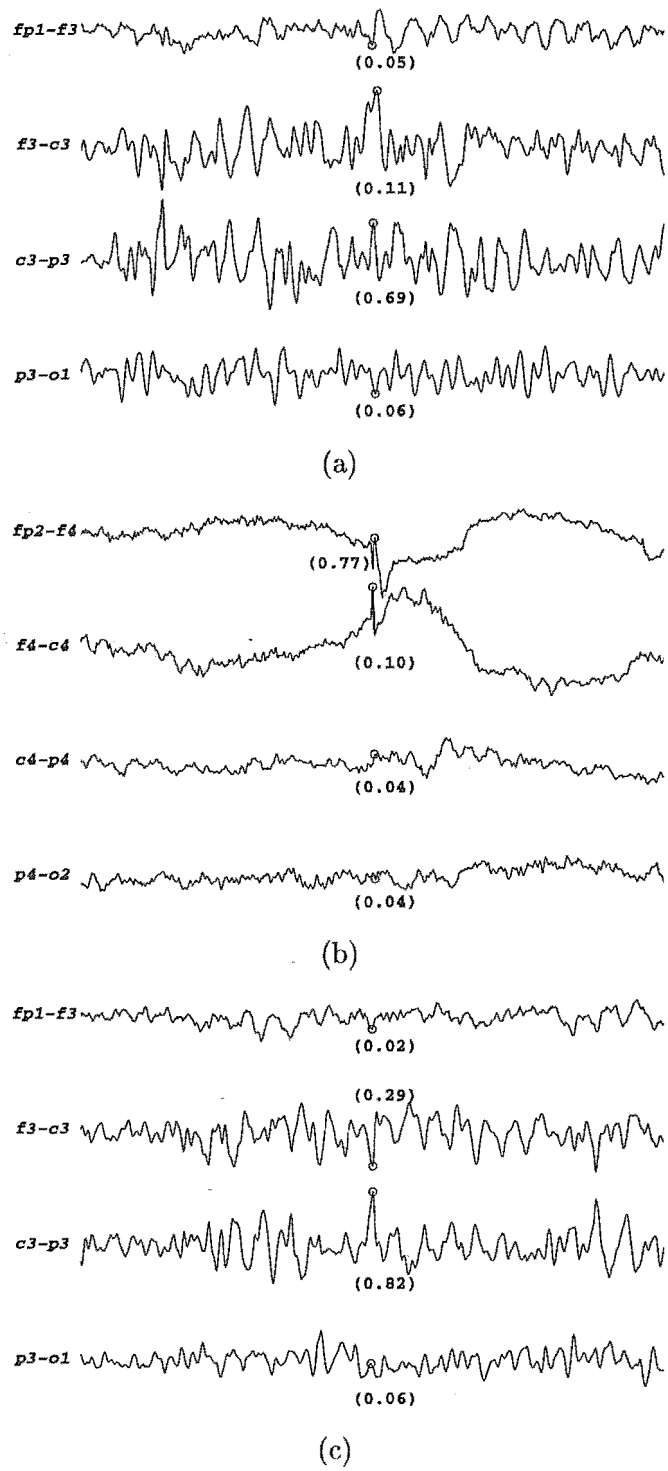


Figure 10.9 (a) An example of an artifact (sharp background) correctly rejected by the spatial-combiner following spatial reasoning. (b) & (c) depict examples of electrode 'pop' and sharp alpha artifact respectively accepted following spatial reasoning due to the high individual probabilities assigned to the 'spike-like' CEDs. (The values in brackets indicate the probability value assigned to each CED by the SOFM stage and the circle indicates the vertex of each CED in question).

probability. Although the CED was centered around the 'wrong' vertex, in each case the SOFM responded with a reasonable probability value for each CED. This seems to indicate that the SOFM stage is relatively insensitive to "jittering" in time of the input waveform making up the CED.

The 5 events missed entirely by the system consisted of both low amplitude CEDs that followed too closely to previous CEDs and were ignored by the mimetic stage. The 15 events assigned low probabilities consisted mainly of CEDs which were 'buried' in noise (EMG mainly) and, hence, were assigned low probabilities by the SOFM stage. Where one CED on an adjacent channel was assigned a relatively high probability, there was no spatial evidence to support a detection and the event was discarded.

Overall, measures taken to remove gross deviations in the EEG recording due to excessive muscle artifact, electrode movement and eye-blink appear to have been quite successful. Only 5 eyeblink artifacts were identified of the many present in the test EEGs, especially during hyperventilation segments. The EMG artifacts picked up consisted of local 'spiking' of the background EEG most probably due to individual motor neurons firing in facial muscles. The detection of electrode 'pop' and sharp background provided the greatest problem in false detections. In each case when viewed on a single channel basis, each CED was assigned a relatively high probability and, when coupled with the correct spatial distribution of the CEDs, resulted in the artifact being reported as an EV. Figure 10.9a depicts an artifact correctly rejected by the spatial-combiner as artifact. Figure 10.9b and Figure 10.9c depict two examples of false detections of artifact due to the 'spike-like' nature of the individual CEDs and their correct spatial arrangement.

The problem of false detections of the type just described is the major contributor to the low overall selectivity of the system as a whole. Possible methods of solving this problem are proposed next.

A reasonable question that may be asked is: Why does the SOFM stage attribute such high probabilities to some artifactual waveforms? On the whole, this is because when looked at on a single channel basis the CEDs are distinctly 'spike-like'. However, a number of non-epileptiform CEDs were not distinctly 'spike-like' but still resulted in large probabilities on presentation to the SOFM. As the SOFM works on the system of the 'closest matching' weight vector to find a 'winner', if an input waveform is under represented in the SOFM weights, then the closest matching weight vector for such an input may not necessarily be that 'close' (in a Euclidean sense) but still be the closest to be assigned the 'winner'. This suggests that there may be a need to re-calibrate the trained SOFM using a calibration set more representative of the various spike morphologies and, even more importantly, more representative of the wide range of 'spike-like' artifacts. The calibration set used to calibrate the SOFM in Chapter 9 was put together with careful attention paid to spike morphology but less so with the

various artifactual waveforms.

Another possible solution may lie in using temporal contextual information to influence the spike detection process based on past history of events on each channel in the multichannel recording – as discussed in Section 10.10.

One further method of handling such a problem would be to use a more *wide-sense* spatial analysis. The spatial-combiner performs spatial analysis based on a 4-channel bipolar chain of electrodes. However, no use is made to corroborate evidence of EVs by comparing the outputs of the sub-systems to upgrade or downgrade a detection based on the relative locations of each bipolar chain. For example, adjacent bipolar chains could be used to confirm strong individual chain probabilities, or chains that share a common electrode could be use to confirm the presence of artifacts. EEGers do just this in order to help confirm the presence of EVs.

When observing the spatial-combiner *per se*, the performance was reasonably good. This is especially so considering the somewhat *ad hoc* placement of the fuzzy input and output membership functions and generation of the fuzzy rule base, which are based more on our expectations of what should be happening than a precise mathematical model of the process. The performance can be altered by changing the shape of the constituent membership functions, but examining the results obtained these indicate that the current problems are not related to the fuzzy inference process and hence any improvements in performance would likely be minimal.

10.10 PROPOSED TEMPORAL UPDATING

In addition to the considerable use made of spatial contextual information when performing spike detection, EEGers also make use of wide-sense temporal information. The current spike detection system makes *considerable* use of spatial contextual information, as described in this chapter. Short-term temporal contextual information is also used in a limited fashion by having the mimetic stage extract the three contextual parameters describing the characteristics of the background EEG within a 1.0 s window around a CED. This temporal information is used to place the CED in context when presenting it to the SOFM for a probability to be assigned to it.

However, the EEGer also uses temporal information on a much wider scale than utilized in the current system. When the EEGer is grading an EEG and marking the presence of EVs, the location of the epileptiform activity seen is kept in mind and this information is used to influence his decision on EV/non-EV as he progresses through the EEG. The EEGer frequently uses this information on completion of a first viewing of the EEG to backtrack through the EEG and upgrade (or downgrade) EVs based on the history of activity throughout the EEG. Traditional expert-system based detectors capitalize substantially by incorporating temporal contextual information [Davey

et al. 1989], [Glover *et al.* 1989] and [Dingle *et al.* 1993].

The spike detection system presented here only makes use of the limited contextual temporal information surrounding each CED. The following system is proposed as a future addition to the spike detections system in order to include temporal information in the spike detection process.

The proposed temporal system hinges around the use of the Bayesian probability method to assign probabilities to each neuron in the SOFM during calibration (see Section 9.5.3.2). So far the SOFM has been calibrated "off-line" where the probabilities assigned to each neuron have been based on the information held in the calibration set. Once calibrated, an identical SOFM is assigned to each channel of the multi-channel recording in order to assign probabilities to each CED in a similar fashion.

The Bayesian system used to calibrate the SOFMs assigns the *posterior* probability to each neuron based on the *prior* probablity and the likelihood of that neuron being declared the 'winner' during calibration. As the calibration process progresses and a given neuron 'wins', its posterior probability is altered.

The system proposed here is to provide each channel SOFM with the ability to alter the posterior probabilities to each 'winning' neuron of the SOFM based on the overall outcome of the spike detection system. However, the probabilities must only be altered if the level of detection exceeds a given threshold such that

$$\psi_c^j(k+1) \triangleq \frac{s_c^j(k+1)+1}{n_c^j(k+1)+2} = \frac{(s_c^j(k)+1)+1}{(n_c^j(k)+1)+2} \quad \text{if threshold exceeded,} \tag{10.2}$$

$$\psi_c^j(k+1) = \psi_c^j(k) \quad \text{otherwise,} \tag{10.3}$$

where $\psi_c^j(k+1)$ represents the new value of the posterior probability ψ (as described in Section 9.5.3.2) for 'winning' neuron c in SOFM j $(1 \leq j \leq 16)$ at the discrete epoch of time k; s_c^j and n_c^j represent the total number of successful 'wins' and the total number of 'wins' respectively for neuron c in SOFM j.

This system is proposed in order to strengthen the probabilities assigned to neurons most responsive to true CEDs differently for each channel of the EEG such that each SOFM on each channel of the system should become selectively tuned to the particular waveforms on each channel that make up an EV for each particular patient. Before spike detection commences, the initial probabilities assigned to each SOFM should be the probabilities assigned by the calibration process.

Careful consideration must accompany the choice of the threshold level, it is most likely that assigning a value too large will result in minimal updates to the posterior probabilities due to the already large probablities assigned to the constituent CEDs making up the detected EV. Conversely, it is equally likely that a threshold too low will result in the posterior probabilities of too many neurons in the SOFMs being altered

resulting in increased false detections due to unwanted strengthening of probabilities related to artifactual EEG. Of the two extremes, opting for a threshold value closer to the higher threshold seems be the safer choice, hence minimizing the risk of increasing false detections.

It could also be possible to make the system into a two pass system, whereby the final probability values assigned to each SOFM at the end of the first pass through the data could be used during a second pass of the entire EEG. In this way the SOFMs become 'tuned' to the characteristics of the EEG being tested. [Glover *et al.* 1989] implemented an expert-system based detector which involved the concept of 'on-line' learning discussed here.

10.11 SUMMARY

This chapter introduced the use of spatial contextual information in performing spike detection and the concept of using temporal contextual information to aid in that process. Through the use of fuzzy logic, an approximation to the spatial reasoning used by EEGers in the spatial combination of the single channel EEG was possible. This avoids the need for exact mathematical models to represent the distribution of EVs across channels, and maximizes the use of the probabilistic nature of the outputs from the single channel SOFM stages.

The fuzzy rules implemented in this chapter were derived on our expectations of allowable combinations of single channel CEDs and resulted in the formation of a fuzzy rule base consisting of 127 such rules. The rules derived have performed quite well.

The system implemented in this chapter highlighted several problems when tested, the most crucial being the relatively high number of false detections of EVs. These are mostly due to 'ED-like' artifacts exhibiting 'allowable' spatial distribution. This seems to indicate that the SOFM stage may benefit from re-calibration using a calibration set which is more representative of the spectrum of ED and 'ED-like' waveforms. The majority of the missed events were primarily due to the wrong polarity being assigned to CEDs, despite their being assigned strong probabilities by the SOFM stage (and despite the measures taken as described in Section 10.7.2). Furthermore, the possibility of using a more wide-sense spatial context could confer improved performance.

Chapter 11

SYSTEM PERFORMANCE

11.1 INTRODUCTION

This chapter looks at the spike detection system that has been developed and examines the performance of the system at each stage. A direct comparison is then made with the Christchurch-based spike detection system developed by Dingle *et al.* [1993] using the same test data for both systems. Finally, the overall performance of the spike detection system is compared with that of a number of other leading spike detection systems reported in the literature.

11.2 PERFORMANCE AT EACH STAGE

The experimental spike detection system which has been implemented does not include the spike enhancer of Chapter 8 and, hence, the performance of the system is assessed: (a) after the mimetic stage, (b) after the SOFM stage and (c) at the output of the spatial-combiner. For both (a) and (b) above, no spatial contextual information is present, whereas (c) uses spatial information to act on the outputs of (b) making the final EV/non-EV decision.

The system has been developed using the MATLAB (Vers 4.2c1) package (The Math Works Inc.) with the Neural Network toolbox (Vers 2.0b) and the Signal Processing toolbox (Vers 3.0b). The final system used to perform the tests described here were further developed using the 'C' programming language. The tests were performed on a PC system with a Pentium processor running at 90 MHz. Spike-detection was performed 'off-line' on data stored on hard disk. On average, the system works in 'real-time', i.e., spike detection for a 20 minute EEG lasts about 20 minutes, but for EEGs with 'noisy' backgrounds the spike detection process can take twice as long. This is due to the large number of CEDs put forward by the mimetic stage for 'noisy' EEGs.

In order to view the advantages of each stage of the multistage system, both the sensitivity and selectivity are calculated at each stage for the same test data used in

Chapters 9 and 10. In addition, an EEG which contained no epileptiform activity (as graded by all 3 EEGers) was added to the previous test set of 6 EEGs as shown in Table 11.1. The performance at each stage was calculated as follows:

1. Mimetic stage: a detection was considered to have taken place if at least one CED on any of the 16 channels of EEG passed the thresholds and, hence, was forwarded to the SOFM stage.

2. SOFM stage: if the probability assigned to at least one CED on any of the 16 channels of EEG by each SOFM stage exceeded a threshold level of $d_t = 0.5$ then the CED was counted as a detection.

3. Spatial combiner: if an output was assigned any one of the fuzzy variables POS, PRO or DEF, the CED was counted as a detection.

EEG	Age	Duration	EEGers	Epileptiform Events (EVs)		
				Definite	Quest.	Total
1	71	25m 8s	2	17	17	34
2	3	16m 3s	2	8	17	25
3	31	24m 59s	2	2	19	21
4	11	23m 24s	2	2	1	3
5	5	27m 24s	3	9	0	9
6	24	26m 17s	2	25	16	41
7	28	25m 16s	3	0	0	0
Totals		2h 48m 31s		63	70	133

Table 11.1 The test set comprising 6 definite epileptiform EEGs (with a total of 63 definite and 70 questionable EVs) and 1 normal EEG (i.e., no epileptiform activity).

Table 11.2 gives the performance at each stage of the system for each patient in the test set for a [20 × 20] SOFM and Table 11.3 gives the performance across all six patients as the SOFM size varies from [10 × 10] to [20 × 20].

From Table 11.2 it can be seen that for the EEG of patient 7, 5 false detections took place at the overall output of the system which, if taken at face value, would have resulted in the EEG being incorrectly reported as containing epileptiform activity. The 5 false detections were due to sharp background and muscle spikes with low individual probabilities but with strong focal characteristics. For each patient, the mimetic stage picked up a very large number of CEDs giving high sensitivities and correspondingly low selectivities. This is not surprising as the thresholds for the mimetic stage were set such that it performed a screening of the incoming EEG with a high sensitivity to true EDs. Looking at the output of the SOFMs above a probability threshold of $d_t = 0.5$ resulted in an increase in selectivity due to a considerable reduction in the number of false detections, but also resulted in a reduced sensitivity as CEDs with relatively

Sensitivity/Selectivity (%)							
		Mimetic		SOFM ($d_t = 0.5$)		Spatial-combiner	
Patient	EEGer	Sen/Sel	System	Sen/Sel	System	Sen/Sel	System
1	(34)	100/2	(2123)	26/28	(32)	56/68	(28)
2	(25)	96/3	(890)	40/14	(70)	56/25	(57)
3	(21)	100/2	(887)	38/9	(91)	57/22	(55)
4	(3)	100/0	(977)	67/3	(74)	67/9	(22)
5	(9)	100/1	(1624)	89/10	(79)	78/17	(42)
6	(41)	88/3	(1192)	56/35	(66)	59/59	(41)
7	(0)	−/0	(1632)	−/0	(28)	−/0	(5)
Total	(133)	95/2	(9325)	45/14	(440)	59/31	(250)

Table 11.2 The sensitivities and selectivities of the system at each stage to definite & questionable EVs for each patient. (The SOFM stage values are for a [20 × 20] SOFM followed by fine-tuning with LVQ2).

Sensitivity/Selectivity (%)							
		Mimetic		SOFM ($d_t = 0.5$)		Spatial-combiner	
SOFM	EEGer	Sen/Sel	System	Sen/Sel	System	Sen/Sel	System
[10 × 10]	(133)	95/2	(9325)	34/14	(323)	48/25	(254)
[12 × 12]	(133)	95/2	(9325)	37/13	(381)	37/28	(178)
[14 × 14]	(133)	95/2	(9325)	35/13	(363)	44/31	(187)
[16 × 16]	(133)	95/2	(9325)	41/12	(449)	50/25	(259)
[18 × 18]	(133)	95/2	(9325)	41/13	(413)	53/22	(321)
[20 × 20]	(133)	95/2	(9325)	45/14	(440)	59/31	(250)

Table 11.3 The sensitivities and selectivities of the system across all test patients at each stage, to definite & questionable EVs for the size of the SOFM stage varying between [10 × 10] and [20 × 20]. (The SOFM was followed by fine-tuning with LVQ2).

low individual probabilities were discarded. The spatial combiner further increased the selectivity and increased the sensitivity as the number of false detections was reduced when the data was observed using spatial analysis. This trend was repeated for each patient in the test set.

When looking at the overall performance of the test data as a function of the SOFM size (Table 11.3), for each SOFM tested the performance is seen to increase at each stage and reaches the best balance between sensitivity and selectivity at the spatial combiner stage. As discussed in Chapter 10 the relatively low sensitivities are mainly due to the assignment of wrong polarities to the CEDs before the spatial-combiner stage. A consequence of the low selectivity described in Chapter 10 is that the system graded a non-epileptiform EEG as epileptiform, although for all SOFM sizes the number of false detections for the normal EEG did not exceed 5.

11.3 COMPARISON WITH OTHER SYSTEMS

In order to better assess the performance of the system, a comparison is needed with some of the various spike detection systems reported in the literature. However such comparisons between systems are made difficult by the wide range of measures used for evaluating their performance.

The greatest differences in assessing the performance occur when obtaining a measure for false detections and missed detections. Gotman and Wang [1992] define false detections as detections which are obviously artifact, whereas other methods include defining a false detection as an event not marked by any of the 6 EEGers who graded the system [Eberhart *et al.* 1989] or marked by fewer than 6 of 7 EEGers [Fischer *et al.* 1980].

In a similar way Gotman and Wang [1992] states that missed detections are those falsely rejected by the system as non-epileptiform, but ignores EVs/EDs missed by the system altogether. In contrast, Eberhart *et al.* [1989] identified missed detections as those not detected by the system but marked by at least 4 of 6 EEGers.

The performance measures used for this system have been adapted from Webber *et al.* [1994] and, where possible, will be used to compare between systems. In addition, a direct comparison of the system described herein with that of Dingle *et al.* [1993] is possible using the same 7 test EEGs for both systems. In addition to the same performance measures, a further measure of performance is introduced — the number of false detections per hour.

The system of Dingle *et al.* [1993] uses a mimetic stage followed by an expert system and makes considerable use of both spatial and temporal contextual information in the spike detection process. Table 11.4 shows the performance of both systems on the 7 test EEGs of Table 11.1 (an SOFM size of [20×20] was used for SOFM-based spike detection

system). From the table it can be seen that the system of Dingle *et al.* [1993] is the more conservative of the two systems resulting in considerably fewer false detections than the SOFM-based system and hence a higher overall selectivity (78% vs 31%). Conversely, the cost of the higher selectivity is a much lower sensitivity (21% vs 59%). This low sensitivity resulted in zero detections achieved for patient 4; the EEG was consequently classified as 'normal' in error. Although the EEG of patient 5 was correctly marked 'epileptiform' by the system of Dingle *et al.*, this was inadvertently due to 3 false detections as none of the 9 EVs marked definite were detected! However, for the normal patient (patient 7) no detections were made, correctly identifying patient 7 as a 'normal' EEG, whereas the SOFM-based system falsely detected 5 artifacts which would have led to this EEG being incorrectly regarded as 'epileptiform'.

Sensitivity/Selectivity (%)									
		Dingle *et al.*				James *et al.*			
Patient	EEGer	Correct	False	Total	Sens/Sel	Correct	False	Total	Sens/Sel
1	34	7	0	7	(21/100)	19	9	28	(56/68)
2	25	5	2	7	(20/71)	14	43	57	(56/25)
3	21	3	2	5	(14/60)	12	43	55	(57/22)
4	3	0	0	0	(0/–)	2	20	22	(67/9)
5	9	0	3	3	(0/0)	7	35	42	(78/17)
6	41	13	1	14	(32/93)	24	17	41	(59/59)
7	0	—	0	0	(–/–)	—	5	5	(–/–)
Total	133	28	8	36	(21/78)	78	172	250	(59/31)

Table 11.4 A comparison between the sensitivities and selectivities of the SOFM based spike detection system and that of Dingle *et al.* [1993] to definite & questionable EVs for each patient in the test set. (The [20 × 20] SOFM after fine-tuning with LVQ2 was used in the SOFM stage.)

The measure of number of false detections per hour of EEG can be used to place the reported performance of the system into context when considering the length of EEGs considered in the test sets. For the test set above, the SOFM-based system resulted in a false detection rate of 61 false detections/hour whereas that of Dingle *et al.* [1993] resulted in 3 false detections/hour.

Table 11.5 shows a number of spike detection systems found in the literature listing the method used in each case. Table 11.6 compares the performance of the SOFM based spike detection system to the performances reported in the literature for the spike detection systems shown in Table 11.5. In each case the performance for EVs marked either definite or questionable by the EEGer(s) is calculated.

The results given in Table 11.6 show a great variability in the measures of performance, especially false detection rate. For the methods of Gotman and Wang [1992] no measures for the sensitivity are given or deducible from the data given although for the method involving wide temporal context (state) a reduction of 15% in the total number of detections is reported along with the increase in true detections. Of the systems described, only those of Gotman and Wang [1992], Dingle *et al.* (1997) and the

	System	Method
1	Özdamar *et al.* [1991]	ANN + ANN
2	Hostetler *et al.* [1992]	Mimetic
3	Gotman and Wang [1992]	Mimetic
4	Gotman and Wang [1992]	Mimetic + state
5	Webber *et al.* [1994]	Mimetic + ANN (Parameters)
6	Webber *et al.* [1994]	Mimetic + ANN (Raw EEG)
7	Dingle *et al.* [1993]	Mimetic + expert system
8	Dingle *et al.* (1997)	Mimetic + expert system
9	James *et al.* (1997)	Mimetic + SOFM + Fuzzy logic

Table 11.5 The spike detection systems found in the literature (systems 1–7) to be compared with the Christchurch based system (#8) and the SOFM-based spike detection system (#9).

System	EEGs	Hours	Epileptic (%)	EEGers	Train/ Test	Sen (%)	Sel (%)	False /hour
1	10	0.043	?	4	6/4	90	69	~1023
2	5	2	100	5	Blind	59	89	37
3	20	33	100	2	Blind	—	41	117
4	20	33	100	2	Blind	—	67	47
5	10	0.3	100	1	Same	74	74	~804
6	10	0.3	100	1	Same	46	46	~5598
7	11	3	73	1	Same	53	100	0
8	7	2.8	86	2/3	Blind	21	78	3
9	7	2.8	86	2/3	Blind	59	31	61

Table 11.6 A comparison of the sensitivities, selectivities and false detection rates between the SOFM based spike detection system and others in the literature. (The [20 × 20] SOFM after fine-tuning with LVQ2 was used in the SOFM based system.) Refer to Table 11.5 for the names of the systems.

system described herein use a totally new set of test data to validate the performance of the system. Gotman and Wang [1992] use the largest number of EEGs giving a total of 33 hours of recordings. Of all of the systems, only Dingle *et al.* [1993] and the system described herein have included normal EEGs in their tests. Webber *et al.* [1994] tested their system on the EEGs obtained from 10 patients (the same EEGs were used for training), and report satisfactory sensitivity and selectivity for the mimetic+ANN (parameters) case (both 74%), but at a cost of around 804 false detections/hour. For their performance using 'raw' EEG instead of parameters, both the sensitivity and selectivity were relatively low (both 46%), with an even higher false detection rate (over 5000 per hour). These extremely high rates of false detections may be partly due to the short length of EEGs tested (20 minutes total across 10 EEGs). Özdamar *et al.* [1991] report similarly good results for sensitivity and selectivity but their method resulted in a similarly high false detection rate (~1023/hour). Hostetler *et al.* [1992], which perform an independent evaluation of Gotman's [1978] system, report a much lower false detection rate (37 false detections/hour) but this was achieved at a cost of a relatively low sensitivity (59%). Gotman and Wang [1992] report a similar false detection rate for their system (which estimates the sleep stage of a subject during recording to improve performance). In contrast, an impressive false detection rate of *zero* is reported by Dingle *et al.* [1993], albeit with a reasonably low sensitivity (53%), for a data set which included 3 normal EEGs. However, the system used the training data to test the system (as Dingle *et al.* [1993] use an expert system there is no 'training set' as such but the system can still be said to be tailored, and hence biased, towards the data used). When tested with the novel data set used in this thesis, the false detection rate rose to 3/hour with a rather low sensitivity (21%).

The work of Dingle *et al.* [1993] has been reassessed with more EEGs by Jones *et al.* [1994] and Jones *et al.* [1996]. Jones *et al.* [1994] report on a study involving 148 EEGs (41 epileptiform and 107 normal) giving 49 hours of EEG. Performance on a *global EEG* level (i.e., performance is measured on the quantities of *entire EEGs* graded epileptiform or normal as opposed to individual events within each EEG) is reported at a sensitivity and selectivity of 100% giving *zero* false detections per hour. The system was, however, tested on the same EEGs used for training. Jones *et al.* [1996] report results of a major clinical study involving 521 EEGs (50 epileptiform and 471 normal) giving 173 hours of EEG. A sensitivity of 95% and selectivity of 72% are reported at the global EEG level, giving an estimated false detection rate of 0.29 per hour. All 521 EEGs were new to the system. Unfortunately, no values are given for the performance at an individual event level making a comparison to these studies not possible.

The data used to test the current system was reasonably long (2.8 hours), involved a normal EEG and was novel to the system. The overall values of 59% for sensitivity and 31% for selectivity, with a false detection rate of 61/hour compare reasonably well to other systems when one places the values reported by other systems in context.

One must also bear in mind that a sensitivity of 85% would be attained by 'simply' correcting the recognized CED polarity problem (see Chapter 10). Furthermore, the spatial combiner is still (for the most part) experimental. The membership functions and fuzzy rules have been derived according to our initial expectations of how they should be represented. There has been no 'fine-tuning' of the rules (i.e., adding or removing rules) or membership functions (i.e., changing the shape and/or parameters) attempted as yet. It is not considered unreasonable to expect increases (especially in the selectivity) in performance after fine-tuning the spatial combiner stage.

The benefits which might be derived from wide-sense temporal information are clearly recognized as an important addition to the spike detection system; the same applies to the additional spatial cues which are not being taken advantage of at the moment (as described in Chapter 10). The method of raising and lowering thresholds within the spike detection system dependent on the state of awareness of the subject is also noted and has yielded positive results for Gotman and Wang [1992].

Chapter 12

CONCLUSIONS AND FUTURE RESEARCH

12.1 ARTIFICIAL NEURAL NETWORKS

Throughout the relatively short history of ANNs, engineers have realised what important tools ANNs could be in engineering, especially in the field of signal processing. The ability of ANNs to embody knowledge in the relatively few weights of the network is an important feature of ANNs. The ability of the ANN to 'learn' from a set of training examples and then to generalize to new examples makes the ANN an appealing alternative to expert systems which try to embody the knowledge of an expert in a given field through the use of many rules. Although there is no real comparison between the processing power of the human brain, with its billions of synapses, and its 'artificial' counterpart, an ANN with a few hundred 'artificial' neurons still has a remarkable ability to perform, for example, pattern recognition. Finally, due to the inherent parallel structure of ANNs they are quite fast in operation, an important aspect for many engineering applications.

The two main types of learning in ANNs which dominate in the literature are supervised and unsupervised (or self-organised) learning. With supervised learning the weights and biases of the ANN are altered with the help of an external 'teacher' (but, in contrast to expert systems, without the need to determine precisely what rules/cue are used by the teacher). The choice of exemplars used to train the ANN will affect the ability of the ANN to generalize to novel input data. The most popular of the supervised ANNs in the literature is the multi-layer perceptron (MLP), which can be trained through the use of the error-backpropagation algorithm. In order to train an MLP well for a given problem, not only must the input data set be representative of the inputs to be presented to the MLP in operation, but the parameters which define the architecture of the ANN must be chosen with care. As yet, there are no precise guidelines/formulae to obtain exact values for these parameters, currently, 'rules-of-thumb' and 'trial-and-error' are the norm.

With self-organised learning, no external 'teacher' is required in order to adjust the weights of the ANN, but the choice of the input data set will still reflect the

generalizing ability of the ANN (as for the supervised case). An important advantage of self-organising ANNs over their supervised counterparts is that they can be exposed to and make use of vast quantities of input data for training purposes without the need for assigning 'labels' to each input forwarded to the ANN. This means that the resulting ANN trained with such data can be more representative of the underlying structure in the input set than its supervised counterpart trained on a much smaller set of 'labelled' inputs.

One such ANN is the self-organising feature map (SOFM) which presents a method of identifying similarities (or features) in a vast (unlabelled) training set. What is more, the features are arranged spatially such that there is topological ordering in the neurons which make up the ANN. Since the SOFM is made up of a single layer of neurons arranged in the form of a lattice (1D or 2D lattices are normally implemented for ease of visualisation) the operation of a trained SOFM is remarkably fast and the computations involved on presentation of an input to the SOFM are simple distance calculations. Training times for such ANNs tend to be longer than their supervised counterparts but this is primarily due to either the large inputs sets used or the large number of times a small input set is presented to the network during training.

On reviewing MLPs as pattern classifiers in particular, the choice of training parameters falls into two categories. The first is the choice of the MLP architecture (i.e., number of hidden layers, number of hidden neurons, type of activation function) and the second is the choice of the training parameters (i.e., the type of learning algorithm, learning rates, momentum factors, etc.). Both in the literature and in this thesis it has been shown that the parameters of the MLP require careful choice and require a certain amount of 'trial-and-error'. In contrast the SOFM consists of a single layer of neurons and requires fewer training parameters. The main parameters required are the SOFM size (i.e., the number of neurons) and the decay parameters for the learning rate and neighbourhood size. On performing the pattern classification simulations with the SOFM in Chapter 6 a number of conclusions can be made. The 'rule-of-thumb' values (as suggested by Kohonen) for the major SOFM training parameters result in trained SOFMs which, when followed by LVQ techniques, perform pattern classification with acceptable performance (when assessing sensitivity and selectivity). More importantly, small variations in the training parameters are less likely to affect the performance of the trained system; which cannot be guaranteed for the MLP. Another important conclusion is that the SOFM training is relatively insensitive to the dimensions of the input data, the MLP, on the other hand, requires a change in architecture to accommodate different sized inputs and hence a different set of training parameters. Although no method exists for accurately determining the 'right' size of the SOFM for the task at hand, by observing the measure introduced which gives the mean change in the Euclidean distance as training progresses ($\delta(k)$) for different sized SOFMs, in conjunction with the performance measures, it is possible to obtain an idea of the minimum sized

SOFM for the task at hand.

The conclusions put forward following the computer simulations of Chapter 6 provide useful starting values for training an SOFM for pattern classification purposes. Overall, the simulations show that the SOFM is relatively insensitive to changes in the major training parameters. This, however, has not been shown with any statistical significance. A proposal for future research into using the SOFM for pattern classification is to perform a full statistical analysis of the results obtained following a similar set of simulations. In this way each parameter examined can be assigned a recommended value with a corresponding degree of confidence.

12.2 SPIKE DETECTION IN THE EEG

The EEG has been proven as a valuable tool in the diagnosis of many brain disorders. Such disorders are usually diagnosed through the presence of abnormal EEG waveforms in the EEG recording. A normal EEG can be defined as an EEG which exhibits no abnormal waveforms. However, there is a wide variety of normal patterns for EEGs of persons the same age and an even wider variety over various age groups. The state of the person during recording also influences the appearance of the EEG. In short, processing the EEG can be divided into two categories: analysing and looking for abnormalities in (a) the background EEG and (b) transient/non-stationary activity in the EEG. The most frequent clinical use of the EEG is for detection of epileptiform activity.

This thesis deals with the problem of detecting epileptiform activity in the interictal EEG (i.e., abnormal activity occurring between seizures). An automated method is sought to detect epileptiform discharges (or spikes) in the EEG. Many attempts have been made in the literature to solve this problem but all have achieved only a limited success. The main problem lies with the extreme difficulty met in attempting to eliminate considerable numbers of false detections due to 'spike-like' artifacts and sharp background activity in the EEG. The EEGer makes *considerable* use of spatial and temporal information when visually performing spike detection to influence his decision in the process. Surprisingly, few systems incorporate any aspects of such reasoning in their spike detection algorithms.

This thesis approaches the spike detection problem through the use and attributes of ANNs. In particular, the ability of ANNs to be trained to solve problems and their ability to generalize to novel data once training is complete are drawn upon to the advantage of the spike detection system. The spike detection problem is broken down into two major components. The first component is the spike-enhancer whose sole aim is to enhance the presence of 'spike-like' transients in the EEG by attenuating the surrounding background EEG. The second component involves detecting the presence

of, and classifying, spikes in the EEG. Both stages have been developed and tested independently but, at this stage, have not been tested jointly.

For both components, the spatial information inherent in the multichannel EEG recording is made use of as much as possible and a limited amount of temporal contextual information is also used.

The spike enhancer has been developed through the use of a MLP trained with the error backpropagation algorithm utilising variable learning. In effect, the MLP forms an adaptive filter which adapts 'on-line', utilising spatial and limited temporal information to adaptively cancel the background EEG from a primary channel of EEG through a technique known as multi-reference adaptive noise cancelling (MRANC). As the correlated background EEG on the primary channel is cancelled, any waveforms present on the primary channel which are uncorrelated with the background EEG on the other channels are left behind (with some distortion taking place).

The system has been implemented experimentally and has been tested on the data of six patients. The spike enhancer results in a substantial increase in the SNR of confirmed spikes but at a cost of slight distortion of the spike itself. Non-linear implementation of the spike enhancer resulted in superior performance (121% on average) when compared to its linear counterpart (76%). In effect, the MRANC system implements a variable HPF which alters its characteristics to follow the slowly changing EEG for a given patient and can completely adapt to the characteristics of each new patient.

The system was implemented such that the MLP constantly trains in an 'on-line' manner, with variable learning used to increase the speed of convergence. For each primary channel the remaining 15 channels (of the 16 channel recording) are used as reference channels. This results in the system being slow to process segments of EEG and, hence, unable to process EEG in 'real time' as it stands (i.e., $\sim \frac{1}{6}\times$realtime for each channel of EEG). Hardware implementation of MRANC would be possible through the use of one of the many DSP ICs available on the market as a means of speeding up the process and is a possible avenue for future work in this area. More importantly, a study of the correlation of the background EEG as measured at various locations on the scalp could be an important area of research. If it were possible to gain a better understanding of the background EEG, especially as a function of its location on the scalp, it would perhaps be easier to analyse 'abnormal' waveforms superimposed on the background EEG. In particular for the spike enhancer, a better knowledge of the background EEG characteristics as function of its location would be important in choosing the channels to be used as reference for each primary channel in turn.

The choice of the MLP to represent the adaptive filter carries with it the inherent difficulties of choosing the 'right' network and training parameters. Further research into MLP ANNs in particular may yield methods of choosing the 'right' parameters for

the problem at hand. Alternatively choosing another ANN for the adaptive filter may yield similar results with, maybe, less dependence on ANN parameters.

The spike detector/classifier is based around the SOFM. The capability of the SOFM to extract identifying features from large amounts of input data is drawn upon heavily in this stage. This is particularly useful in the spike detection problem where there is a great difficulty in accurately grading large numbers of candidate epileptiform discharges (CEDs) due to the amount of disagreement between EEGers. Using the self-organising abilities of the SOFM it is possible to train an ANN with a large number of EEGs known to contain EDs and then use only a selected sub-set of EDs (which have been graded with a high degree of agreement amongst EEGers) to accurately label each weight abstracted from the input data by the SOFM. The SOFM stage is preceded by a mimetic stage in order to screen the incoming EEG and reduce the number of CEDs presented to the SOFM stage. More importantly the mimetic stage gives a time reference to each CED such that it is presented to the SOFM stage with its vertex positioned at the same point in each case.

Each mimetic/SOFM stage works independently on each channel of EEG (i.e., utilising no spatial information). This thesis also introduces the novel use of assigning a probability to the output of the SOFM whereby a probability is assigned to each CED on each channel indicating the probability that a particular CED is a true ED.

Finally, a spatial-combiner stage is presented which groups the single-channel probabilities according the (bipolar) montage in use and forms decision on the detection of an epileptiform event (EV) based on the spatial distribution of the CEDs. This is implemented through the use of fuzzy logic. Fuzzy logic allows a set of rules to be drawn up which describe our knowledge of how an EV is manifested spatially across the 16 channel EEG recording, without the need for an accurate mathematical model describing the process. This allows for a method that incorporates the spatial reasoning an EEGer uses in detecting EVs in the multichannel recording.

A system has been implemented which performs the above in approximately 'real time'. To date, the system has been evaluated on 7 novel EEGs. Overall, the system has a sensitivity of 59% and a selectivity of 31%, with an average of 61 false detections per hour. As preliminary results, and when compared with the results of other spike detection systems in the literature, this performance is very encouraging. Although the selectivity is low, the corresponding number of false detections (61/hour) is far less than that of systems in the literature describing higher values of selectivity. Several factors have been identified which are known to have resulted in a lower sensitivity and selectivity than would otherwise have been the case. In particular the major factor contributing to missed detections was due to the mimetic stage picking up the constituent CEDs of a given EV and then assigning the wrong polarity to one or more of the CEDs resulting in the CEV being rejected on spatial grounds by the spatial-

combiner. This was despite the *ad hoc* method employed to correct the problem as described in Chapter 10. The false detections were mainly due to artifacts which were assigned a relatively high probability value by the SOFM stage and showed particularly strong focal characteristics.

Future research includes finding an alternative method to assign the polarities of the constituent CEDs of a given CEV. The problem with the false detections seems to indicate that the trained SOFM could benefit from recalibration using a 'revised' calibration data set. The revised data set should place more emphasis on 'spike-like' artifacts rather than mainly on waveforms representing the various spike morphologies.

The spatial-combiner stage is intended as a 'proof of principle' study and has indeed resulted in confirming the utility of such an approach in the spatial analysis of the EEG. The shape and placement of the input and output membership functions have, so far, been based on the ease of implementation of the functions and our initial expectations of the location of each variable. Future research should involve an analysis of the effect on performance of varying the shape and characteristics of the membership functions for both the input and output fuzzy variables. Further research should also include a closer examination of the derived fuzzy rules with the aim of (a) 'pruning' any rules which may be superfluous to the operation of the spatial-combiner and (b) in order to see if any allowable spatial combinations are not covered by the fuzzy rules.

An alternative approach to determining each of the required characteristics of each component of the fuzzy spatial-combiner could be a 'fuzzy' ANN implementation of the whole process. Such an ANN would in essence embody the membership functions and spatial rules in the weights of the trained ANN without the explicit need of knowing each characteristic.

The use of temporal contextual information to influence the spike detection process has already been stressed in Chapter 10 where a proposal has been put forward and is a definite avenue of future research. This 'wide-sense' temporal contextual information could also benefit from information regarding the state of the subject during the EEG recording as used by Gotman and Wang [1992] and should be examined as a possible addition to the temporal analysis of the EEG.

Overall, although still at a preliminary stage, such a system will be an important tool in the automatic detection of epileptiform activity in both routine and long-term EEG recordings. In particular the system benefits from a number of important features, which include: (a) Enhancing the presence of spikes in the background EEG (via MRANC) has the potential of increasing the performance of a spike detection system. (b) The use of the SOFM as a pattern classifier allows for an accurate description of the various spike-morphologies to be derived from a large amount of data in a self-organised manner. (c) By assigning probabilities to each CED via the Bayesian based modified calibration method, the uncertainty inherent in the spike detection process

is embodied at the single channel level. (d) Through the use of a fuzzy rule-base it is possible to perform spatial analysis on the single channel data in a manner which emulates the EEGer's way of thinking without the need of deriving an accurate model which describes the process.

Appendix A

EEG RECORDING MONTAGES AND PROTOCOLS

A.1 BIPOLAR AND REFERENTIAL MONTAGES

The EEG is recorded using either bipolar or referential montages. The electrode placement is based on the international 10–20 electrode placement system as shown in Figure A.1.

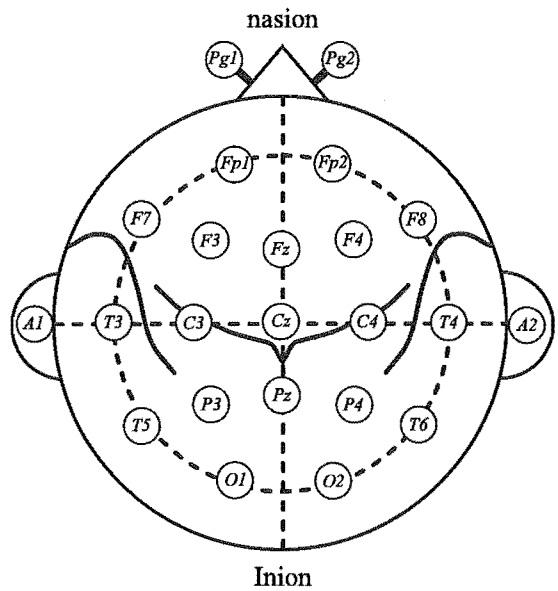


Figure A.1 The 10–20 system of electrode localisation.

Four bipolar montages are used in this system and these are:

1. Longitudinal: $fp2$-$f4$-$c4$-$p4$-$o2$, $fp1$-$f3$-$c3$-$p3$-$o1$, $fp2$-$f8$-$t4$-$t6$-$o2$, $fp1$-$f7$-$t3$-$t5$-$o1$.

2. Transverse: $f8$-$f4$-fz-$f3$-$f7$, $a2$-$t4$-$c4$-cz-$c3$-$t3$-$a1$, $t6$-$p4$-pz-$p3$-$t5$, fz-cz-pz.

3. Longitudinal–transverse: $f8$-$f4$-fz-$f3$-$f7$, $t6$-$p4$-pz-$p3$-$t5$, $fp2$-$f4$-$c4$-$p4$-$o2$, $fp1$-$f3$-$c3$-$p3$-$o1$.

4. Circumferential: $f4$-$c4$-$p4$-$p3$-$c3$-$f3$-$f4$, $fp2$-$f8$-$t4$-$t6$-$o2$-$o1$-$t5$-$t3$-$f7$-$fp1$-$fp2$.

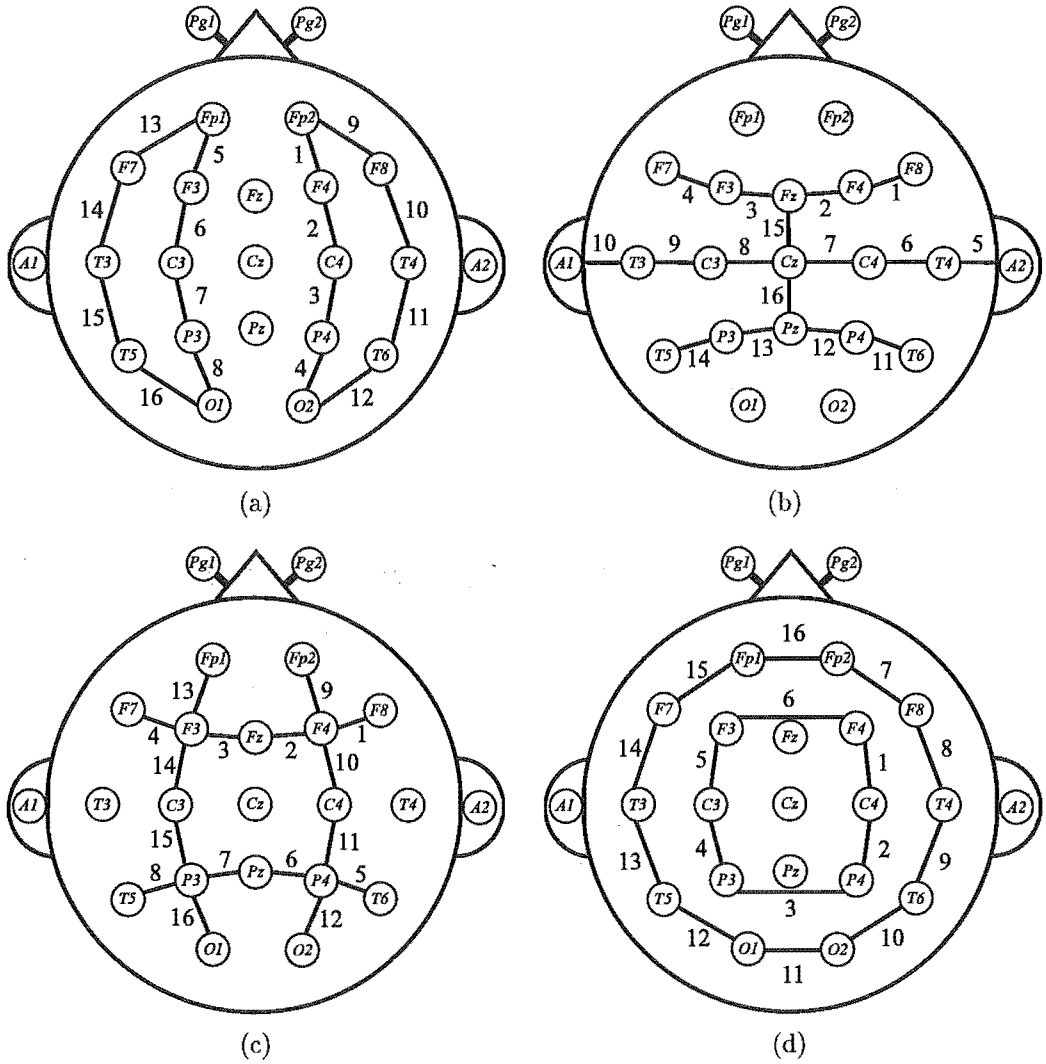


Figure A.2 The four bipolar montages used in this system. (a) Longitudinal, (b) Transverse, (c) Longitudinal–transverse and (d) circumferential.

Three referential montages are used and are given by:

1. Average: $fp2$-Ave, $f4$-Ave, $c4$-Ave, $p4$-Ave, $o2$-Ave, $fp1$-Ave, $f3$-Ave, $c3$-Ave, $p3$-Ave, $o1$-Ave, $f8$-Ave, $t4$-Ave, $t6$-Ave, $f7$-Ave, $t3$-Ave, $t5$-Ave.

2. Ipsilateral-ears reference: $fp2$-$a2$, $f4$-$a2$, $c4$-$a2$, $p4$-$a2$, $o2$-$a2$, $fp1$-$a1$, $f3$-$a1$, $c3$-$a1$, $p3$-$a1$, $o1$-$a1$, $f8$-$a2$, $t4$-$a2$, $t6$-$a2$, $f7$-$a1$, $t3$-$a1$, $t5$-$a1$.

3. Vertex reference: $fp2$-cz, $f4$-cz, $c4$-cz, $p4$-cz, $o2$-cz, $fp1$-cz, $f3$-cz, $c3$-cz, $p3$-cz, $o1$-cz, $f8$-cz, $t4$-cz, $t6$-cz, $f7$-cz, $t3$-cz, $t5$-cz.

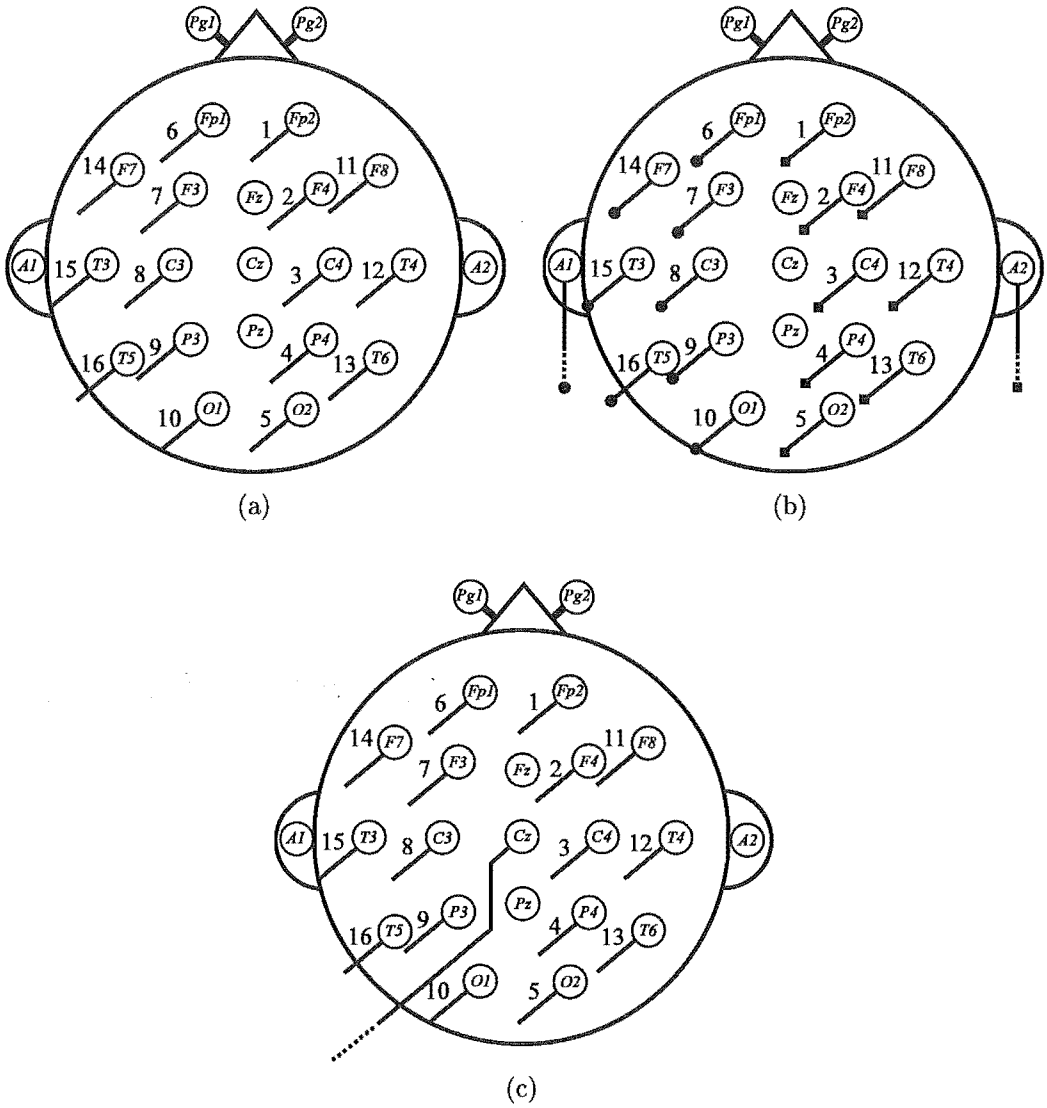


Figure A.3 The three referential montages used in this system. (a) Average, (b) ipsilateral-ears reference and (c) vertex reference.

A.2 STANDARD RECORDING PROTOCOLS

The EEG is recorded using a number of standard protocols which are changed according to the patient's age. The protocols change for (a) adults ($\geq$ 5 years), (b) children (1–5 years) and (c) babies ($<$ 12 months). Each recording protocol is described in the Tables A.1, A.2 and A.3 respectively.

Both the OVER5 and UNDER5 protocols include bipolar and referential montages, whereas BABY protocol includes only bipolar longitudinal montage. Each protocol includes one run with photic stimulation and the OVER5 protocol includes a run with hyperventilation.

OVER5 recordings tend to be around 22 minutes long, whereas UNDER 5 are

around 12 minutes long and BABY around 9 minutes long.

Adult protocol (OVER5)		
Run	Montage	Duration
1	Longitudinal	200 s
6	Ipsi-ears reference	100 s
2	Transverse	100 s
3	Long-transverse	100 s
4	Circumferential	100 s
5	Average	100 s
6	Ipsi-ears reference	100 s
7	Vertex reference	100 s
1	Longitudinal – (hyperventilation)	300 s
1	Longitudinal – (photic stimulation)	100 s

Table A.1 The protocol used for recording adult EEGs ($\geq$ 5 years).

Child protocol (UNDER5)		
Run	Montage	Duration
1	Longitudinal	200 s
2	Transverse	100 s
3	Long-transverse	100 s
4	Circumferential	100 s
5	Average	100 s
7	Vertex reference	100 s
1	Longitudinal – (photic stimulation)	60 s

Table A.2 The protocol used for recording child EEGs (1–5 years).

Baby protocol		
Run	Montage	Duration
1	Longitudinal	500 s
1	Longitudinal – (photic stimulation)	60 s

Table A.3 The protocol used for recording baby EEGs (< 12 months).

Appendix B

SOFM SIMULATION RESULTS

Decay	×	α_{min}		Uniform taper		Gaussian taper		Quadratic taper	
				Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
Lin	200	0.01	SOFM	89.0	92.9	86.9	93.5	87.8	93.7
			LVQ1	88.0	93.9	85.3	94.1	85.7	93.9
			LVQ2	87.5	94.1	86.3	94.3	87.3	93.7
		0.001	SOFM	86.5	96.1	87.8	93.3	89.0	92.0
			LVQ1	86.3	95.3	87.1	92.9	87.1	94.9
			LVQ2	84.7	96.1	86.3	95.7	86.3	90.8
	500	0.01	SOFM	88.2	93.5	87.1	94.3	87.8	93.7
			LVQ1	88.8	93.5	88.2	93.3	88.4	93.7
			LVQ2	88.2	93.9	85.1	94.7	86.9	94.3
		0.001	SOFM	87.6	93.1	87.1	93.7	84.5	95.5
			LVQ1	85.7	93.5	86.9	93.3	86.7	93.5
			LVQ2	87.5	93.7	88.2	94.1	84.3	95.5
Exp	200	0.01	SOFM	87.5	92.2	88.6	90.8	87.8	93.7
			LVQ1	89.2	92.7	90.2	91.4	88.2	93.5
			LVQ2	86.9	93.7	87.6	92.9	86.9	94.7
		0.001	SOFM	87.8	94.3	88.0	93.7	87.8	93.3
			LVQ1	87.1	95.1	88.0	93.7	87.5	93.5
			LVQ2	87.3	95.9	88.6	94.3	88.4	93.7
	500	0.01	SOFM	88.6	92.7	88.0	92.9	87.8	91.8
			LVQ1	88.0	92.7	88.0	94.5	89.8	91.0
			LVQ2	88.0	93.9	88.0	93.7	86.3	94.7
		0.001	SOFM	88.4	94.1	89.0	92.0	86.9	94.5
			LVQ1	87.6	93.9	88.0	92.0	86.9	94.3
			LVQ2	88.2	95.3	87.3	93.9	87.6	94.7

Table B.1 SENSITIVITY of [8 × 8] SOFM for Simulation 1.

Decay	×	α_{min}		Uniform taper		Gaussian taper		Quadratic taper	
				Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
Lin	200	0.01	SOFM	92.8	89.0	93.3	87.2	93.5	88.1
			LVQ1	93.7	88.3	93.8	86.0	93.6	86.3
			LVQ2	93.9	87.8	93.1	88.1	93.5	87.6
		0.001	SOFM	95.9	87.2	92.7	93.3	92.1	89.0
			LVQ1	95.0	87.0	92.7	87.3	94.7	87.6
			LVQ2	95.8	85.8	95.4	87.0	90.7	86.4
	500	0.01	SOFM	93.4	88.4	94.1	87.5	93.5	88.1
			LVQ1	93.4	88.9	93.2	88.4	93.6	88.6
			LVQ2	93.8	88.5	94.3	85.9	94.1	87.3
		0.001	SOFM	92.9	87.9	93.5	87.4	95.1	85.6
			LVQ1	93.2	86.3	93.1	87.2	93.2	87.1
			LVQ2	93.5	87.8	93.9	88.5	95.1	85.4
Exp	200	0.01	SOFM	92.1	87.6	90.9	88.5	93.5	88.1
			LVQ1	92.7	89.2	91.6	90.0	93.4	88.4
			LVQ2	93.5	87.3	92.7	87.8	94.5	87.4
		0.001	SOFM	94.1	88.2	93.5	88.3	93.1	88.1
			LVQ1	94.9	87.6	93.5	88.3	93.3	87.7
			LVQ2	95.7	87.9	94.2	88.8	93.6	88.6
	500	0.01	SOFM	92.6	88.7	92.8	88.2	91.8	87.9
			LVQ1	92.6	88.2	94.3	88.4	91.2	89.6
			LVQ2	93.7	88.3	93.5	88.3	94.4	86.9
		0.001	SOFM	94.0	88.7	92.1	89.0	94.3	87.4
			LVQ1	93.7	88.0	92.0	88.1	94.1	87.3
			LVQ2	95.1	88.6	93.7	87.6	94.5	88.0

Table B.2 SELECTIVITY of [8 × 8] SOFM for Simulation 1.

Decay	×	α_{min}		Uniform taper		
				Class 1	Class 2	Class 3
Lin	200	0.01	SOFM	56.3	88.0	60.9
			LVQ1	54.1	88.3	65.5
			LVQ2	57.9	87.8	58.2
		0.001	SOFM	59.6	88.0	57.3
			LVQ1	55.7	89.7	56.4
			LVQ2	56.8	89.5	58.2
	500	0.01	SOFM	59.6	87.4	61.8
			LVQ1	58.5	87.8	62.7
			LVQ2	55.7	87.8	57.3
		0.001	SOFM	64.5	86.1	60.9
			LVQ1	62.3	87.0	61.8
			LVQ2	56.3	88.1	56.4
Exp	200	0.01	SOFM	59.0	86.7	60.9
			LVQ1	61.2	86.7	67.3
			LVQ2	55.7	88.3	62.7
		0.001	SOFM	55.7	88.1	57.3
			LVQ1	59.0	87.6	56.4
			LVQ2	54.6	87.6	60.0
	500	0.01	SOFM	59.6	88.7	58.2
			LVQ1	57.4	88.4	62.7
			LVQ2	55.7	87.8	61.8
		0.001	SOFM	53.6	87.8	63.6
			LVQ1	58.5	86.1	65.5
			LVQ2	49.2	90.0	59.1

Table B.3 SENSITIVITY of [10 × 10] SOFM for Simulation 2 using Uniform taper.

Decay	×	α_{min}		Gaussian taper		
				Class 1	Class 2	Class 3
Lin	200	0.01	SOFM	51.9	90.9	57.3
			LVQ1	50.8	90.4	63.6
			LVQ2	62.3	88.1	60.9
		0.001	SOFM	58.5	87.0	69.1
			LVQ1	58.5	88.0	63.6
			LVQ2	54.6	89.5	63.6
	500	0.01	SOFM	56.8	85.6	64.5
			LVQ1	57.9	88.0	59.1
			LVQ2	54.6	89.4	62.7
		0.001	SOFM	50.8	89.7	61.8
			LVQ1	52.5	90.1	60.0
			LVQ2	53.6	90.5	55.5
Exp	200	0.01	SOFM	67.2	85.3	56.4
			LVQ1	64.5	85.7	60.0
			LVQ2	57.4	88.5	58.2
		0.001	SOFM	59.0	88.8	58.2
			LVQ1	60.1	85.4	61.8
			LVQ2	54.1	88.4	56.4
	500	0.01	SOFM	55.7	88.1	66.4
			LVQ1	53.6	69.1	60.0
			LVQ2	59.0	88.5	65.5
		0.001	SOFM	51.4	90.8	54.5
			LVQ1	61.2	88.1	59.1
			LVQ2	56.3	88.4	53.6

Table B.4 SENSITIVITY of [10 × 10] SOFM for Simulation 2 using Gaussian taper.

Decay	×	α_{min}		Quadratic taper		
				Class 1	Class 2	Class 3
Lin	200	0.01	SOFM	57.4	87.8	64.5
			LVQ1	57.4	88.4	61.8
			LVQ2	61.2	88.7	59.1
		0.001	SOFM	61.7	85.0	64.5
			LVQ1	62.8	86.3	59.1
			LVQ2	60.7	86.7	55.5
	500	0.01	SOFM	52.5	89.5	62.7
			LVQ1	55.7	88.5	62.7
			LVQ2	49.2	88.7	66.4
		0.001	SOFM	57.9	88.1	57.3
			LVQ1	61.2	88.1	56.4
			LVQ2	60.7	90.0	52.7
Exp	200	0.01	SOFM	53.0	87.6	61.8
			LVQ1	55.2	88.1	63.6
			LVQ2	55.7	88.0	65.5
		0.001	SOFM	55.7	89.3	66.4
			LVQ1	60.7	86.8	64.5
			LVQ2	53.6	89.3	67.3
	500	0.01	SOFM	55.7	88.7	58.2
			LVQ1	53.6	89.3	59.1
			LVQ2	52.5	88.7	60.9
		0.001	SOFM	51.9	89.1	63.6
			LVQ1	55.2	87.7	64.5
			LVQ2	56.8	88.8	60.9

Table B.5 SENSITIVITY of [10 × 10] SOFM for Simulation 2 using Quadratic taper.

Decay	×	α_{min}		Uniform taper		
				Class 1	Class 2	Class 3
Lin	200	0.01	SOFM	72.0	83.6	59.3
			LVQ1	72.8	84.0	59.5
			LVQ2	68.8	83.7	61.5
		0.001	SOFM	69.9	84.1	60.6
			LVQ1	73.4	83.5	60.8
			LVQ2	70.7	83.8	64.0
	500	0.01	SOFM	72.2	84.3	58.6
			LVQ1	72.3	84.3	60.0
			LVQ2	70.8	83.2	57.3
		0.001	SOFM	67.4	85.1	61.5
			LVQ1	69.9	85.2	59.1
			LVQ2	70.5	83.2	59.0
Exp	200	0.01	SOFM	70.6	84.0	57.3
			LVQ1	70.9	85.4	59.7
			LVQ2	71.8	84.0	60.0
		0.001	SOFM	70.3	83.2	59.4
			LVQ1	68.4	83.6	60.8
			LVQ2	67.6	83.3	60.6
	500	0.01	SOFM	75.2	83.9	59.3
			LVQ1	72.9	84.0	60.5
			LVQ2	70.8	83.8	59.1
		0.001	SOFM	69.5	83.4	61.4
			LVQ1	67.3	84.5	60.5
			LVQ2	71.4	82.4	63.7

Table B.6 SELECTIVITY of [10 × 10] SOFM for Simulation 2 using Uniform taper.

Decay	×	α_{min}		Gaussian taper		
				Class 1	Class 2	Class 3
Lin	200	0.01	SOFM	76.0	82.9	63.6
			LVQ1	76.9	83.3	62.5
			LVQ2	70.4	85.0	63.8
		0.001	SOFM	72.8	85.2	58.0
			LVQ1	70.9	84.6	61.4
			LVQ2	73.5	84.0	63.6
	500	0.01	SOFM	65.8	83.8	59.2
			LVQ1	71.1	83.8	59.6
			LVQ2	74.6	83.8	61.6
		0.001	SOFM	74.4	83.0	61.3
			LVQ1	75.6	83.3	61.1
			LVQ2	74.2	82.9	63.5
Exp	200	0.01	SOFM	66.5	85.3	57.4
			LVQ1	65.2	85.2	61.1
			LVQ2	69.1	83.9	62.7
		0.001	SOFM	70.1	84.0	65.3
			LVQ1	65.9	84.4	58.1
			LVQ2	64.7	82.8	67.4
	500	0.01	SOFM	71.3	83.2	62.4
			LVQ1	74.8	83.3	58.4
			LVQ2	72.5	84.9	63.2
		0.001	SOFM	74.0	82.3	64.5
			LVQ1	69.1	84.4	65.0
			LVQ2	66.0	82.9	65.6

Table B.7 SELECTIVITY of [10 × 10] SOFM for Simulation 2 using Gaussian taper.

Decay	×	α_{min}		Quadratic taper		
				Class 1	Class 2	Class 3
Lin	200	0.01	SOFM	70.9	84.4	61.2
			LVQ1	72.4	84.3	59.6
			LVQ2	71.3	84.7	63.1
		0.001	SOFM	68.1	84.8	56.8
			LVQ1	68.9	84.6	58.0
			LVQ2	68.1	83.7	58.1
	500	0.01	SOFM	73.8	83.2	63.3
			LVQ1	69.9	83.8	64.5
			LVQ2	73.8	83.0	59.3
		0.001	SOFM	69.3	83.5	62.4
			LVQ1	70.0	84.1	62.6
			LVQ2	69.8	83.9	69.9
Exp	200	0.01	SOFM	66.4	82.9	63.6
			LVQ1	68.2	83.7	64.8
			LVQ2	70.3	84.2	62.1
		0.001	SOFM	74.5	84.4	63.5
			LVQ1	69.4	85.0	60.2
			LVQ2	73.7	84.1	63.2
	500	0.01	SOFM	72.9	83.2	60.4
			LVQ1	72.1	83.0	61.3
			LVQ2	70.6	83.0	61.5
		0.001	SOFM	72.5	83.2	62.5
			LVQ1	73.7	83.9	57.3
			LVQ2	70.7	84.0	63.8

Table B.8 SELECTIVITY of [10 × 10] SOFM for Simulation 2 using Quadratic taper.

Decay	×	α_{min}		Uniform taper					
				Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Lin	200	0.01	SOFM	49.0	51.0	43.0	48.9	57.6	59.8
			LVQ1	63.5	71.6	64.0	62.2	75.0	73.2
			LVQ2	75.0	78.4	74.0	68.9	80.4	84.8
		0.001	SOFM	41.3	52.9	55.0	25.6	60.9	59.8
			LVQ1	62.5	72.5	70.0	53.3	60.9	79.5
			LVQ2	69.2	73.5	79.0	57.8	76.1	85.7
	500	0.01	SOFM	53.8	54.9	41.0	40.0	62.0	46.4
			LVQ1	75.0	77.5	60.0	62.2	78.3	71.4
			LVQ2	78.8	82.4	68.0	61.1	89.1	77.7
		0.001	SOFM	48.1	56.9	48.0	33.3	58.7	73.2
			LVQ1	71.2	74.5	60.0	41.1	64.1	78.6
			LVQ2	81.7	82.4	59.0	63.3	76.1	83.9
Exp	200	0.01	SOFM	60.6	52.0	48.0	41.1	69.6	60.7
			LVQ1	75.0	73.5	57.0	62.2	83.7	69.6
			LVQ2	81.7	80.4	63.0	57.8	81.5	77.7
		0.001	SOFM	53.8	58.8	53.0	31.1	60.9	66.1
			LVQ1	71.2	65.7	64.0	51.1	63.0	75.9
			LVQ2	73.1	71.6	72.0	46.7	78.3	80.4
	500	0.01	SOFM	57.7	51.0	42.0	38.9	71.7	58.0
			LVQ1	75.0	72.5	49.0	51.1	72.8	70.5
			LVQ2	82.7	73.5	61.0	73.3	80.4	75.9
		0.001	SOFM	43.3	53.9	51.0	43.3	60.9	54.5
			LVQ1	68.3	61.8	58.0	61.1	66.3	63.4
			LVQ2	69.2	75.5	62.0	57.8	75.0	79.5

Table B.9 SENSITIVITY of [10 × 10] SOFM for Simulation 3 using Uniform taper.

Decay	×	α_{min}		Gaussian taper					
				Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Lin	200	0.01	SOFM	68.3	52.9	45.0	40.0	67.4	64.3
			LVQ1	77.9	67.6	56.0	46.7	80.4	78.6
			LVQ2	82.7	77.5	61.0	62.2	73.9	83.9
		0.001	SOFM	46.2	57.8	61.0	27.8	62.0	67.0
			LVQ1	62.5	67.6	70.0	56.7	66.3	81.2
			LVQ2	69.2	89.2	65.0	51.1	77.2	83.9
	500	0.01	SOFM	61.5	63.7	42.0	36.7	72.8	47.3
			LVQ1	74.0	77.5	54.0	50.0	78.3	62.5
			LVQ2	72.1	86.3	56.0	70.0	82.6	76.8
		0.001	SOFM	46.2	56.9	47.0	50.0	69.6	62.5
			LVQ1	68.3	74.5	72.0	60.0	73.9	75.0
			LVQ2	73.1	77.5	70.0	66.7	83.7	79.5
Exp	200	0.01	SOFM	36.5	53.9	58.0	37.8	72.8	58.9
			LVQ1	53.8	59.8	74.0	61.1	85.9	75.9
			LVQ2	71.2	75.5	72.0	61.1	84.8	80.4
		0.001	SOFM	51.9	55.9	57.0	32.2	64.1	55.4
			LVQ1	54.8	63.7	68.0	53.3	68.5	70.5
			LVQ2	75.0	81.4	63.0	56.7	80.4	80.4
	500	0.01	SOFM	42.3	53.9	66.0	40.0	66.3	56.2
			LVQ1	63.5	62.7	60.0	48.9	70.7	75.0
			LVQ2	69.2	70.6	65.0	56.7	84.8	75.9
		0.001	SOFM	47.1	56.9	52.0	35.6	62.0	64.3
			LVQ1	70.2	74.5	64.0	47.8	66.3	74.1
			LVQ2	79.8	77.5	65.0	54.4	78.3	82.1

Table B.10 SENSITIVITY of [10 × 10] SOFM for Simulation 3 using Gaussian taper.

Decay	×	α_{min}		Quadratic taper					
				Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Lin	200	0.01	SOFM	54.8	63.7	48.0	28.9	76.1	53.6
			LVQ1	73.1	76.5	74.0	45.6	83.7	73.2
			LVQ2	76.9	87.3	65.0	68.9	76.1	77.7
		0.001	SOFM	38.5	60.8	53.0	34.4	62.0	66.1
			LVQ1	61.5	76.5	70.0	50.0	65.2	79.5
			LVQ2	79.8	86.3	74.0	70.0	78.3	82.1
	500	0.01	SOFM	53.8	58.8	52.0	23.3	66.3	69.6
			LVQ1	67.3	70.6	57.0	42.2	66.3	79.5
			LVQ2	81.7	74.5	66.0	68.9	75.0	87.5
		0.001	SOFM	53.8	49.0	53.0	46.7	70.7	53.6
			LVQ1	73.1	61.8	56.0	61.1	76.1	66.1
			LVQ2	80.8	71.6	62.0	67.8	88.0	74.1
Exp	200	0.01	SOFM	44.2	68.6	41.0	33.3	63.0	60.7
			LVQ1	61.5	78.4	64.0	52.2	68.5	72.3
			LVQ2	70.2	85.3	64.0	54.4	78.3	77.7
		0.001	SOFM	48.1	54.9	55.0	43.3	71.7	55.4
			LVQ1	54.8	65.7	61.0	60.0	76.1	67.9
			LVQ2	68.3	82.4	65.0	65.6	87.0	79.5
	500	0.01	SOFM	45.2	55.9	57.0	58.9	67.4	51.8
			LVQ1	71.2	65.7	54.0	71.1	76.1	67.0
			LVQ2	71.2	70.6	68.0	73.3	81.5	75.9
		0.001	SOFM	51.9	56.9	53.0	45.6	64.1	67.0
			LVQ1	68.3	70.6	60.0	52.2	72.8	75.0
			LVQ2	80.8	69.6	59.0	65.6	85.9	76.8

Table B.11 SENSITIVITY of [10 × 10] SOFM for Simulation 3 using Quadratic taper.

Decay	×	α_{min}		Uniform taper					
				Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Lin	200	0.01	SOFM	44.7	71.2	37.1	53.0	75.5	46.5
			LVQ1	65.3	92.4	52.5	61.5	98.6	59.9
			LVQ2	75.7	97.6	63.2	81.6	96.1	65.5
		0.001	SOFM	48.9	72.0	38.5	35.4	81.2	41.9
			LVQ1	70.0	91.4	52.6	61.5	100.0	55.6
			LVQ2	81.8	93.8	57.2	76.5	100.0	61.5
	500	0.01	SOFM	48.7	62.9	40.2	46.2	58.2	44.1
			LVQ1	69.0	85.9	59.4	68.3	92.3	59.7
			LVQ2	75.9	86.6	67.3	69.6	97.6	66.4
		0.001	SOFM	52.1	63.0	40.7	43.5	93.1	49.1
			LVQ1	65.5	89.4	56.1	55.2	95.2	53.0
			LVQ2	67.5	94.4	64.8	78.1	94.6	63.9
Exp	200	0.01	SOFM	52.1	76.8	41.4	50.7	79.0	48.6
			LVQ1	62.4	87.2	60.0	63.6	92.8	63.4
			LVQ2	72.0	86.3	68.5	69.3	97.4	60.8
		0.001	SOFM	53.3	83.3	41.4	45.2	73.7	47.1
			LVQ1	65.5	95.7	53.3	54.1	100.0	55.2
			LVQ2	73.1	90.1	59.5	66.7	97.3	57.3
	500	0.01	SOFM	50.0	68.4	38.2	45.5	81.5	47.8
			LVQ1	60.9	81.3	56.3	56.8	93.1	56.0
			LVQ2	69.9	85.2	64.2	71.0	100.0	66.9
		0.001	SOFM	48.9	69.6	38.9	45.3	77.8	43.6
			LVQ1	59.2	94.0	50.9	52.9	93.8	54.6
			LVQ2	64.9	98.7	54.4	68.4	100.0	58.6

Table B.12 SELECTIVITY of [10 × 10] SOFM for Simulation 3 using Uniform taper.

Decay	×	α_{min}		Gaussian taper					
				Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Lin	200	0.01	SOFM	59.2	64.3	44.1	48.6	84.9	49.0
			LVQ1	61.8	85.2	59.6	62.7	100.0	57.5
			LVQ2	69.9	86.8	66.3	74.7	100.0	62.3
		0.001	SOFM	60.0	64.8	45.9	43.9	85.1	43.6
			LVQ1	69.9	84.1	54.3	68.0	93.8	58.3
			LVQ2	72.0	92.9	64.4	69.7	100.0	57.3
	500	0.01	SOFM	56.6	67.7	43.3	41.8	71.3	43.8
			LVQ1	65.3	84.9	56.8	59.2	84.7	52.6
			LVQ2	65.8	91.7	58.3	70.0	100.0	67.2
		0.001	SOFM	52.7	61.1	39.2	52.9	84.2	52.6
			LVQ1	70.3	92.7	58.5	63.5	98.6	60.0
			LVQ2	70.4	96.3	60.3	73.2	98.7	66.4
Exp	200	0.01	SOFM	50.7	80.9	35.6	53.1	69.1	49.6
			LVQ1	67.5	91.0	48.1	68.8	91.9	65.4
			LVQ2	74.0	93.9	58.1	71.4	100.0	64.7
		0.001	SOFM	48.2	90.5	43.8	39.7	74.7	43.4
			LVQ1	59.4	98.5	48.6	57.8	88.7	54.9
			LVQ2	67.8	93.3	61.8	70.8	97.4	61.6
	500	0.01	SOFM	53.7	78.6	42.9	50.0	73.5	45.3
			LVQ1	61.7	97.0	47.6	57.1	98.5	53.2
			LVQ2	66.7	90.0	55.1	63.8	98.7	63.0
		0.001	SOFM	53.8	63.0	42.3	40.0	87.7	48.3
			LVQ1	66.4	89.4	58.2	55.1	92.4	55.0
			LVQ2	70.9	95.2	61.3	74.2	96.0	60.1

Table B.13 SELECTIVITY of [10 × 10] SOFM for Simulation 3 using Gaussian taper.

Decay	×	α_{min}		Quadratic taper					
				Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Lin	200	0.01	SOFM	61.3	65.0	42.5	41.9	73.7	43.8
			LVQ1	72.4	100.0	60.7	57.7	97.5	56.6
			LVQ2	74.8	84.0	69.9	72.1	98.6	63.5
		0.001	SOFM	51.9	67.4	38.7	47.7	83.8	46.0
			LVQ1	71.1	91.8	53.8	62.5	95.2	55.6
			LVQ2	78.3	95.7	68.5	75.9	100.0	66.2
	500	0.01	SOFM	56.0	73.2	41.9	46.7	78.2	45.6
			LVQ1	66.7	86.7	50.9	57.6	96.8	52.0
			LVQ2	73.3	97.4	62.3	82.7	98.6	63.2
		0.001	SOFM	52.3	80.6	38.7	43.8	83.3	50.0
			LVQ1	55.1	98.4	53.8	57.3	94.6	59.7
			LVQ2	67.7	89.0	62.0	68.5	98.8	67.5
Exp	200	0.01	SOFM	58.2	60.9	36.6	38.5	84.1	46.3
			LVQ1	70.3	81.6	54.7	54.7	98.4	56.2
			LVQ2	69.5	88.8	62.1	65.3	96.0	60.4
		0.001	SOFM	45.9	84.8	42.0	47.6	78.6	48.4
			LVQ1	62.6	88.2	43.9	60.0	93.3	58.9
			LVQ2	66.4	97.7	57.5	73.8	96.4	67.9
	500	0.01	SOFM	48.5	80.3	41.3	51.5	79.5	51.3
			LVQ1	57.8	95.7	50.0	62.7	100.0	61.5
			LVQ2	74.0	85.7	56.2	71.0	100.0	66.9
		0.001	SOFM	54.0	66.7	44.9	53.9	84.3	50.3
			LVQ1	65.1	85.7	53.1	68.1	82.7	58.3
			LVQ2	70.6	89.9	56.7	72.8	92.9	65.2

Table B.14 SELECTIVITY of [10 × 10] SOFM for Simulation 3 using Quadratic taper.

Appendix C

THE FUZZY RULES

The following are the 127 fuzzy rules obtained using the method described in Section 10.5. Each rule has 4 input variables (CH1, CH2, CH3 and CH4) representing the probability values assigned to each CED and one output variable (OP) representing the detection level based on the outcome of each fuzzy rule. The inputs are fuzzified using the fuzzy variables PB (positive big), PS (positive small), ZE (zero), NS (negative small) and NB (negative big), and defuzzified using the fuzzy variables DEF (definite), PRO (probable), POS (possible) and ZE (zero).

Each rule is of the form

IF CH1 is *fvar* **AND** CH2 is *fvar* **AND** CH3 is *fvar* **AND** CH4 is *fvar* **THEN** OP is *fvar*

where *fvar* represents the fuzzy input and fuzzy output variables described above.

Spatial fuzzy rules											
Rule	CH1	CH2	CH3	CH4	OP	Rule	CH1	CH2	CH3	CH4	OP
1	PB	PB	PB	PB	DEF	2	PS	PB	PB	PB	DEF
3	PS	PS	PB	PB	PRO	4	PS	PS	PS	PB	PRO
5	PS	PS	PS	PS	PRO	6	ZE	PB	PB	PB	PRO
7	ZE	PS	PB	PB	PRO	8	ZE	PS	PS	PB	PRO
9	ZE	PS	PS	PS	POS	10	ZE	ZE	PB	PB	PRO
11	ZE	ZE	PS	PB	POS	12	ZE	ZE	PS	PS	POS
13	ZE	ZE	ZE	PB	POS	14	ZE	ZE	ZE	PS	ZE
15	PB	PB	PB	NB	DEF	16	PB	PB	PB	NS	DEF
17	PS	PB	PB	NB	DEF	18	PS	PB	PB	NS	PRO
19	PS	PS	PB	NB	PRO	20	PS	PS	PB	NS	PRO
21	PS	PS	PS	NB	PRO	22	PS	PS	PS	NS	PRO
23	ZE	PB	PB	NB	PRO	24	ZE	PB	PB	NS	PRO
25	ZE	PS	PB	NB	PRO	26	ZE	PS	PB	NS	PRO
27	ZE	PS	PS	NB	PRO	28	ZE	PS	PS	NS	POS
29	ZE	ZE	PB	NB	PRO	30	ZE	ZE	PB	NS	POS
31	ZE	ZE	PS	NB	POS	32	ZE	ZE	PS	NS	POS

Spatial fuzzy rules											
Rule	CH1	CH2	CH3	CH4	OP	Rule	CH1	CH2	CH3	CH4	OP
33	PB	PB	NB	NB	DEF	34	PB	PB	NB	NS	DEF
35	PB	PB	NB	ZE	PRO	36	PB	PB	NS	NS	PRO
37	PB	PB	NS	ZE	PRO	38	PS	PB	NB	NB	DEF
39	PS	PB	NB	NS	PRO	40	PS	PB	NB	ZE	PRO
41	PS	PB	NS	NS	PRO	42	PS	PB	NS	ZE	PRO
43	PS	PS	NB	NB	PRO	44	PS	PS	NB	NS	PRO
45	PS	PS	NB	ZE	PRO	46	PS	PS	NS	NS	PRO
47	PS	PS	NS	ZE	POS	48	ZE	PB	NB	NB	PRO
49	ZE	PB	NB	NS	PRO	50	ZE	PB	NB	ZE	PRO
51	ZE	PB	NS	NS	PRO	52	ZE	PB	NS	ZE	POS
53	ZE	PS	NB	NB	PRO	54	ZE	PS	NB	NS	PRO
55	ZE	PS	NB	ZE	POS	56	ZE	PS	NS	NS	POS
57	ZE	PS	NS	ZE	POS	58	PB	NB	NB	NB	DEF
59	PB	NB	NB	NS	DEF	60	PB	NB	NB	ZE	PRO
61	PB	NB	NS	NS	PRO	62	PB	NB	NS	ZE	PRO
63	PB	NB	ZE	ZE	PRO	64	PB	NS	NS	NS	PRO
65	PB	NS	NS	ZE	PRO	66	PB	NS	ZE	ZE	POS
67	PS	NB	NB	NB	DEF	68	PS	NB	NB	NS	PRO
69	PS	NB	NB	ZE	PRO	70	PS	NB	NS	NS	PRO
71	PS	NB	NS	ZE	PRO	72	PS	NB	ZE	ZE	POS
73	PS	NS	NS	NS	PRO	74	PS	NS	NS	ZE	POS
75	PS	NS	ZE	ZE	POS	76	NB	NB	NB	NB	DEF
77	NB	NB	NB	NS	DEF	78	NB	NB	NB	ZE	PRO
79	NB	NB	NS	NS	PRO	80	NB	NB	NS	ZE	PRO
81	NB	NB	ZE	ZE	PRO	82	NB	NS	NS	NS	PRO
83	NB	NS	NS	ZE	PRO	84	NB	NS	ZE	ZE	POS
85	NB	ZE	ZE	ZE	POS	86	NS	NS	NS	NS	PRO
87	NS	NS	NS	ZE	POS	88	NS	NS	ZE	ZE	POS
89	NS	ZE	ZE	ZE	ZE	90	ZE	NB	NB	NB	PRO
91	ZE	NB	NB	NS	PRO	92	ZE	NB	NB	ZE	PRO
93	ZE	NB	NS	NS	PRO	94	ZE	NB	NS	ZE	POS
95	ZE	NB	ZE	ZE	POS	96	ZE	NS	NS	NS	POS
97	ZE	NS	NS	ZE	POS	98	ZE	NS	ZE	ZE	ZE
99	PB	ZE	NB	NB	PRO	100	PB	ZE	NB	NS	PRO
101	PB	ZE	NB	ZE	PRO	102	PB	ZE	NS	NS	PRO
103	PB	ZE	NS	ZE	POS	104	PS	ZE	NB	NB	PRO
105	PS	ZE	NB	NS	PRO	106	PS	ZE	NB	ZE	POS
107	PS	ZE	NS	NS	POS	108	PS	ZE	NS	ZE	POS
109	PB	PB	ZE	NB	PRO	110	PB	PB	ZE	NS	PRO
111	PS	PB	ZE	NB	PRO	112	PS	PB	ZE	NS	PRO
113	PS	PS	ZE	NB	PRO	114	PS	PS	ZE	NS	POS
115	ZE	PB	ZE	NB	PRO	116	ZE	PB	ZE	NS	POS
117	ZE	PS	ZE	NB	POS	118	ZE	PS	ZE	NS	POS
119	PB	PB	PB	ZE	PRO	120	PS	PB	PB	ZE	PRO
121	PS	PS	PB	ZE	PRO	122	PS	PS	PS	ZE	POS
123	ZE	PB	PB	ZE	PRO	124	ZE	PS	PB	ZE	POS
125	ZE	PS	PS	ZE	POS	126	ZE	ZE	PB	ZE	POS
127	ZE	ZE	PS	ZE	ZE						

REFERENCES

ADRIAN, E. and MATTHEWS, B. (1934), 'The Berger rhythm: potential changes from the occipital lobes of man', *Brain*, Vol. 57, p. 355.

ADRIAN, E. and YAMIGAWA, K. (1935), 'The origin of the Berger rhythm', *Brain*, Vol. 58, p. 323.

ALEKSANDER, I. and MORTON, H. (1990), *An Introduction to Neural Computing*, Chapman & Hall, London.

AMBROS-INGERSO, J., GRANGER, R. and LYNCH, G. (1990), 'Simulation of paleocortex performs hierarchical clustering', *Science*, Vol. 247, pp. 1344–1348.

AMIT, D. (1989), *Modeling brain function*, Cambridge University Press.

ANDERSEN, P. and ANDERSSON, S. (1968), *Physiological basis of the alpha rhythm*, Appleton Century Crofts, New York.

ANDERSON, J. (1972), 'A simple neural network generating an interactive memory', *Mathematical Biosciences*, Vol. 14, pp. 197–220.

BARNARD, E. (1992), 'Optimization for training neural nets', *IEEE Transactions on Neural Networks*, Vol. 3, pp. 232–240.

BATTITI, R. (1992), 'First and second order methods for learning: Between steepest descent and Newton's method', *Neural Computation*, Vol. 4, pp. 141–166.

BECK, A. (1890), 'Die bestimmung der localization der gehirn- und ruckenmarkfunctionen vermittelst der electrischen erscheinungen', *Zbl. Physiol.*, Vol. 4, p. 473.

BECKER, K., RAU, G., KAESMACHER, H., PETERMEYER, M., KALFF, G. and ZIMMERMANN, H. (1994), 'Fuzzy logic approaches to intelligent alarms', *IEEE Engineering in Medicine and Biology*, Vol. 13, pp. 710–716.

BERGER, H. (1929), 'Uber das ellektrenkephalogramm des menschen', *Arch. Psychiat. NervKrankh.*, Vol. 87, p. 527.

BERNARDO, J. and SMITH, F. (1994), *Bayesian Theory*, John Wiley and Sons, New York NY.

BIRKEMEIER, W., FONTAINE, A., CELESIA, G. and MA, K. (1978), 'Pattern recognition techniques for the detection of epileptic transients in EEG', *IEEE Trans. Biomed. Eng.*, Vol. BME-25, pp. 213–217.

BRODAL, A. (1981), *Neurological Anatomy in Relation to Clinical Medicine, 3rd ed.*, Oxford University Press, New York.

CARLSON, N. (1986), *Physiology of behaviour*, Allyn and Bacon.

CARPENTER, G. and GROSSBERG, S. (1991), *Pattern Recognition by Self-Organizing Neural Networks*, MIT Press, Cambridge MA.

CATON, R. (1875), 'The electric currents of the brain', *British Medical Journal*, Vol. 2, p. 278.

CHARALAMBOUS, C. (1992), 'Conjugate gradient algorithm for efficient training of artificial neural networks', *IEEE Proceedings*, Vol. 139, pp. 301–310.

CHATRIAN, G., BERGAMINI, L., DONDEY, M., KLASS, D., PETERSON, I. and LENNOX-BUCHTALL, M. (1974), 'A glossary of terms most commonly used by clinical electroencephalographers', *Electroencephalography and Clinical Neurophysiology*, Vol. 37, pp. 538–548.

CLARKE, E. and DEWHURST, K. (1972), *An illustrated history of brain function*, Sanford Publications, Oxford.

COEN, C. (1985), *Functions of the brain*, Clarendon Press, Oxford.

DAVEY, B., FRIGHT, W., CARROLL, G. and JONES, R. (1989), 'Expert system approach to detection of epileptiform activity in the EEG', *Medical and Biological Engineering and Computing*, Vol. 27, pp. 365–370.

DESIENO, D. (1988), 'Adding a conscience to competitive learning', In *Proc. IEEE Intl. Conf. on Neural Networks*, pp. I–117–I–124.

DINGLE, A. (1992), *Engineering in Brain Research: Processing electroenecephalograms and Chaos in neural networks*, PhD thesis, University of Canterbury, Christchurch, New Zealand.

DINGLE, A., JONES, R., CARROLL, G. and FRIGHT, W. (1993), 'A multistage system to detect epileptiform activity in the EEG', *IEEE Trans. Biomed. Eng.*, Vol. BME-40, pp. 1260–1268.

DURBIN, R., MIALL, C. and MITCHISON, G. (1989), *The Computing Neuron*, MA: Addison-Wesley, Reading.

DYRO, F. (1989), *The EEG Handbook*, MA: Little, Brown and Company, Boston.

EBERHART, R., DOBBINS, R. and WEBBER, W. (1989), 'Neural network design considerations for EEG spike detection', In *Proc. 15th Northeast Bioengineering Conference*, pp. 97–98.

ELUL, R. (1972), 'The genesis of the EEG', *International Review of Neurobiology*, Vol. 15, p. 227.

EMPSON, J. (1986), *Human Brainwaves: The Pyschological Significance of the EEG*, Macmillan Press, London.

FERRARA, E. and WIDROW, B. (1981), 'Multichannel adaptive filtering for signal enhancement', *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-29, pp. 766–770.

FISCHER, G., MARS, N. and LOPES DA SILVA, F. (1980), *Pattern recognition of epileptiform transients in the electroencephalogram*, Progress report 7., Institute of Medical Physics, Da Costakade, Utrecht.

GABOR, A. and SEYAL, M. (1992), 'Automated interictal EEG spike detection using artificial neural networks', *Electroenceph. Clin. Neurophysiol.*, Vol. 83, pp. 271–280.

GERSHO, R. (1982), 'On the structure of vector quantizers', *IEEE Transactions on Information Theory*, Vol. IT-28, pp. 157–166.

GLOVER, JR., J., RAGHAVAN, N., KTONAS, P. and FROST, JR., J. (1989), 'Context-based automated detection of epileptogenic sharp transients in the EEG: Elimination of false positives', *IEEE Trans. Biomed. Eng.*, Vol. BME-36, pp. 519–527.

GOTMAN, J. (1980), 'Quantitative measurements of epileptic spike morphology in the human EEG', *Electroenceph. Clin. Neurophysiol.*, Vol. 48, pp. 551–557.

GOTMAN, J. and GLOOR, P. (1976), 'Automatic recognition and quantification of interictal epileptic activity in the human scalp EEG', *Electroenceph. Clin. Neurophysiol.*, Vol. 41, pp. 513–529.

GOTMAN, J. and WANG, L. (1991), 'State dependant spike detection: concepts and preliminary results', *Electroenceph. Clin. Neurophysiol.*, Vol. 79, pp. 1–19.

GOTMAN, J. and WANG, L. (1992), 'State dependant spike detection: validation', *Electroenceph. Clin. Neurophysiol.*, Vol. 83, pp. 12–18.

GOTMAN, J., GLOOR, P. and SCHAUL, N. (1978), 'Comparison of traditional reading of the EEG and automatic recognition of interictal epileptic activity', *Electroenceph. Clin. Neurophysiol.*, Vol. 44, pp. 48–60.

GRAY, R. (1984), 'Vector quantization', *IEEE ASSP Magazine*, Vol. 1, pp. 4–29.

GROSSBERG, S. (1976), 'Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors', *Biological Cybernetics*, Vol. 23, pp. 121–134.

HAGAN, M. and MENHAJ, M. (1994), 'Training feedforward networks with the Marquardt algorithm', *IEEE Transactions on Neural Networks*, Vol. 5, pp. 989–993.

HAGAN, M., DEMUTH, H. and BEALE, M. (1996), *Neural Network Design*, PWS Publishing Company, MA: Boston.

HAYKIN, S. (1991), *Adaptive Filter Theory*, NJ: Prentice-Hall, 2nd ed. Englewood Cliffs.

HAYKIN, S. (1994), *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, New York.

HEBB, D. (1949), *The organisation of behavior*, Wiley, New York.

HERTZ, J., KROGH, A. and PALMER, R. (1991), *Introduction to the Theory of Neural Computation*, MA: Addison-Wesley, Reading.

HILGARD, E., ATKINSON, R. and ATKINSON, R. (1979), *Introduction to pyschology*, Harcourt Brace Jovanovich Inc.

HOPFIELD, J. (1982), 'Neural networks and physical systems with emergent collective computational properties', *Proceedings of the National Academy of Sciences*, Vol. 79, pp. 2554–2558.

HOPFIELD, J. and TANK (1985), 'Neural computations of decisions in optimization problems', *Biological Cybernetics*, Vol. 52, pp. 141–154.

HOSTETLER, W., DOLLER, H. and HOMAN, W. (1992), 'Assessment of a computer program to detect epileptiform spikes', *Electroenceph. Clin. Neurophysiol.*, Vol. 83, pp. 1–11.

HUDSON, D. and COHEN, M. (1994), 'Fuzzy logic in medical expert systems', *IEEE Engineering in Medicine and Biology*, Vol. 13, pp. 693–698.

HUSH, D. and HORNE, B. (1993), 'Progress in supervised neural networks: what's new since Lippman ?', *IEEE Acoustics, Speech and Signal Processing Magazine*, Vol. , January, pp. 8–39.

JACOBS, R. (1988), 'Increased rates of convergence through learning rate adaption', *Neural Networks*, Vol. 1, pp. 295–308.

JASPER, H. (1958), 'Report of the committee of methods of clinical examination in electroencephalography', *EEG Journal*, Vol. 10, p. 370.

JERVIS, B., SAATCHI, M., LACEY, A., ROBERTS, T., ALLEN, E., HUDSON, N., OKE, S. and GRIMSLEY, M. (1994), 'Artificial neural network and spectrum analysis methods for detecting brain diseases from the CNV response in the electroencephalogram', *IEE Proceedings–Science, Measurement and Technology*, Vol. 141, November, pp. 432–440.

JONES, R., DINGLE, A., CARROLL, G., SATHERLEY, B., MUIR, S., DONALDSON, I., PARKIN, P. and BONES, P. (1994), 'A pc-based system for automated analysis of the EEG', In *Proc. 1st Medical Engineering Week of the World Conference, Taipei, Taiwan*, pp. 153–157.

JONES, R., DINGLE, A., CARROLL, G., GREEN, R., BLACK, M., DONALDSON, I., PARKIN, P., BONES, P. and BURGESS, K. (1996), 'A system for detecting epileptiform discharges in the EEG: real-time operation and clinical trial', In *Proc. 18th Ann. Int. Conf. of IEEE Engineering in Medicine and Biology Society, 31 Oct–3 Nov, Amsterdam.*

KAAS, J., MERZENICH, M. and KILLACKEY, H. (1983), 'The reorganization of somatosensory cortex following peripheral nerve damage in adult and developing mammals', *Ann. Rev. Neurosci.*, Vol. 6, pp. 325–356.

KALAYCI, T. and ÖZDAMAR, O. (1995), 'Wavelet preprocessing for automated neural network detection of EEG spikes', *IEEE Eng. Med. Biol.*, Vol. 14, pp. 160–166.

KETY, S. (1979), 'Disorders of the human brain', *Scientific American*, Vol. 241, pp. 172–179.

KLIR, G. and FOLGER, T. (1988), *Fuzzy sets, uncertainty, and information*, Prentice-Hall, Englewood Cliffs NJ.

KNUDSEN, E., DU LAC, S. and ESTERLY, S. (1987), 'Computational maps in the brain', *Ann. Rev. Neurosci.*, Vol. 10, pp. 41–65.

KOHONEN, T. (1972), 'Correlation matrix memories', *IEEE Transactions on Computers*, Vol. 21, pp. 353–359.

KOHONEN, T. (1982), 'Self-organized formation of topologically correct feature maps', *Biological Cybernetics*, Vol. 43, pp. 59–69.

KOHONEN, T. (1988), 'Learning vector quantization', *Neural Networks*, Vol. 1, p. 303.

KOHONEN, T. (1989a), *Self-Organization and Associative Memory, 3rd ed.*, Springer-Verlag, Berlin, Heidelberg, Germany.

KOHONEN, T. (1989b), 'Self-organizing semantic maps', *Biological Cybernetics*, Vol. 61, pp. 241–254.

KOHONEN, T. (1990), 'The self-organizing map', *Proc. IEEE*, Vol. 78, September, pp. 1464–1480.

KOHONEN, T. (1995), *Self-Organizing Maps*, Springer-Verlag, Berlin, Heidelberg, Germany.

KOHONEN, T., BARNA, G. and CHRISLEY, R. (1988), 'Statistical pattern recognition with neural networks: Benchmarking studies', In *Proc. IEEE Intl. Conf. on Neural Networks*, pp. I–61–I–68.

KTONAS, P. and SMITH, J. (1974), 'Quantification of abnormal EEG spike characteristics', *Comput. Biol. Med.*, Vol. 4, pp. 157–163.

KTONAS, P., LUOH, W., KEJARIWAL, M., REILLY, E. and SEWARD, M. (1981), 'Computer-aided quantification of EEG spike and sharp wave characteristics', *Electroenceph. Clin. Neurophsyiol.*, Vol. 51, pp. 237–243.

LAVIGNA, A. (1989), *Nonparametric classification using learning vector quantisation*, PhD thesis, University of Maryland, U.S.A.

LIPPMANN, R. (1987), 'An introduction to computing with neural nets', *IEEE Acoustics, Speech and Signal Processing Magazine*, Vol. 4(2), April, pp. 4–22.

LOPES DA SILVA, F., DJIK, A. and SMITS, H. (1974), 'Detection of nonstationarities in EEGs using the autoregressive model — an application to EEGs of epileptics', In (Ed.), *CEAN : Computerized EEG Analysis*, Gustav Fischer Verlag, pp. 180–199.

LOPES DA SILVA, F., VAN HULTEN, K., LOMMEN, J., VAN LEEUWEN, W., VAN VEELEN, C. and VLIEGENTHART, W. (1977), 'Automatic detection and localization of epileptic foci', *Electroenceph. Clin. Neurophysiol.*, Vol. 43, pp. 1–13.

LURIA, A. (1973), *The working brain*, Penguin Press, Harmondsworth, Middlesex.

MA, K., CELESIA, G. and BIRKEMEIER, W. (1976), 'Non-linear boundaries for differentiation between epileptic transients and background activities in EEG', *IEEE Trans. Biomed. Eng.*, Vol. , May, pp. 288–290.

MAKHOUL, J. (1975), 'Linear prediction: a tutorial review', *Proc. IEEE*, Vol. 63, pp. 561–580.

MASON, D., LINKENS, D., ABBOD, M., EDWARDS, N. and REILLY, C. (1994), 'Automated delivery of muscle relaxants using fuzzy logic control', *IEEE Engineering in Medicine and Biology*, Vol. 13, pp. 678–686.

MCCULLOCH, W. and PITTS, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115–133.

MEIER, R., NIEUWLAND, J., ZBINDEN, A. and HACISALIHZADE, S. (1992), 'Fuzzy logic control of blood pressure during anesthesia', *IEEE Control Systems*, Vol. , pp. 12–17.

MINSKY, M. (1961), 'Steps towards artificial intelligence', *Proceedings of the Institute of Radio Engineers*, Vol. 49, pp. 8–30.

MINSKY, M. and PAPERT, S. (1969), *Perceptrons*, MA:MIT Press, Cambridge.

NARENDRA, K. and PARTHASARATHY, K. (1990), 'Identification and control of dynamical systems using neural networks', *IEEE Trans. Neural Networks*, Vol. 1, March, pp. 4–27.

NGUYEN, D. and WIDROW, H. (1990), 'Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights', In *Proceedings of the IJCNN, Vol. 3*, pp. 21–26.

NILSSON, N. (1965), *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, McGraw-Hill, New York.

OLDS, J. and MILNER, P. (1954), 'Positive reinforcement produced by electrical stimulation of the septal area and other regions of the rat brain', *J. comp. Physiol. Psychol.*, Vol. 47, pp. 419–427.

ÖZDAMAR, O., YAYLALI, I., JAYAKAR, P. and LOPEZ, C. (1991), 'Multilevel neural network system for EEG spike detection', In BANKMAN, I. and TSITLIK, J. (Eds.), *Computer-based medical systems. Proc. 4th IEEE symp.*, IEEE Computer society press, Washington DC, pp. 272–279.

ÖZDAMAR, O., LOPEZ, C. and YAYLALI, I. (1992), 'Detection of EEG patterns with adaptive unsupervised neural networks', In *Proc. 1992 Intl. Biomedical Engineering days*, IEEE New York, NY, USA, pp. 192–197.

PARSA, V. and PARKER, P. (1994), 'Multireference adaptive noise cancellation applied to somatosensory evoked potentials', *IEEE Trans. Biomed. Eng.*, Vol. BME-41, August, pp. 792–800.

PFURTSCHELLER, G. and FISCHER, G. (1978), 'A new approach to spike detection using a combination of inverse and matched filter techniques', *Electroenceph. Clin. Neurophysiol.*, Vol. 44, pp. 243–247.

PRADHAN, N., SADASIVAN, P. and ARUNODAYA, G. (1996), 'Detection of seizure activity in EEG by an artificial neural network: A preliminary study', *Computers and Biomedical research*, Vol. 29, pp. 303–313.

PURPURA, D. (1959), 'Nature of electrocortical potentials and synaptic organizations in cerebral and cerebellar cortex', In PFEIFFER, C. and SMYTHIES, J. (Eds.), *International Review of Neurobiology*, Academic Press, New York, pp. 47–163.

RIGLER, A., IRVINE, J. and VOGL, T. (1990), 'Rescaling of variables in back propagation learning', *Neural Networks*, Vol. 3, pp. 561–573.

ROBERTS, S. and TARASSENKO, L. (1992), 'Analysis of the sleep EEG using a mutilayer network with spatial organisation', *IEE Proceedings*, Vol. 139, December, pp. 420–425.

ROSENBLATT, F. (1958), 'The perceptron: A probabilistic model for information storage and organisation in the brain', *Psychological review*, Vol. 65, pp. 386–408.

RUMELHART, D. and MCCLELLAND, J. (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition Vol. 1*, MA:MIT Press, Cambridge.

SADASIVAN, P. and NARAYANA DUTT, D. (1996), 'ANC schemes for the enhancement of EEG signals in the presence of EOG artifacts', *Computers and Biomedical Research*, Vol. 29, pp. 27–40.

SALTZBERG, B., LUSTICK, L. and HEATH, R. (1971), 'Detection of focal depth spiking in the scalp EEG of monkeys', *Electroenceph. Clin. Neurophysiol.*, Vol. 31, pp. 327–333.

SCALES, L. (1985), *Introduction to Non-Linear Optimization*, Springer-Verlag, New York.

SCHIFF, S., ALDROUBI, A., UNSER, M. and SATO, S. (1994), 'Fast wavelet transformation of EEG', *Electroenceph. Clin. Neurophysiol.*, Vol. , April, p. .

SCHMITT, S. (1969), *Measuring Uncertainty: An Elementary Introduction to Bayesian Statistics*, Adison-Wesley, Reading MA.

SHANNO, D. (1990), 'Recent advances in numerical techniques for large-scale optimization', In MILLER, S. and WERBOS (Eds.), *Neural Networks for Control*, Cambridge, MA: MIT Press, p. .

SHEPHERD, G. (1988), *Neurobiology, 2nd ed.*, Oxford University Press, New York.

SHEPHERD, G. and KOCH, C. (1990), 'Introduction to synaptic circuits', In SHEPHERD, G. (Ed.), *The Synaptic Organisation of the Brain*, Oxford University press, New York, pp. 3–31.

SINGH, M., PATEL, P., KHOSLA, D. and KIM, T. (1996), 'Segmentation of functional MRI by k-means clustering', *IEEE Trans. on Nuclear Science*, Vol. 43, June, pp. 2030–2036.

SPEHLMANN, R. (1981), *EEG Primer*, NY: Elsevier North-Holland, New York.

THOMPSON, R. (1967), *Foundations of Physiological Psychology*, Harper and Row.

THORNDIKE, E. (1911), *Animal Intelligence*, CT: Hafner, Darien.

TOLLENAERE, T. (1990), 'SuperSAB: Fast adaptive back propagation with good scaling properties', *Neural Networks*, Vol. 3, pp. 561–573.

VOGL, T., MANGIS, J., ZIGLER, A., ZINK, W. and ALKON, D. (1988), 'Accelerating the convergence of the backpropagation method', *Biological Cybernetics*, Vol. 59, September, pp. 256–264.

VON DER MALSBURG, C. (1973), 'Self-organisation of orientation sensitive cells in the striate cortex', *Kybernetik*, Vol. 14, pp. 85–100.

WALSH, K. (1978), *Neuropyschology : A clinical approach*, Churchill Livingstone.

WALTER, D., MULLER, H. and JELL, R. (1973), 'Semiautomatic quantification of sharpness of EEG phenomena', *IEEE Trans. Biomed. Eng.*, Vol. BME-20, pp. 53–55.

WEBBER, W., LITT, B., WILSON, K. and LESSER, R. (1994), 'Practical detection of epileptiform discharges (EDs) in the EEG using an artificial neural network: a comparison of raw and parameterized EEG data.', *Electroenceph. Clin. Neurophysiol.*, Vol. 91, pp. 194–204.

WIDROW, H. and HOFF, M. (1960), 'Adaptive switching circuits', In *1960 IRE WESCON Convention Record*, New York: IRE Part4, pp. 96–104.

WIDROW, B. and STEARNS, S. (1985), *Adaptive Signal Processing*, NJ: Prentice-Hall, Englewood Cliffs.

WIDROW, B., GLOVER, J., MCCOOL, J., KAUNITZ, J., WILLIAMS, C., HEARN, R., ZEIDLER, J., DONG, E. and GOODLIN, R. (1975), 'Adaptive noise cancelling: principles and applications', *Proc. IEEE*, Vol. 63, pp. 1692–1716.

WILLSHAW, D. and VON DER MALSBURG, C. (1976), 'How patterned neural connections can be set up by self-organisation', *Proc. R. Soc. London*, Vol. B 287, pp. 431–445.

WILSON, K., WEBBER, W., LESSER, R., FISHER, R., EBERHART, R. and DOBBINS, R. (1991), 'Detection of epileptiform spikes in the EEG using a patient-independant neural network', In BANKMAN, I. and TSITLIK, J. (Eds.), *Computer-based medical systems. Proc. 4th IEEE symp.*, IEEE Computer society press, Washington DC, pp. 264–271.

ZADEH, L.A. (1965), 'Fuzzy sets', *Inform. Contr.*, Vol. 8, pp. 338–353.

ZIMMERMANN, H. (1986), *Fuzzy sets, decision making and expert systems*, Kluwer Academic Publishers, Boston.

ZURADA, J. (1992), *Introduction to Artificial Neural Systems*, West Publishing Company, MN: St.Paul.