Modelling Chaotic Systems with Neural Networks: Application to Seismic Event Predicting in Gold Mines

Jacobus van Zyl

Thesis presented in partial fulfilment
of the requirements for the degree of
Master of Science
at the University of Stellenbosch

Supervisor: Prof. Christian W. Omlin
Co-supervisor: Prof. Andries P. J. van der Walt

December 2001

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: . . . . . .

Date: . . . . . .

Abstract

This thesis explores the use of neural networks for predicting difficult, real-world time series. We first establish and demonstrate methods for characterising, modelling and predicting well-known systems. The real-world system we explore is seismic event data obtained from a South African gold mine. We show that this data is chaotic. After preprocessing the raw data, we show that neural networks are able to predict seismic activity reasonably well.

Samevatting

Hierdie tesis ondersoek die gebruik van neurale netwerke om komplekse, werklik bestaande tydreekse te voorspel. Ter aanvang noem en demonstreer ons metodes vir die karakterisering, modelering en voorspelling van bekende stelsels. Ons gaan dan voort en ondersoek seismiese gebeurlikheidsdata afkomstig van ’n Suid-Afrikaanse goudmyn. Ons wys dat die data chaoties van aard is. Nadat ons die rou data verwerk, wys ons dat neurale netwerke die tydreekse redelik goed kan voorspel.

Acknowledgements

I would like to thank professor Omlin for all the work and helpful guidance during the course of this thesis. I would also like to thank Integrated Seismic Systems International for supporting this project by providing us with data and financial assistance. I am also thankful to Reg Dodds for advice, and help with the spelling and grammar of the thesis. Finally, I would like to thank my parents for their love and faith during all my years of study.

Publications

J. van Zyl and C.W. Omlin, ”Prediction of Seismic Events in Mines Using Neural Networks,” *International Joint Conference on Neural Networks 2001*, vol. 2, pp. 1410-1414, 2001

J. van Zyl and C.W. Omlin, ”Knowledge-Based Neural Networks for Modelling Time Series,” *Lecture Notes in Computer Science*, vol. 2085, pp. 579-586, 2001

Contents

Chapter 1

Introduction

Chaotic systems have been studied for more than 35 years. In 1963, Edward Lorenz discovered the first chaotic system, almost by accident, while searching for equations explaining the behaviour of weather patterns [62]. Since then, many methods for the analysis and synthesis of chaotic systems have been proposed.

Early work on neural networks dates back to McCulloch and Pits in 1943 [41]. Learning algorithms for neural networks that can be used for time series modelling and prediction were only developed in the 1980's [56]. Furthermore, it has been shown that neural networks are able to model and predict time series generated by chaotic systems [69].

Mankind has long been aware of seismicity and its often devastating effects. As recently as the 1970's, seismic monitoring was still only used to assist in locating the origin of disasters in order to coordinate rescue operations. In the 1990's, advances made in seismic monitoring technology have improved the accuracy of seismic measurements, making prediction feasible.

1.1 Motivation

The study of chaotic systems is motivated by the fact that many interesting natural and man-made phenomena, such as earthquakes [59], laser systems [25], and epileptic seizures [26], are of a chaotic nature. These phenomena have previously been thought to be stochastic, and therefore to be unpredictable. However, research has revealed that it is indeed possible to predict time series generated by chaotic systems. Furthermore, one can measure the degree of predictability. However, since colored noise is also predictable to some degree, it is necessary to establish the chaotic nature of the time series.

In South Africa and all over the world, many people get hurt, and even lose lives in mining accidents. Rockbursts and other mining related accidents have killed one person, and injured two others per day in South African mines during the period 2000/2001. Thus, the ability to predict these events is of great importance. It has been shown that seismic activity precipitates large seismic events, and also that this seismicity is of chaotic nature [42]. The claim of chaotic behaviour is based on the existence of attractors of fractal dimensions. Unfortunately, the seismic data contains a high degree of noise, at times up to 100%. Thus, the signal contains a large stochastic component. This study investigates the modelling and predictability of these time series. Mines with the capability to predict rockbursts will have a definite economic and social advantage since they can save lives and costs originating from rescue operations, loss of equipment, and medical care.

1.2 Problem Statement

We attempt to extract models from seismic time series that are good enough to be used for prediction. For this purpose, we use characterisation and analysis methods from the fields of dynamical systems and chaotic attractors. We use neural networks to model and predict the time series.

1.3 Objectives

Seismic events in mines can be characterised by their radiated energy and observed moments (please see Section 4.3 for a detailed discussion of source parameters that characterise seismic events). Recordings of past events form two time series: energy and moment. This thesis addresses the following technical questions:

1. Characterisation of the energy and moment time series in terms of their dynamic nature. Are they stochastic, periodic, or chaotic?
2. Determining the embedding dimension of the two coupled systems. How do past recordings of energy and moment time series determine the future behaviour of the systems?
3. Modelling the energy and moment time series with neural networks. Which neural networks are appropriate?
4. Applying regularisation methods to trained neural networks. Can their accuracy be improved by regularisation methods?
5. Define measures which quantify the performance of trained neural networks.
6. Prediction of seismic events. How well can the neural network architectures and learning algorithms predict future events from past histories of energy and moments, and what are their limitations?

1.4 Methodology

We achieve the first objective by studying chaotic attractors and their behaviour. We characterise the attractor by use of Fourier analysis, correlation dimension, Poincaré maps, and Lyapunov exponents.

We use the methods of false nearest neighbours and false strands to determine embedding dimension of chaotic attractors. We also use autocorrelation and mutual information to determine the amount of correlation of a time series with itself. This is used to obtain time series that are more suitable for modelling purposes.

We will use neural networks to model attractors. Context sensitive neural networks can be used for time series prediction. It has been shown in the literature that these networks are able to model chaotic time series obtained from the observation of system states. We investigate time delay, finite impulse response, and recurrent neural networks.

We test various techniques to improve the results of the neural networks. These techniques include pruning techniques such as the optimal brain surgeon, and improved learning techniques such as adaptive learning rates and noisy weight updates. We implement these techniques ourselves, and test them on well-known chaotic systems.

We evaluate the performance of networks for event prediction using various statistical techniques. We use ROC curves, histograms, and the mean squared error.

In order to obtain a time series suitable for modelling by a neural network, we preprocess raw data obtained from the monitoring system. We then apply all of the above-mentioned techniques to the problem of seismic event prediction. We discuss the results of these experiments in Section 5.5.

1.5 Accomplishments

We implemented most of the analysis techniques, and the remainder we obtained from their respective authors. We tested our implementations on various well-known chaotic systems. Our results were in accordance to those in the literature. Using the methods of false strands and false nearest neighbours, we were able to determine the embedding dimensions of all the chaotic systems.

We implemented various neural network architectures. We used them to model and predict the above-mentioned chaotic systems, using the information we extracted from them.

We implemented the various regularisation techniques, and applied them to the trained models. The results were as expected from the literature. The optimal brain surgeon removed excess weights, while retaining prediction accuracy. Methods such as adaptive learning rates and training with noisy weight updates all seemed to have the desired effect.

We applied the analysis methods to the seismic time series. The methods all indicate that the seismic time series are chaotic. However, we found it much harder to extract parameters such as the embedding dimension from the seismic time series. The improved learning methods again had a favourable effect on model extraction from chaotic time series.

Given their high dimensionality, and the amount of noise present in the data, we were able to track smaller events from both energy and moment seismic time series reasonably well. In general we found the energy time series easier to model than the moment time series. We were also able to predict larger events. We found that the networks tend to have a high probability of false alarms when the events to be predicted become too big. Nonetheless, we believe our results to be significant, especially in light of the fact that current state-of-the-art geophysical models predict only about 30% of big events with an undisclosed false alarm probability [38]. Thus, we believe this explorative study indicates that neural networks are indeed worthy of further investigation.

1.6 Thesis Outline

In Chapter 2, we describe examples of chaotic systems found in science and engineering. We give definitions of attractors and chaotic attractors. We then describe the various analysis techniques used for attractor characterisation. This is followed by a description techniques for extracting parameters useful for attractor modelling . Application of these techniques to various well-known systems conclude the chapter.

In Chapter 3, we introduce neural networks for time series prediction. We discuss the use of neural networks for pattern classification and time series prediction. We present three different kinds of neural networks used for time series prediction: time delay neural networks, finite impulse

response neural networks, and recurrent neural networks. Next, we discuss various methods for choosing an appropriate network architecture. Network pruning techniques go hand-in-hand with these methods. We also discuss methods for improving training and generalisation performance. We include a section on using knowledge-based neural networks to improve training performance. We briefly discuss Diks' Test which provides a method for determining whether two time series are generated by the same attractor. We conclude the chapter by applying some of these techniques to the above-mentioned example chaotic attractors.

In Chapter 4, we discuss modern seismic monitoring systems used in mines. The description of seismic events is followed by an introduction to rock dynamics. We focus our discussion on two key seismic parameters: moment and radiated energy. Next, we discuss seismic transducers, the instruments used to measure ground motion. We define the parameters used during seismic event prediction. We conclude the chapter with a discussion of current event prediction methods.

In Chapter 5, we report on the application of the work from the previous chapters to the problem of seismic event prediction. We first discuss the preprocessing of raw seismic data, and continue by going through the process of system identification for both an energy and a moment seismic time series. This is followed by a discussion of various methods for measuring the performance of neural networks when applied to event prediction. We also discuss appropriate stopping criteria. We then show results for modelling and predicting both energy and moment seismic time series time delay neural networks (TDNNs) and finite impulse respnse neural networks (FIRNNs). Afterwards, we discuss the use of regularisation techniques on these neural networks. We conclude the chapter with a discussion of the results obtained. The last chapter contains our conclusions, and discuss various interesting possibilities for future research.

Chapter 2

Chaotic Systems

2.1 Introduction

Much of modern physics can be described by ordinary differential equations. However, many interesting systems exist for which the ordinary differential equations are unknown, and that are either highly nonlinear or chaotic. Examples of such systems are physiological processes such as the cardiorespiratory system [24], postural control [5], electrical activity in the brain during epileptic seizures [26] [36] [64], and voice generation [22]. Other systems include secure data communication [52] and laser control [25] [53]. We briefly discuss some of these systems.

2.1.1 Epileptic Seizure

Epilepsy is a symptom complex characterised by attacks of unconsciousness. An epileptic seizure is a state produced by an abnormal excessive neuronal discharge within the central nervous system [8]. Brain activity can be measured with an electroencephalogram (EEG) or an electrocorticogram or (ECoG). The electrodes of an EEG are attached to the scalp, and those of the ECoG are in direct contact with the cortex. Physicians and physiologists try to predict seizures by inspecting EEG or ECoG recordings. Seizures seem to occur abruptly, within a time scale of seconds. The EEG

signal is complex, non-stationary, and much of its power is concentrated in the 0-30 Hz range. The presence of such low frequencies is characteristic of very complex, strongly cooperative systems [26].

EEG and ECoG signals are chaotic. The development of chaos theory made the prediction of seizures possible. Some authors suggest that seizures may be predicted up to 10 minutes in advance [26]. Various seizure prediction methods have been proposed, including methods analysing the Lyapunov exponents [26] and methods analysing the correlation integral [36].

2.1.2 Cardiorespiratory System

The cardiovascular and respiratory systems have evolved to provide the body with an adequate continuous supply of oxygen and nutrients, as well as the clearance of waste products. The stable operation of these systems involves complex coordination. Nonlinear data analysis can attribute to predicting the risk of sudden cardiac arrest.

Some physiological processes are highly non-stationary, e.g. switches between different sleep s-tates. This necessitates data analysis on the the short-term. Investigations on the cardiorespiratory system have found that many aspects of the system are in fact highly nonlinear [24].

2.1.3 Secure Transmission

The fact that the trajectories of chaotic systems are by definition highly dependent on initial conditions, can be exploited for secure data transmission. For example, an information carrying signal can be transmitted together with a state variable from a chaotic attractor. Only the authorised receiver can recover the information signal since only he can separate the attractor signal, which is known to him, from the data signal [49].

When a low dimensional attractor is used to mask the information signal, the encryption can often be broken. Pyragas gives a method for improving the security of the information signal by utilising

the Mackey-Glass attractor [52]. This method uses the simple delay differential equation of the Mackey-Glass attractor to construct a very high dimensional attractor that has several positive Lyapunov exponents. The attractor can be based on a single Mackey-Glass system, or on a set of coupled Mackey-Glass systems. The receiver only needs to know the attractor equations, not the initial state. This method claims to be robust with respect to noise as well.

2.1.4 Chaos in Laser Measurements

It has been shown that NH_3 exhibits periodic and chaotic self-pulsing [25]. Accurate measurements of laser systems with good signal-to-noise ratios can be obtained. Thus, accurate analysis of these systems can be performed.

Analysis of the system can be used to predict system evolution. It can also be used to reduce the effect of noise on the measured data, or to constrain the motion to the close neighbourhood of an unstable periodic orbit. This latter procedure is called control. The first method for control has been proposed by Ott, Grebogi, and Yorke [48]. Reyl *et al.* successfully applied this method to a NMR laser [53].

2.1.5 Chapter Outline

We start by describing the various properties of chaotic attractors. We discuss state space description, different kinds of attractors, and differentiate between strange attractors and chaotic attractors. Next, we discuss the various methods available to identify chaotic time series. These methods include Fourier analysis, the correlation dimension, Lyapunov exponents, and the Poincaré map. We describe methods for extracting parameters that aid in attractor modelling including the method of delays, autocorrelation, mutual information, false nearest neighbours, and false strands. The methods of false strands and false nearest neighbours provides an estimation for the embedding dimension. We conclude the chapter by analysing well-known systems.

2.2 Chaotic Attractors

2.2.1 State Space Description

The state of a system is defined as the value of the smallest vector such that at time t_0 it completely determines system behaviour for any time $t \geq t_0$ [47]. The components of the state vector $\vec{x}$ are called state variables. The evolution of a system can be visualised as a path in state space. Since a dynamical system must contain memory elements, integrators can be used to describe the state space representation. The evolution of the state space can therefore be described with the following equations:

$$\dot{\vec{x}}(t) = f(\vec{x}, \vec{u}, t) \tag{2.1}$$

$$\vec{y}(t) = g(\vec{x}, \vec{u}, t) \tag{2.2}$$

where $\vec{u}$ and $\vec{y}$ are the system input and output, respectively. This set of differential equations completely describes the system. The collection of all possible states is called the phase space. Thus, the phase space is a subset of the state space.

2.2.2 Equilibrium Points, Periodic Solutions, Quasiperiodic Solutions and Chaos

As time goes to infinity, the asymptotic behaviour of a system that is not purely noise-driven can be categorised as being one of four general types: equilibrium points, periodic solutions, quasiperiodic solutions, or chaos [67].

An equilibrium point can be either stable or unstable. Stable equilibrium points are called sinks, and unstable points are called sources. A sink causes trajectories near it to move towards it with an increase in time, and is an example of an attractor.

When a trajectory precisely returns to itself, the system has a periodic solution with a fixed period T. The value of T is the time needed to reach the same point in state space again. A limit cycle is an example of an attractor that has a periodic solution.

The period of a quasiperiodic system is not fixed, i.e. the phase space is formed by the sum of periodic solutions that have periods whose ratio is irrational. A torus is an example of a quasiperiodic attractor.

These where the only known attractors until Lorenz's discovery in 1963. Lorenz discovered the first example of a chaotic attractor while searching for the solution of a model for weather prediction [62]. One feature common to these systems is that they are all completely deterministic. Once the system equations and the initial conditions are known, the system can be accurately described for all time.

Qualitatively speaking, a chaotic attractor is an attractor that is not of the previous three types. A system of this type is very dependent on initial conditions, i.e. two points in state space that are separated by a small distance will diverge exponentially as the system evolves. The exponents characterising the rate of separation are called the Lyapunov characteristic exponents and will be described in more detail in Section 2.3.3.

2.2.3 Fractals, Strange, and Chaotic Attractors

A system is considered *fractional* if it contains similar structures at all length scales [67]. Technically, this is known as self-similarity. A fractional system is often called a *fractal*. An attracting limit set is a set of stable asymptotic motions.

Definition 2.2.1 *A strange attractor is an attracting limit set that is chaotic.*

A strange attractor can also be defined as an attractor that is fractal, and the term strange refers to a static geometrical property.

Definition 2.2.2 *A chaotic attractor is an attractor that has a dependence sensitive to initial conditions.*

Two points in the state space of a chaotic attractor that are initially separated by a small distance will diverge exponentially as the system evolves. Thus, the term chaotic describes a dynamical property. It is possible that an attractor is strange, but not chaotic, although this is not usually the case [14]. We continue in Section 2.3 by describing how to identify a chaotic attractor.

2.3 System Characterisation

It is not always immediately clear what type of system has generated a time series. A number of techniques have been developed to establish the nature of a system. We describe some of these techniques.

2.3.1 Fourier Analysis

Any signal f(t) can be expressed as a Fourier series. The Fourier series for a discrete system is given by

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi k F_0 t} \tag{2.3}$$

where F_0 is a constant frequency, and c_k is given by

$$c_k = \frac{1}{T_p} \int_{T_p} x(t) e^{-j2\pi k F_0 t} dt \tag{2.4}$$

The power spectrum for the discrete signal is given by

$$P_k = |c_k|^2 \tag{2.5}$$

and the phase spectrum by

$$\Phi_k = \tan^{-1} c_k \tag{2.6}$$

The power spectrum represents the amount of energy associated with a specific frequency component, and the phase spectrum the phase angle associated with each frequency component. For an in-depth discussion of the continuous and discrete Fourier analysis, please see [51].

The power spectrum for a system that is periodic or quasiperiodic is characterised by dominating frequencies and sub-harmonics. Chaotic and stochastic systems are easily distinguishable from periodic or quasiperiodic systems, as they contain rich broadband power spectra, as well as widely varying phase spectra. Until the advent of chaos theory, systems with broadband spectra were perceived to be completely stochastic, and therefore to be unpredictable. Chaos theory showed that this is not necessarily the case; new tools were necessary to categorise systems with broadband power spectra.

2.3.2 Correlation Dimension

The difference between strange attractors and purely stochastic (random) processes is that the evolution of points in the phase space of a strange attractor has definite structure. The correlation integral provides a measure of the spatial organisation of this structure, and is given by

$$C(r) \equiv \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1}^{N} \theta(r - |\vec{X}_i - \vec{X}_j|) \equiv \int_0^r d^d r' c(\vec{r}') \tag{2.7}$$

where $\vec{X}_i = (x_i, x_{i+l}, x_{i+2l}, ..., x_{i+(m-1)l})$ for $i = 1, 2, 3, ..., N$ is a window of points from the time series, l an arbitrary (but fixed) constant, θ the Heaviside function, and $c(\vec{r}')$ the standard correlation integral. Grassberger and Procaccia found that, for a strange attractor, $C(r) \propto r^v$ for a limited range of r [19]. The power v is called the *correlation dimension* of the attractor. Thus, to identify an attractor, we plot the $\log C(r)$ - $\log r$ graph. The correlation dimension of the attractor is the gradient of the function at the point where the graph is linear. The correlation dimension is one for a periodic process, and two for a two dimensional quasiperiodic process. In contrast, the correlation dimension for a strange attractor can be a non-integer number, called a fractional dimension. In principle, the correlation dimension for a completely stochastic process is infinite. Thus, the correlation dimension can be used to distinguish a strange attractor from a purely stochastic process.

It is important to exclude temporally correlated points from the pair counting. This can be done by excluding points with indices $|i - j| < w$ where w is referred to as the Theiler window. The loss of $O(N)$ points is negligible given that $O(N^2)$ data points are available.

In Section 2.4.1, we show that an attractor can be represented by the method of delays. This method embeds the attractor by using m consecutive values of the time series generated by the attractor to represent points in a usually lower dimensional phase space. Ding *et al.* show that if the embedding dimension m is increased, the correlation dimension will increase until it reaches a plateau. They show that if a big enough data set is used, this value of the correlation dimension represents a good estimate for Ceil(D_2), where D_2 denotes the correlation dimension of the attractor in the original nonembedded full phase space [13]. For shorter data sets with observational noise, the value for D_2 can be underestimated [55]. For more discussion on the dimension of attractors, please see [17].

2.3.3 Lyapunov Characteristic Exponents

The above-mentioned techniques cannot conclusively prove that an attractor is indeed chaotic. Strangeness only suggests, but not necessary implies chaos. Positive Lyapunov exponents provide stronger evidence that the attractor is chaotic.

In a chaotic system, two nearby orbits in phase space diverge at an exponential rate. Thus, two nearly identical systems start behaving very differently as time progresses. The Lyapunov characteristic exponents quantify this property. The magnitude of the Lyapunov exponents reflects the time scale in which system dynamics becomes unpredictable. Formally, the Lyapunov spectrum is defined as follows [70]: given an n-dimensional phase space, the long-term evolution of an infinitesimal n-sphere is monitored. As the sphere evolves, it will turn into an n-ellipsoid. The ith one-dimensional Lyapunov exponent is then defined in terms of the length of the resulting ellipsoid's principal axis $p_i(t)$:

$$\lambda_i = \lim_{t \to \infty} \frac{1}{t} \log_2 \frac{p_i(t)}{p_i(0)} \tag{2.8}$$

The Lyapunov spectrum is then formed by the set $(\lambda_1, \lambda_2, ..., \lambda_n)$, where λ_i are arranged in de-

creasing order.

The λ_i are called the Lyapunov characteristic exponents, or Lyapunov spectrum. An attractor that has one or more positive Lyapunov exponents is chaotic [70]. We discuss the calculation of the Lyapunov spectrum next.

2.3.4 Calculating the Lyapunov Spectrum

Wolf *et al.* present an algorithm for calculating the Lyapunov exponents from a time series of experimental data [70]. First, the attractor is embedded, and the m-dimensional phase space constructed from the time series (see Section 2.4.1). The Euclidean distance $L(t_0)$ to nearest neighbour of the first point in phase space is calculated. This length element is evolved until it starts to shrink. Since the exponential expansion indicated by positive Lyapunov exponents is incompatible with motion on a bounded attractor, a *folding* process merges widely separated trajectories again. At this time, the length element has length $L'(t_1)$, and the data is searched for a point satisfying the following criteria: its separation $L(t_1)$ from the evolved point is small relative to the geometric size of the attractor, and the angular separation between the two points is small. If no such point can be found, the original point is retained. The procedure is continued until the data is exhausted. Thus, only the small scale structure of the attractor is examined. Therefore, the underestimation of λ_i is avoided by not measuring lengths in regions of the attractor that contain folds. The first Lyapunov exponent is then approximated by:

$$\lambda_1 = \frac{1}{t_M - t_0} \sum_{k=1}^{M} \log_2 \frac{L'(t_k)}{L(t_{k-1})} \tag{2.9}$$

where M is the number of replacements. Wolf *et al.* continue by giving an algorithm for calculating $\lambda_1 + \lambda_2$. Instead of following this route, we describe the algorithm by Eckmann *et al.* for calculating the complete spectrum [16]. In short, the algorithm consists of three steps: (a) reconstruct the dynamics in a finite dimensional space, (b) obtain the tangent maps to the reconstruction by a least-squares fit, and (c) deduce the Lyapunov exponents from the tangent maps. We now describe the algorithm in more detail.

Step (a): the time series is embedded with dimension d_E (see Section 2.4.1).

Step (b): Find matrix T_i that describes how the time evolution sends small vectors around $\vec{x}_i$ to small vectors around $\vec{x}_{i+1}$. Thus we define matrix T_i:

$$T_i(\vec{x}_j - \vec{x}_i) \approx \vec{x}_{j+1} - \vec{x}_{i+1} \tag{2.10}$$

Since the vectors $\vec{x}_j - \vec{x}_i$ may not span R^{d_E}, T_i may only be partially defined. Thus, it is assumed that there exists an integer m such that $m \geq 1$, and

$$d_E = (d_M - 1)m + 1 \tag{2.11}$$

with $d_M \leq d_E$. Associate with $\vec{x}_i$ a d_M-dimensional vector

$$\vec{x}_i = (x_i, x_{i+m}, ..., x_{i+(d_M-1)m}) \tag{2.12}$$

$$= (x_i, x_{i+m}, ..., x_{i+d_E-1}) \tag{2.13}$$

Thus, we are searching for T_i such that

$$T_i(\vec{x}_j - \vec{x}_i) \approx \vec{x}_{j+m} - \vec{x}_{i+m} \tag{2.14}$$

Even though $m > 1$, the distance measurements are still made in dimension d_E. We define $S_i^E(r)$ as the set of indices j of neighbours $\vec{x}_j$ of $\vec{x}_i$ such that

$$|\vec{x}_j - \vec{x}_i| \leq r \tag{2.15}$$

We compute the least square fit and obtain values for a_k by

$$\sum_{j \in S_i^E(r)} \left[\sum_{k=0}^{d_M-1} a_{k+1}(x_{j+km} - x_{i+km}) - (x_{j+d_M m} - x_{i+d_M m}) \right]^2 = \text{minimum} \tag{2.16}$$

The values a_k then define T_i such that

$$T_i = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_1 & a_2 & a_3 & \cdots & a_{d_M} \end{bmatrix} \tag{2.17}$$

Step (c): Define successively orthogonal matrices $Q_{(j)}$ and upper triangular matrices $R_{(j)}$ with positive diagonal elements such that

$$T_1 Q_{(0)} = Q_{(1)} R_{(1)} \tag{2.18}$$

$$T_{1+m} Q_{(1)} = Q_{(2)} R_{(2)} \tag{2.19}$$

$$\vdots$$

$$T_{1+jm} Q_{(j)} = Q_{(j+1)} R_{(j+1)} \tag{2.20}$$

where $Q_{(0)}$ is the unit matrix. Now, the Lyapunov exponents can be expressed as

$$\lambda_k = \frac{1}{m\tau K} \sum_{j=0}^{K-1} \ln R_{(j)kk} \tag{2.21}$$

where K is the available number of matrices, and τ the period that the time series was sampled with. There exist several commercial and public domain software packages for calculating the Lyapunov spectrum, and one does not need to implement these algorithms by one's self[1].

2.3.5 The Poincaré Map

Before describing the Poincaré map, we first define the two terms, maps and flows. At each point in state space, the differential equations for the system can be thought of as defining a vector at that point in space, determining the behaviour for the system at that point. Thus, the individual vectors describe local behaviour. An integral curve or trajectory is the solution to the set of differential equations. The flow of a system is defined as the collection of all solutions, or all integral curves [67].

Maps are the discrete time versions of flows. As flows are specified by differential equations, maps are specified by difference equations. Maps are easier to analyse and were therefore the first chaotic systems to be studied numerically.

Any continuous flow can generate a map by sampling the flow at points $t = nT$, where T is the sampling period and $n = 0, 1, 2, 3, ...$ To generate the Poincaré map, this sampling is done

[1]For this thesis we made use of the TISEAN package, available at *http://www.mpipks-dresden.mpg.de*

according to certain rules. Tufillaro *et al.* give the following formal definition: if γ is the orbit of a flow ϕ_t in R^n, choose a local cross section $\Sigma \in R^n$ about γ, such that Σ is of dimension $n-1$ and transverse to ϕ_t. Thus, $F(\vec{x})\cdot N(\vec{x}) \neq 0$ for all $\vec{x} \in \Sigma$, where $F(\vec{x})$ is the vector field defined at points $\vec{x}$ in phase space, and $N(\vec{x})$ the unit normal vector to Σ at $\vec{x}$. If $\vec{p}$ is a point where γ intersects Σ, and $\vec{q} \in \Sigma$ a point in the neighbourhood of $\vec{p}$, then the Poincaré map is defined by

$$P : \Sigma \to \Sigma, \quad P(\vec{q}) = \phi_\tau(\vec{q}) \tag{2.22}$$

and $\tau = \tau(\vec{q})$ is the time needed for an orbit starting at $\vec{q}$ to return to Σ.

To identify a flow on a Poincaré map the following general rules apply: a periodic motion will have a finite number of points on the Poincaré map whereas a quasiperiodic system will have an infinite number of points tracing out a closed contour [65]. Since a chaotic system never revisits the same state, it will trace out contours on the Poincaré map. However, unlike a purely random process, these contours will have definite structure and will graphically indicate the presence of the responsible attractor.

2.4 Modelling the Attractor

Having established that a system contains a chaotic attractor, we need to model the process, by reconstructing the state space. Two methods are available: the method of delays and principal component analysis.

We will not give a detailed description of principal component analysis, as we will use the method of delays for attractor modelling. Instead we refer the reader to work by Broomhead and King [7]. The method of delays is the most popular; recent work by Bakker *et al* proposes a method for improving principal component analysis and used it to model chaotic attractors [3].

2.4.1 Method of Delays

Broomhead and King also give a good introduction to the method of delays [7]. Before discussing the method of delays, we first define some mathematical concepts.

Definition 2.4.1 *A manifold is any smooth geometric space, e.g. a line, surface, or solid.*

Since the manifold is smooth, it cannot have any sharp edges [67]. Circles and lines are examples of one-dimensional manifolds, and the surface of a sphere is an example of a two-dimensional manifold. We defined the term *map* in Section 2.3.5.

Definition 2.4.2 *A map is a homeomorphism if it is bijective (one-to-one), continuous, and has a continuous inverse.*

Definition 2.4.3 *A diffeomorphism is a differentiable homeomorphism.*

Let the original system have phase space S with dimension s, and let the attractor of interest exist within S. It is generally the case that there exists a smooth manifold M with dimension $m < s$ such that the attractor can be described completely within M [7]. For initial states near the attractor, the asymptotic behaviour of the system will be such that the flow will lie on trajectories near the attractor. Thus, it is only necessary to describe the attractor with dimension m. We can describe the manifold M by way of an embedding.

Definition 2.4.4 *An embedding is a smooth map ϕ from the manifold M to a space U, such that its image, $\phi(M) \subset U$, is a smooth submanifold of U, and that ϕ is a diffeomorphism between M and $\phi(M)$.*

Thus, the embedding is a realization of M within U. Takens proved an important theorem that states that an embedding exists if the dimension d of U is such that $d \geq 2m + 1$:

Theorem 2.4.1 *For pairs (F, v), with F a smooth vector field, and v a smooth function on M, it is a generic property that*

$$\phi_{F,v}(\vec{y}) : M \to \Re^{2m+1} \tag{2.23}$$

where the embedding $\phi_{F,v}(\vec{y})$ is defined as

$$\phi_{F,v}(\vec{y}) = (v(\vec{y}), v(\psi_1(\vec{y})), ..., v(\psi_{2m}(\vec{y})))^T \tag{2.24}$$

where ψ_t is the flow of F.

This theorem gives us a way of representing the attractor, since $v(\vec{y})$ is the measurement made on an observable state variable while the system is in a state $\vec{y} \in M$. For this embedding, d is called the embedding dimension. Takens' theorem is strictly an existence theorem and makes no suggestions as how to find the embedding dimension, the sampling period τ_s, the lag time $\tau_l = J\tau_s$, or the window length $\tau_w = n\tau_l$. The methods for estimating these parameters will be the subject of further discussion in the following sections. For a mathematically thorough description of embedology, please see [58].

2.4.2 Estimating the Time Window

The time window τ_w is composed of two variables: the embedding dimension d, and the sampling period τ_s, such that $\tau_w = \tau_s(d-1)$. Theoretically, for perfect noiseless data, there exists no upper bound on either variable. However, this is almost never the case in practice where inaccurate measurements result in bounds on both variables. In order to introduce the minimum amount of noise into the reconstruction, it is beneficial to choose d as small as possible, such that a valid embedding still exists. Note that $d \geq 2m+1$ is a sufficient condition, not a necessary requirement.

There also exist lower and upper bounds on the sampling period. If τ_s is too small, consecutive vectors will be highly correlated, and the attractor will lie on the diagonal in R^d. This problem can usually be addressed by employing a lag time such that $\tau_w = \tau_l(d-1)$ where $\tau_l = J\tau_s$. When τ_s

becomes too long, folding may result, and the components may again be correlated. We discuss methods for estimating τ_s and τ_l next.

2.4.3 Autocorrelation

The autocorrelation function provides a measure of the similarity of a signal with a delayed version of itself. Thus, the autocorrelation can be used as a crude approximation for τ_s. We simply choose τ_s as the first point where the autocorrelation drops below a certain threshold, often chosen as $\frac{1}{e}$, or where the autocorrelation goes to zero. Autocorrelation only measures linear dependencies, and therefore only provides a first order approximation for τ_s.

2.4.4 Mutual Information

In 1986, Frasier and Swinney proposed *mutual information* to obtain an estimate for τ_s [18]. Mutual information provides a general measure for the dependence of two variables. Thus, the value of τ_s for which the mutual information goes to zero is preferred. Frasier and Swinney recommend using the first occurrence of zero mutual information as an estimation of the value for τ_s. Lieber *et al.* lists additional arguments for choosing the first zero [37]. We continue by describing the algorithm for computing the mutual information, according to Frasier and Swinney [18].

Mutual information is a measure found in the field of information theory. Let S be a communication system with $s_1, s_2, ..., s_n$ a set of possible messages with associated probabilities $P_s(s_1), P_s(s_2), ..., P_s(s_n)$. The entropy H of the system is the average amount of information gained from measuring s. H is defined as

$$H(S) = -\sum_i P_s(s_i) \log P_s(s_i) \tag{2.25}$$

For a logarithmic base of two, H is measured in bits. Mutual information measures the dependency of $x(t+T)$ on $x(t)$. Let $[s, q] = [x(t), x(t+T)]$, and consider a coupled system (S, Q). Then, for

sent message s and corresponding measurement s_i,

$$H(Q|s_i) = -\sum_j P_{q|s}(q_j|s_i)\log[P_{q|s}(q_j|s_i)] \tag{2.26}$$

$$= -\sum_j \frac{P_{sq}(s_i,q_j)}{P_s(s_i)}\log\frac{P_{sq}(s_i,q_j)}{P_s(s_i)} \tag{2.27}$$

where $P_{q|s}(q_j|s_i)$ is the probability that a measurement of q will result in q_j, subject to the condition that the measured value of s is s_i. Next we take the average uncertainty of $H(Q|s_i)$ over s_i,

$$H(Q|S) = \sum_i P_s(s_i)H(Q|s_i) \tag{2.28}$$

$$= -\sum_{i,j} P_{sq}(s_i,q_j)\log\frac{P_{sq}(s_i,q_j)}{P_s(s_i)} \tag{2.29}$$

$$= H(S,Q) - H(S) \tag{2.30}$$

with

$$H(S,Q) = -\sum_{i,j} P_{sq}(s_i,q_j)\log P_{sq}(s_i,q_j) \tag{2.31}$$

The reduction of the uncertainty of q by measuring s is called the mutual information $I(S,Q)$ which can be expressed as

$$I(Q,S) = H(Q) - H(Q|S) \tag{2.32}$$

$$= H(Q) + H(S) - H(S,Q) = I(S,Q) \tag{2.33}$$

where $H(Q)$ is the uncertainty of q in isolation. If both S and Q are continuous, then

$$I(S,Q) = \int P_{sq}(s,q)\log\frac{P_{sq}(s,q)}{P_s(s)P_q(q)}dsdq \tag{2.34}$$

If s and q are different only as a result of noise, then $I(S,Q)$ gives the relative accuracy of the measurements. Thus, it specifies how much information the measurement of x_i provides about x_{i+1}. The mean and variance of the mutual information estimation can be calculated [44].

Although mutual information guarantees decorrelation between x_k and x_{k+t}, and between x_{k+t} and x_{k+2t}, it does not necessary follow that x_k and x_{k+2t} are also uncorrelated [33]. Kugiumtzis proposes setting τ_w equal to the mean orbital period, which can be taken as the mean time between peaks for low dimensional systems.

2.4.5 False Nearest Neighbourhoods

Mutual information gives an estimate for τ_s, but does not determine the embedding dimension d. The theorem by Takens states that an m-dimensional attractor will be completely unfolded with no self-crossings if the embedding dimension is chosen larger than $2m$. Thus, it is certain that an embedding does exist, but we need a method for finding a good value for d. The method of false nearest neighbours gives an algorithm for estimating d [31].

The method is based on the idea that two points close to each other (called neighbours) in dimension d, may in fact not be close at all in dimension $d + 1$. This can happen when the lower dimensional system is simply a projection of a higher dimensional system, and is unable to completely describe the higher dimensional system. Thus, the algorithm searches for "false nearest neighbours" by identifying candidate neighbours, increasing the dimension, and then inspecting the candidate neighbours for false neighbours. When no false neighbours can be identified, it is assumed that the attractor is completely unfolded, and d at this point taken as the embedding dimension.

2.4.6 False Strands

The method of false strands is an improvement on the method of false nearest neighbours [30]. The method of false strands addresses problems arising from high sampling rates. The false nearest neighbour method may classify all neighbours as true neighbours if the sampling rate is too high. The solution is to ignore points within a temporal decorrelation interval around the point being tested.

The method of false nearest neighbours has a second flaw. Consider two orbit segments of an attractor of dimension $d + 1$. In dimension d, the projection of the higher dimensional orbit is a straight line, which has no folds, but in dimension $d + 1$ the orbit folds and makes a strong angle. Thus, the neighbours in dimension d remain neighbours in dimension $d + 1$, and a wrong (lower dimensional) classification is made. This problem is addressed by searching for false strands

instead of false neighbours. Strands are series of neighbours. Two strand pairs are false if the average extra distance between them with an increase in dimension is large. The dimension is increased until the ratio of false strands to the total number of strands is small.

A global linear transformation of the state space to remove all linear cross-correlation among the components has been proposed [30]. The method renders each auto-correlation unity. If the embedding delay time is too short, then the data becomes highly correlated, and $\vec{y}_1 = \vec{y}_2 = ... = \vec{y}_d$, which results in a too small false strand ratio. The authors suggest using a combination of principal component analysis and orthogonal rotation. The final data to be embedded, and used by the false strands method, is contained in an N by $d+1$ matrix B,

$$B = \frac{AVD^{-1}Q}{N} \tag{2.35}$$

where A is the N by d matrix with $A_{ij} = y(i)_j$ and

$$A = UDV^T \tag{2.36}$$

with U and V orthogonal, and D diagonal. Finally, the orthogonal transformation Q is given by

$$Q = H(\vec{x} - |\vec{x}|\vec{e}_{d+1}) \tag{2.37}$$

with $\vec{e}_{d+1}$ the unit vector in direction $d+1$, and the Householder matrix $H(\vec{z}) = \frac{I - 2\vec{z}\vec{z}^T}{|\vec{z}|^2}$. For the details of the derivation of this result, see [30].

We now have tools for estimating the embedding dimension and the sampling period of a chaotic time series. In order to measure their performance, we apply these methods to some well-known chaotic attractors: Lorenz, Rössler, Mackey-Glass, and laser data, as well as a purely stochastic system for comparison.

2.5 Application to Well-Know Attractors

2.5.1 The Lorenz, Rössler and Mackey-Glass Attractors

The differential equations for the Lorenz, Rössler and Mackey-Glass attractors are well known. Thus, they have often been used in the literature to study chaotic attractors. We first define the systems, and then we evaluate the system identification and modelling parameter extraction techniques.

The Mackey-Glass differential equation comes from the field of physiology [40] and is given by

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t) \tag{2.38}$$

The Lorenz equations were derived from a model of fluid convection [62]. They are given by

$$\frac{dx}{dt} = \sigma(y-x) \tag{2.39}$$

$$\frac{dy}{dt} = rx - y - xz \tag{2.40}$$

$$\frac{dz}{dt} = xy - bz \tag{2.41}$$

The Rössler equations have no real physical interpretation [54], and are given by

$$\frac{dx}{dt} = -(y+z) \tag{2.42}$$

$$\frac{dy}{dt} = x + 0.2y \tag{2.43}$$

$$\frac{dz}{dt} = 0.2 + z(x-5.7) \tag{2.44}$$

For each system, we only one consider one state variable. This produces the scalar time series shown in Figure 2.1. We begin by exploring the power spectrum for each time series. Note that we define the sampling period as $1s$. Thus, the highest possible frequency component present in the signal will be 0.5Hz.

Figure 2.2 shows that none of these systems have sharp peaks with sidebands at subharmonic frequencies. However, each graph shows definite structure and further analysis is necessary to

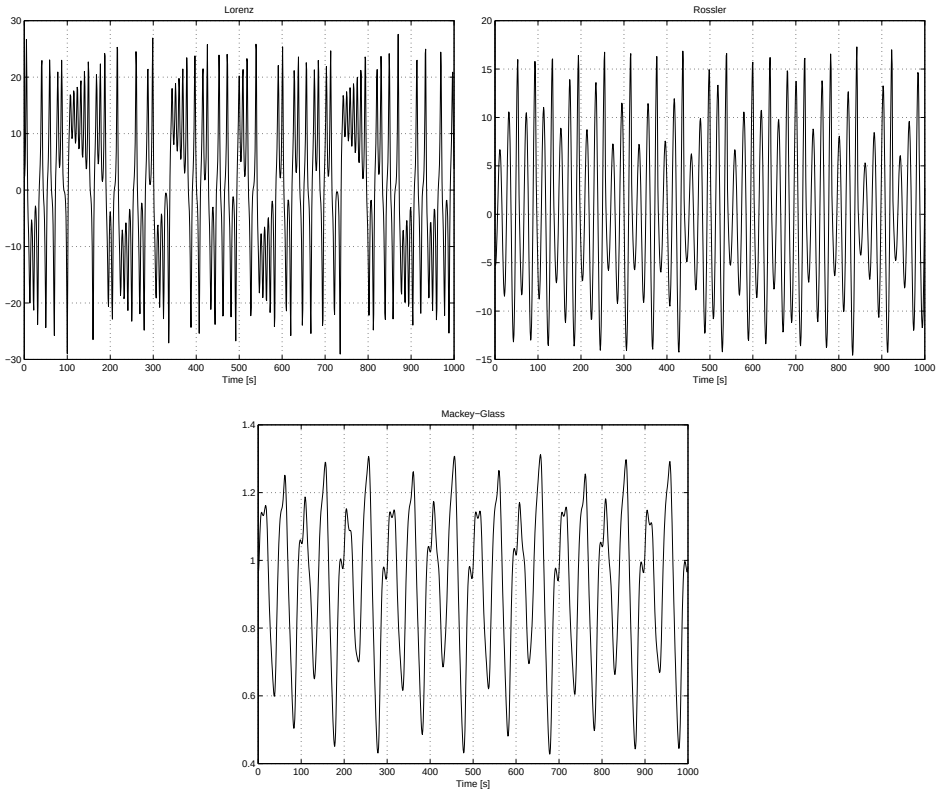

Figure 2.1: The Lorenz, Rössler, and Mackey-Glass time series.

determine the nature of the governing system. Thus, we continue by calculating the correlation dimension for each system.

We show the correlation integral and correlation dimension graphs for the time series in Figure 2.3. Each curve on the graph represents an embedding dimension. The correlation integral has a skewed S-shape and its y-axis scale is shown on the right border of Figure 2.3. The graphs demonstrate the diminishing influence of higher embedding dimensions on the correlation dimension. The slope of the most linear section of the correlation integral graph determines the correlation dimension. The curves for the correlation integral estimation are noisy, and it is difficult to obtain a precise estimation for their slopes. However, if one take the average of the correlation dimensions at their most linear sections, none of the time series have integer correlation dimensions. Thus, we deduce that all the time series are fractal of nature.

The Lyapunov spectrum for each time series is shown in Figure 2.4. Since we are interested

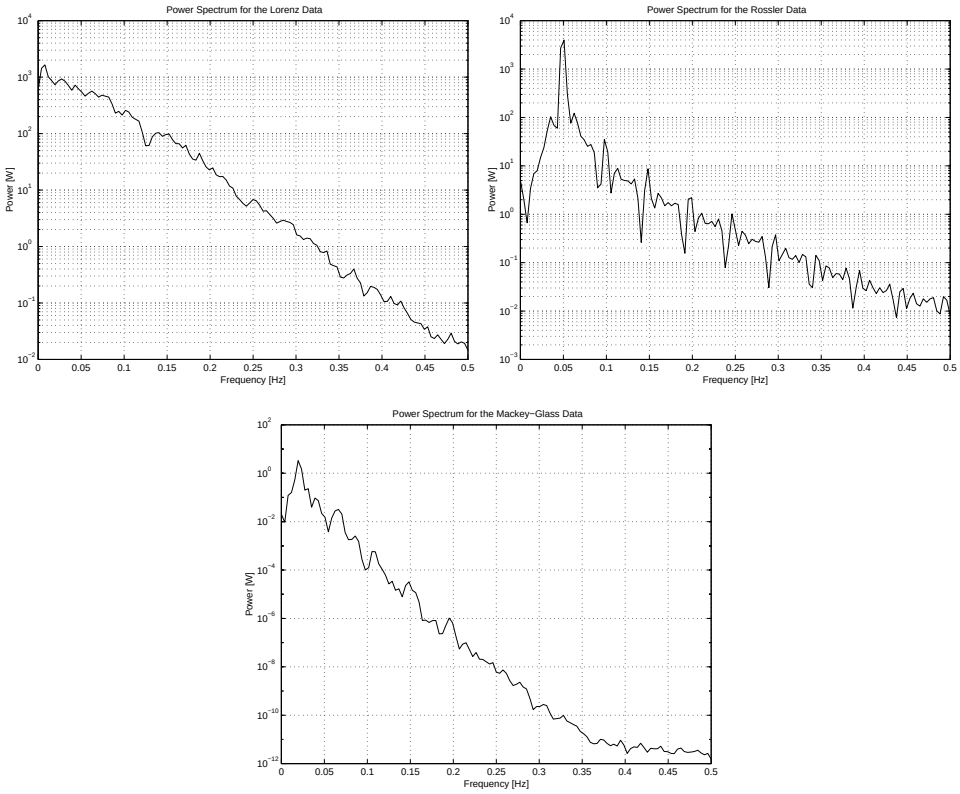

Figure 2.2: The power spectra for the Lorenz, Rössler, and Mackey-Glass time series.

in establishing whether there are positive exponents, we only show the largest exponents. All three time series contain positive Lyapunov characteristic exponents. This indicates the presence of chaotic attractors for all three time series, as illustrated by their three dimensional maps $(x(t), x(t-1), x(t-2))$ shown in Figure 2.5.

Next, we estimate the lag time τ_l using the methods of autocorrelation and mutual information. The lag time can be taken at the first zero on the autocorrelation graph, or at the first minimum on the mutual information graph. From the graphs shown in Figure 2.6, we observe that the mutual information criterion prescribes shorter lag times than autocorrelation. Specifically, from the mutual information graphs we read the lag times for the Lorenz, Rössler, and Mackey-Glass time series as 5s, 6s, and 10s, respectively.

Finally, we estimate the embedding dimension using both the false nearest neighbours and false strand methods. Both methods specify the embedding dimension as the embedding at the point

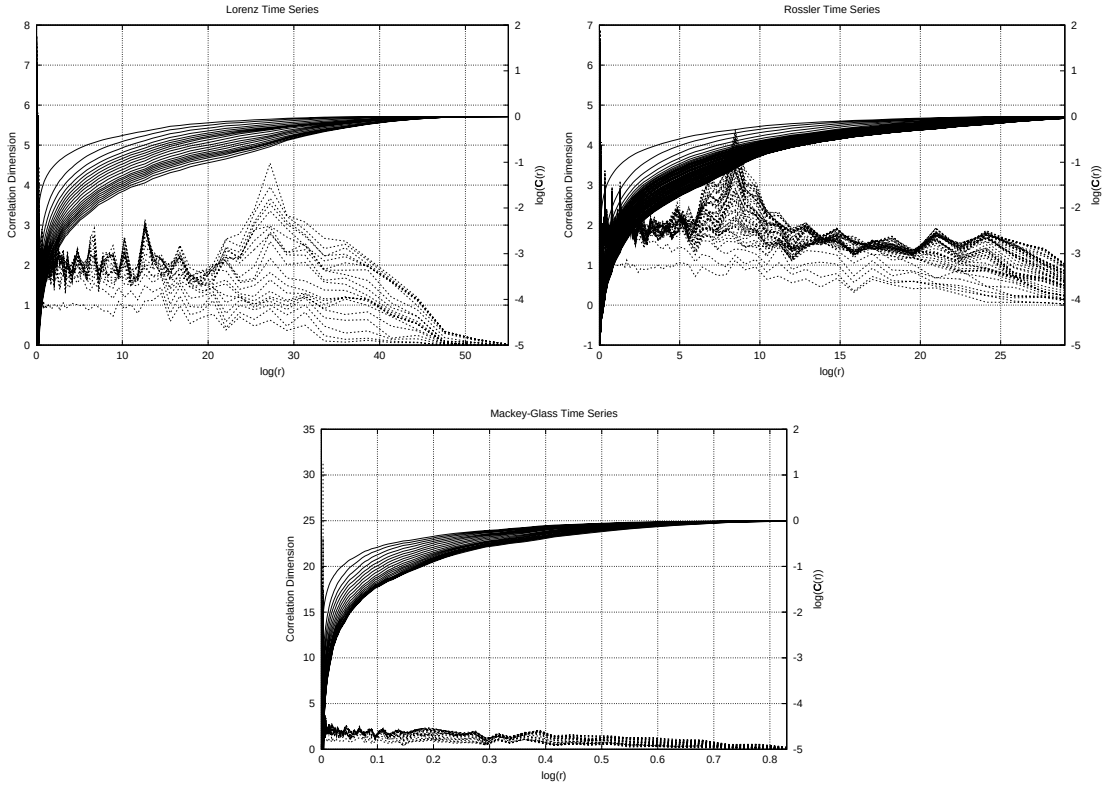

Figure 2.3: The correlation integral and dimension curves for the Lorenz, Rössler, and Mackey-Glass time series. Each curve on the graphs represents an embedding dimension. The correlation integral has a skewed S-shape, and its y-axis is shown on the right border.

where the graph first goes to zero. Figure 2.7 shows that both methods suggest an embedding dimension of 4 for the Rössler system. The false nearest neighbours method suggests an embedding dimension of 4 for both the Lorenz and Mackey-Glass systems. The false strand method suggests embedding dimensions of 3 and 6 for the Lorenz and Mackey-Glass systems, respectively.

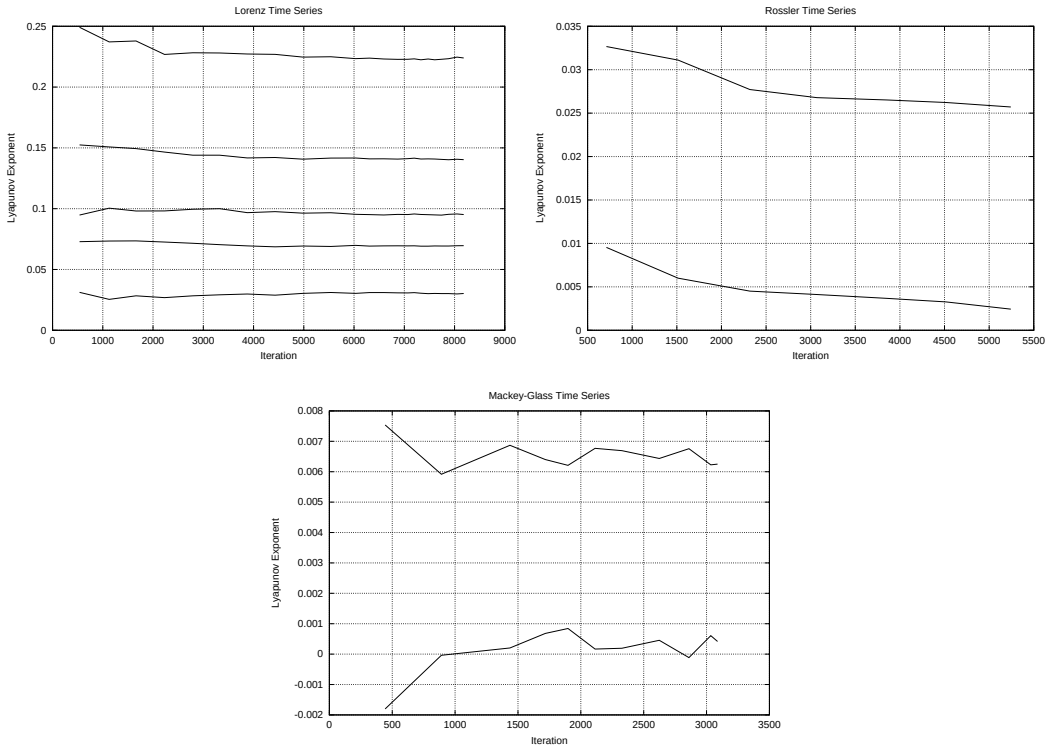

Figure 2.4: The Lyapunov spectrum for the Lorenz, Rössler, and Mackey-Glass time series. Note that, as discussed in Section 2.3.3, the Lyapunov exponents are calculated by evolving length elements. The graph shows the size of the exponents as they are evolved.

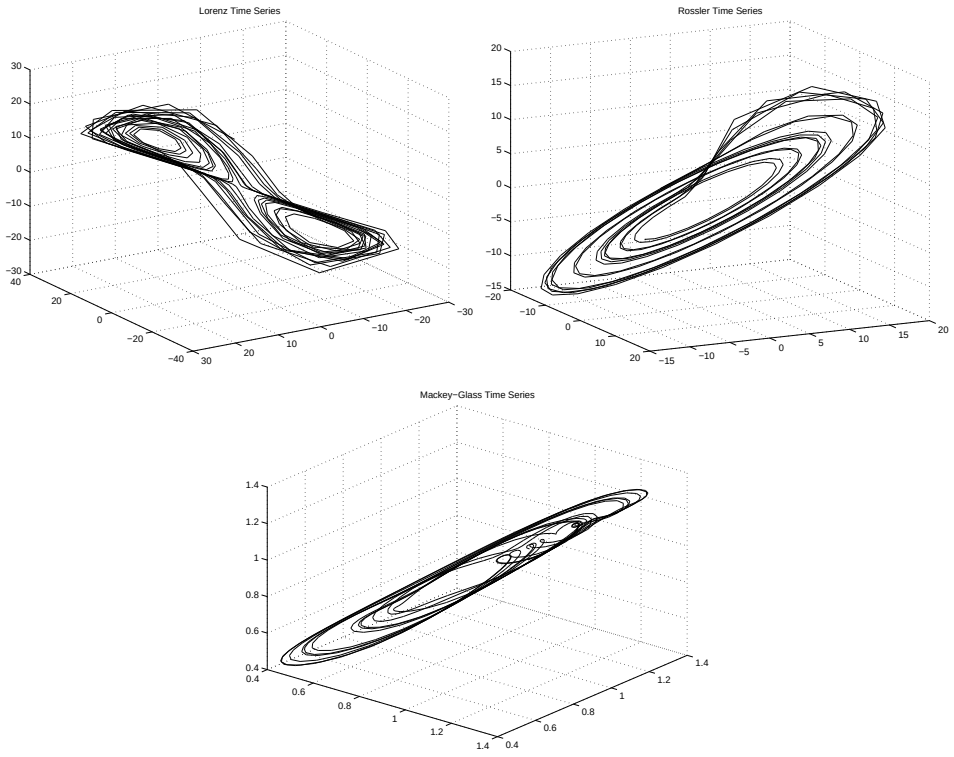

Figure 2.5: The Lorenz, Rössler, and Mackey-Glass attractors.

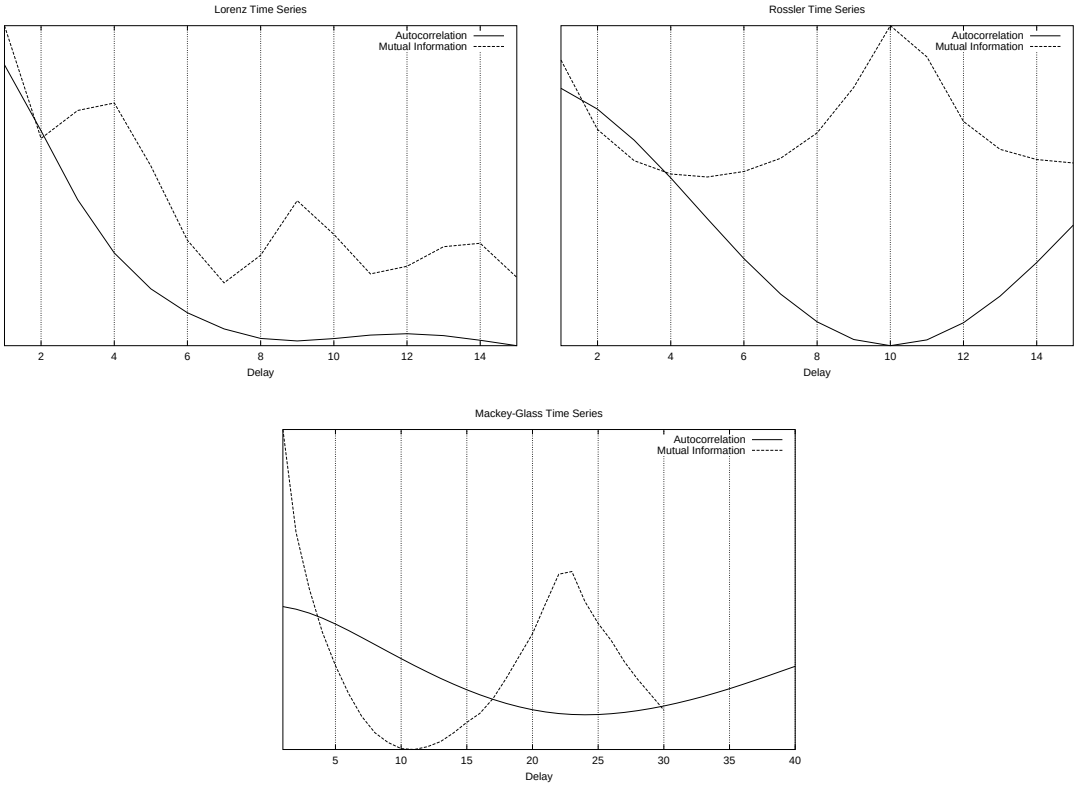

Figure 2.6: The autocorrelation and mutual information for the Lorenz, Rössler, and Mackey-Glass time series. Both measures give an estimation for the time lag between the first two uncorrelated time series data points.

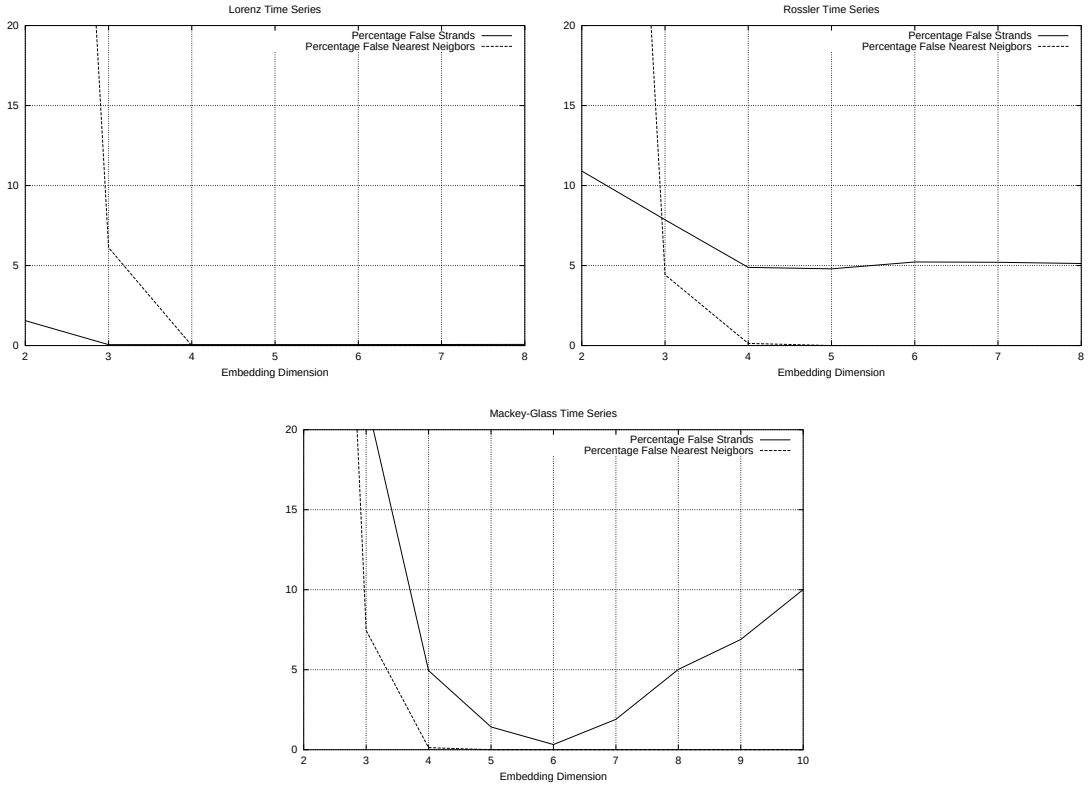

Figure 2.7: The false nearest neighbours and false strand computation for the Lorenz, Rössler, and Mackey-Glass time series provide estimations for the embedding dimensions of the respective attractors.

2.5.2 The Laser and Random Data Time Series

The laser time series contains measurements made on a far-infrared laser, as described in [25]. Figure 2.8 shows this time series, as well as a random time series with a uniform distribution. The power spectrum graph shown in Figure 2.9 indicates that the laser data is generated by a dynamical system. The random data seems a bit suspect, but we cannot say that it is a random process yet.

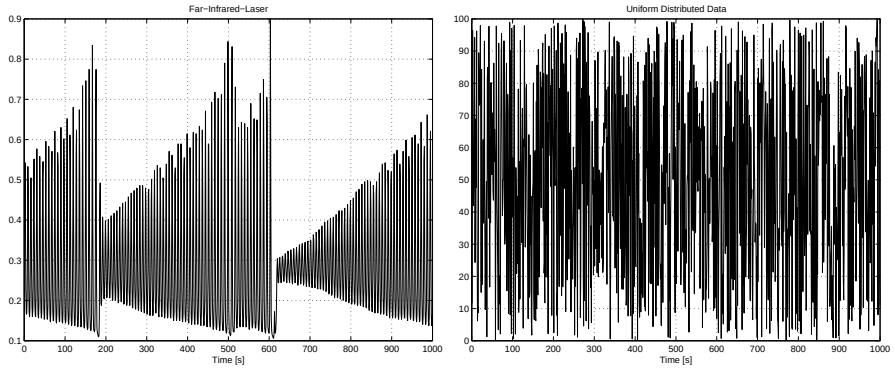

Figure 2.8: The laser and random time series.

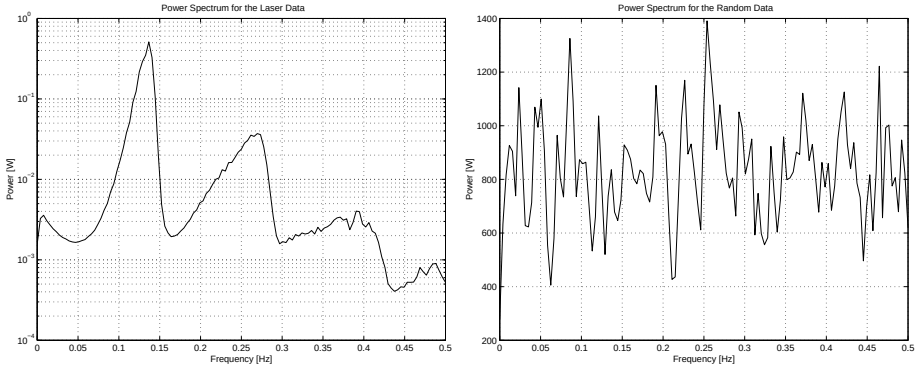

Figure 2.9: The power spectra for the laser and random time series.

The correlation sum and correlation dimension graphs for the laser and random time series are shown in Figure 2.10. We observe that that the random process has a much higher correlation dimension, and that the correlation dimension for neither time series is integral. This is an indication of strangeness. Note that we stated that in principle, the correlation dimension of random processes are infinite. However, this is only true when using time series of infinite size. When using time series of finite size, high correlation dimensions are indicative of randomness.

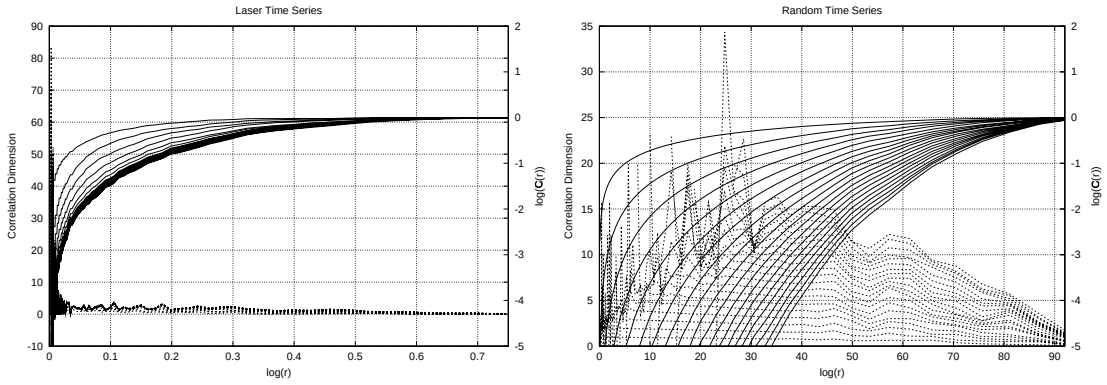

Figure 2.10: The correlation sum and dimension for the laser and random time series. The y-axis of the correlation sum is shown on the right border.

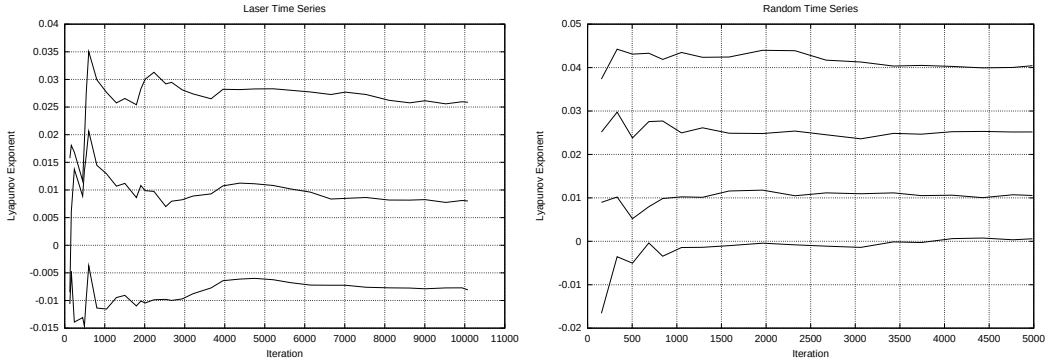

Figure 2.11: The Lyapunov spectra for the laser and random time series.

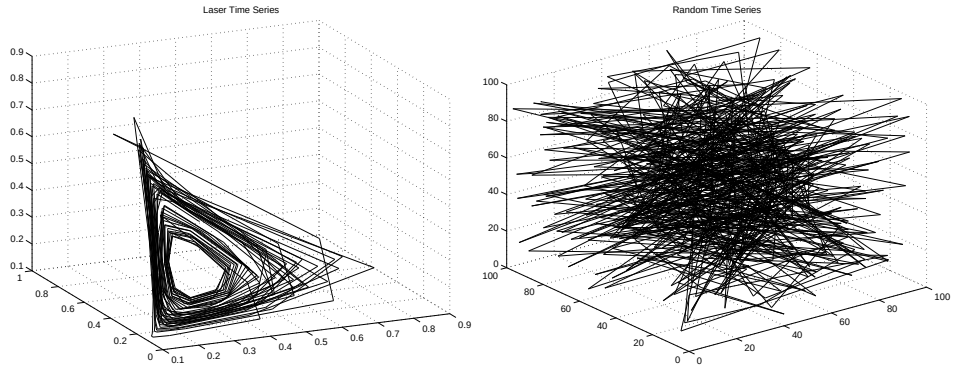

Figure 2.12: The laser and random data maps.

The Lyapunov spectrum for both systems are shown in Figure 2.11. The existence of positive exponents for both systems indicates chaotic behaviour. However, the maps shown in Figure 2.12 illustrate the difference between the two systems. The trajectories of the laser data have definite structure, whereas those of the random data fills the entire state space, and show not even a projection of an attractor. From all the evidence we conclude that the laser data can be modelled, but not the random data.

2.6 Summary

In this chapter, we discussed chaotic time series. Chaos is a relatively new concept, but has attracted much attention since its discovery in the 1960's. The reason for this attention is that many interesting time series (from both a scientific and economic viewpoint) are chaotic. Chaotic systems may, or may not be deterministic. However, only short-term predictions are possible. We discussed methods for identifying chaos in time series. The most used method, the Lyapunov spectrum, gives an estimation of how fast the flow of the attractor expands or shrinks, and thus how predictable the time series is. We also discussed methods for extracting modelling parameters from chaotic time series. We demonstrated the applicability of these methods. Thus, we can now focus our attention on modelling and predicting chaotic time series.

Chapter 3

Neural Networks for Time Series Modelling and Prediction

3.1 Introduction

Neural networks are used to solve two general types of problems: static and time-varying pattern recognition. Multilayer perceptrons are capable of representing arbitrary input/output mappings [4]. Examples of static problems are optical character recognition, DNA analysis and credit scoring systems. There exist two kinds of spatio-temporal problems: time series classification and prediction. Speech recognition and wavelet classification are examples of time series classification. Control and financial time series prediction are examples of prediction problems. Time series prediction can be performed one-step-ahead, multiple-step-ahead, or in closed loop. One-step-ahead prediction uses only past values of the time series for prediction. Multi-step-ahead prediction uses predicted outputs as well. Multi-step-ahead prediction operates the network in closed loop, and predicted outputs are used as if they were time series data points. In this way, predictions further into the future can be made. Neural networks with no feedback loops have limited context, i.e. they have limited memory, and thus only a limited knowledge of the history of the system. Recurrent networks have feedback loops, and consequently have unlimited context. They were designed

specifically to model time-varying patterns. For the remainder of this thesis, we will focus our attention on predicting spatio-temporal patterns.

We continue in the next section with an introduction to time delay feed forward networks (TDNNs). We discuss the myopic mapping theorem which provides the mathematical justification for the use of TDNNs as time series predictors. We discuss finite impulse response neural networks (FIRNNs) and recurrent networks next.

We continue the discussion by describing several techniques for network regularisation. We also discuss techniques for improving the error backpropagation training algorithm. Next, we present the use of knowledge-based artificial neural networks to improve training. We also discuss an early stopping criterion, called Diks' Test. Finally, we apply the discussed methods on the time series analysed in Chapter 2.

3.2 Time Delay Neural Networks

We begin this discussion by defining a neuron. A neuron can be described by the equation

$$y_j = \sigma(\sum_{i=0}^{p} w_{ji} x_i) \tag{3.1}$$

where y_j is the output of the neuron, x_i is a set of i inputs, each connected via a weight w_{ji} and $\sigma(v)$ is a nonlinear activation function. The sigmoidal activation function is by far the most common, and is given by

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

Usually, input x_0 is assigned a constant value of plus one, and is called the bias. The weight w_{j0} that connects the bias to the neuron is called the bias weight; it is also often denoted as b_j. The neuron is shown in Figure 3.1.

Neurons are the building blocks of neural networks. All the neurons of a multilayer perceptron (MLP) are connected such that layers of neurons can be defined. For feedforward neural networks,

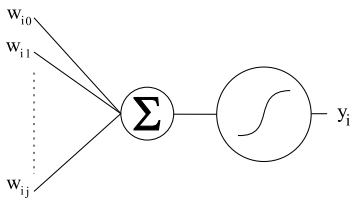

Figure 3.1: A neuron.

the output of neurons are connected as inputs of neurons in layers downstream. For the purposes of this discussion, no neurons will be connected such that a neuron in layer l will be connected as the input for a neuron in layer $l + 2$. Figure 3.2 shows a MLP with three layers. The layers are referred to as the input, hidden, and output layer, respectively. The input layer has three neurons, the hidden layer four neurons, and the output layer one neuron.

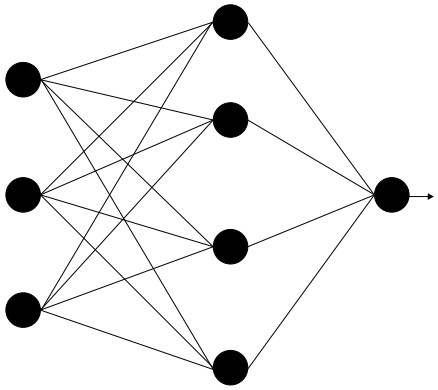

Figure 3.2: A three layer feedforward neural network. The input, hidden, and output layer have three, four, and one neurons, respectively.

We add limited temporal context to the feedforward network by adding short-term memory in the form of a tap delay line (see Figure 3.3). A section of the time series (called the time window) of the form $[x(n), x(n-1), ..., x(n-p)]$ is used as input for the feedforward network; the tap delay line is of order p, and the desired output is $x(n+1)$. This network is referred to as the time delay neural network (TDNN).

Adjusting the weights of the network to fit the problem at hand, is referred to as training the network. There exists various training methods. Here we only introduce one method, called the error backpropagation algorithm. The error backpropagation algorithm for MLPs is described in

more detail in Appendix A.1. This algorithm consists of three distinct phases. During the first phase the weights are initialised to small random values. This random initialisation prevents detrimental symmetrical effects that arise when networks have all their weights the same size. A good choice is to initialise the weights such that the standard deviation of weights of a neuron lies in the transition area between the linear and saturated parts of its sigmoid activation function [21]. During the forward passing phase, input is applied to the input neurons, and then propagated through the network.

The forward pass is followed by a backpropagating phase, during which the weights of the network are adjusted using a gradient descent method. This method minimises the error between the actual and desired output. Given the instantaneous error E at step n,

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \tag{3.3}$$

where $e_j(n) = d_j(n) - y_j(n)$ is the difference between the desired and obtained output at neuron j, and C is the set of neurons in the output layer, then

$$\Delta w_i(n) \propto -\frac{\partial E(n)}{\partial w_i} \tag{3.4}$$

where w_i denotes weight i. The size of updates is usually controlled by a proportionality constant called the learning rate. The update rule can also be augmented with a momentum term, which is proportional to the size of the previous update.

3.3 Universal Myopic Mapping Theorem

The mathematical justification for the use of TDNNs for time series prediction was given in 1997 by Sandberg and Xu [57]. We first define a few terms.

Definition 3.3.1 *A system consisting of a bank of linear filters operating in parallel, and a static (memoryless) feedforward network such as a MLP, is a universal dynamic mapper.*

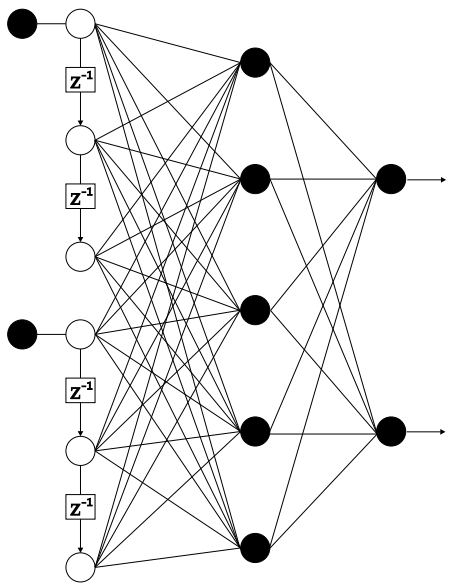

Figure 3.3: A time delay neural network. The input layer has two input neurons, each with tap delay line of order two.

Definition 3.3.2 *A myopic map has uniformly fading memory.*

Definition 3.3.3 *For a system to be causal, the output signal at step $n = 0$ may not depend on input signals applied at $n > 0$.*

Definition 3.3.4 *For a system to be shift invariant the following must hold: if the output $y(n)$ is obtained as a result of the input $x(n)$, then the delayed input $x(n - n_0)$ must have output $y(n - n_0)$, where n_0 is called the time shift, and is an integer.*

The universal myopic mapping theorem holds under the conditions that the map is causal, and can be formulated as follows [21]:

Theorem 3.3.1 *Any shift-invariant myopic dynamic map can be uniformly approximated arbitrarily well by a structure consisting of two functional blocks: a bank of linear filters feeding a static neural network.*

Sandberg and Xu showed that for any single-variable, shift-invariant, causal, uniformly fading memory map there exists a gamma memory (of which the normal time delay line is a special case) and static network of which the combination approximates the map uniformly and arbitrary well.

Thus, the universal myopic mapping theorem separates the roles of the short-term memory and the nonlinearity. Furthermore, since a static neural network is stable, the combination of a bank of linear filters and a static neural network will also be stable if the linear filters are stable. Since the TDNN fulfils all the requirements of the theorem, it can be used to model and predict time series.

3.4 Finite Impulse Response Neural Networks

The FIRNN builds on the TDNN by adding tap delay lines to every neuron in the network. The network is distributive since the short-term memory elements, and therefore the influence of recent events, are distributed throughout the network.

There are two ways to visualise the FIRNN. The tap delay lines can be thought of as storing the last p outputs of every neuron. Another equivalent visualisation is to see each weight in the feedforward network as changing into a vector of p weights. The FIR part of the term FIRNN stems from the fact that the extended weights can be thought of as FIR filters [69]. Figure 3.4 shows a FIRNN. The training algorithm for this kind of architecture is called the temporal error backpropagation algorithm, and is described in Appendix A.2.

Although the FIRNN seem more complex than the TDNN, it can been shown that TDNNs and FIRNNs are computationally equivalent, i.e. they both are able to represent the same class of computational models. In fact, the following theorem holds true:

Theorem 3.4.1 *For any FIRNN there exists an equivalent TDNN that computes the same function.*

Proof: We give a constructive proof that shows how an arbitrary FIRNN can be transformed into an equivalent TDNN.

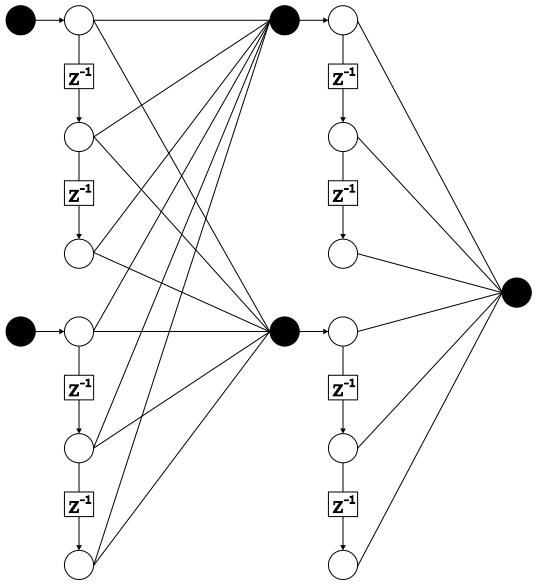

Figure 3.4: A Finite Impulse Response Neural Network. The neurons in both the input and hidden layer all have tap delay lines of order two.

Consider the FIRNN shown in Figure 3.4. The figure shows two successive layers i and $i+1$ with weight matrices W_i and W_{i+1}, respectively. We use the following representation for layer i of a FIRNN:

$$W_i = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdots & \theta_1 \\ w_{21} & w_{22} & w_{23} & \cdots & \theta_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & w_{n_i m_i} & \theta_{n_i} \end{bmatrix} \tag{3.5}$$

where n_i is the number of neurons in the layer and m_i is the number of weights connecting neurons to tap delay inputs.

The objective is to transform the two weight sets W_i and W_{i+1} into two equivalent weight sets V_i and V_{i+1}, such that layer $i+1$ has no tap delay lines. To achieve this, we create two new layers with weight matrixes V_i and V_{i+1}. Without loss of generality, we assume that, for the FIRNN, the order p_i of all the tap delay lines of all the neurons in any layer i is the same. Thus, V_i has $n_i p_{i+1}$ neurons, each with a tap delay line of order $(p_i + p_{i+1} - 1)$; V_{i+1} has n_{i+1} neurons with no taps.

We begin by duplicating W_{i+1} such that $V_{i+1} = W_{i+1}$. Next, we compute the weights of V_i such that equivalence of the two networks is maintained. The tap delays in layer $i+1$ of the FIRNN are used to store the previously computed outputs. These stored outputs must be computed again by the TDNN, and is the reason for the $(p_{i+1}-1)$ extra neurons in V_{i+1}, and the $(p_{i+1}-1)$ extra tap delays added to V_i. The weight matrix V_i is computed using the following algorithm:

Copy the weights of W_i into V_i and set any weights connecting any added neurons or taps to zero.

$$V_i = \begin{bmatrix} w_{11} & \cdots & w_{1p_i} & 0 & \cdots & w_{1(n_i p_i)} & \cdots & 0 & \theta_1 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{21} & \cdots & w_{2p_i} & 0 & \cdots & w_{2(n_i p_i)} & \cdots & 0 & \theta_2 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{3.6}$$

To compute the outputs for previous time steps, the same weights are used, connected in the following way:

Start with the row 1 of V_i:

Duplicate this row into the next row.

For each tap delay line inside this row:

Shift its current weight values one step to the right.

The zero at the right rolls over into a column of the delay line.

Continue until the next non-zero row is reached and then repeat the procedure until the whole matrix is filled.

$$V_i = \begin{bmatrix} w_{11} & \cdots & w_{1p_i} & 0 & \cdots & w_{1(n_ip_i)} & \cdots & \cdots & 0 & \theta_1 \\ 0 & w_{11} & \cdots & w_{1p_i} & 0 & \cdots & w_{1(n_ip_i)} & \cdots & 0 & \theta_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & w_{11} & \cdots & w_{1p_i} & 0 & \cdots & w_{1(n_ip_i)} & \cdots & \theta_1 \\ w_{21} & \cdots & w_{2p_i} & 0 & \cdots & w_{2(n_ip_i)} & \cdots & \cdots & 0 & \theta_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & w_{(n_ip_{i+1})1} & \cdots & w_{(n_ip_{i+1})p_i} & 0 & \cdots & w_{(n_ip_{i+1})(n_ip_i)} & \cdots & \theta_{n_ip_{i+1}} \end{bmatrix} \tag{3.7}$$

Now we have created two layers where the second has no tap delays. Furthermore, V_i has the same number of inputs as W_i, and V_{i+1} has the same number outputs as W_{i+1}. Starting with the output layer, this algorithm can be applied iteratively to transform a FIRNN of arbitrary size into an equivalent TDNN.

Even though FIRNNs and TDNNs are computationally equivalent, their training and generalisation performance are not equivalent. As is clear from the proof above, FIRNNs impose restrictions on the weights of equivalent TDNNs. Certain weights are required to have the same value. The reason for this weight duplication lies in the assumption that there are correlations in time. Another factor to consider, is that for the same number of weights, a FIRNN covers a wider time window, and can thus learn correlations further back in time.

FIRNNs were applied successfully to chaotic time series prediction. The laser data analysed in Section 2.5.2 was modelled and predicted by a four-layer FIRNN [69]. The results obtained by the FIRNN for the laser data were superior to all other prediction methods, and won the *Sante Fe Institute Time Series Prediction and Analysis Competition*.

3.5 Recurrent Neural Networks

Feedforward network structures such as TDNNs and FIRNNs can only have a limited context. Recurrent neural networks have feedback loops, i.e. neuron output is used as input to neurons in the same or previous layers. In this way the limitation on the input set is overcome, and all the available input up to $x(n)$ is used to compute $x(n+1)$. Thus, recurrent networks have, in principle, infinite context, and can therefore address the temporal relationship of the inputs. During the training of a TDNN, the examples can be presented in arbitrary order. This is not the case with recurrent networks, which maintains an internal state.

There exist many different recurrent network architectures. We will discuss an architecture called NARXNN (Nonlinear AutoRegressive models with eXogenous input Neural Networks) as an example of a recurrent neural network. As opposed to the fully recurrent networks, which have feedback from the hidden state neurons, the NARXNN have feedback only from the output neuron. It has been shown that fully recurrent networks are computationally at least as powerful as Turing machines [61]. It has since been proven that NARXNN are at least as strong as fully recurrent neural networks (and thus Turing machines) within a linear slowdown [60]; only a finite number of nodes and taps are needed to obtain this result.

The output of a NARXNN is given by

$$y(t) = \Psi(u(t-n_u), ..., (u(t-1), u(t), y(t-n_y), ..., y(t-1)) \tag{3.8}$$

where $u(t)$ is the input and $y(t)$ the output at time t, n_u and n_y the input and output order, respectively, and Ψ is the mapping of the multilayer perceptron. The NARX neural network architecture is shown in Figure 3.5. It is trained by using the error backpropagation through time (BPTT) training algorithm, which ”unfolds” the network in time and then computes error gradients. The NARXNN architecture has been used to model various systems, such as heat exchangers and wastewater treatment plants, and for tasks such as nonlinear system identification and the control of dynamic systems [39].

Recurrent neural networks are more powerful than non-recurrent network architectures. This added

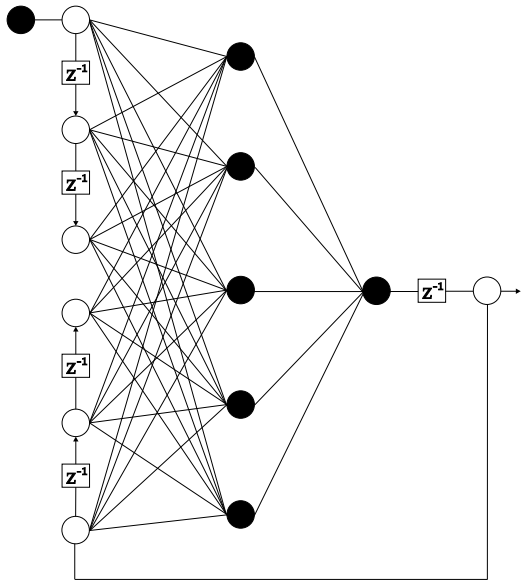

Figure 3.5: A NARX Neural Network. The network has an input order of two, and an output order of three.

power does come at a price though. Recurrent neural networks are notoriously difficult to train, because they struggle to learn long term dependencies [23]. Thus, applying Occam's razor, if a problem is solvable by use of a non-recurrent network architecture, it is better to search for a feedforward network solution first, rather than going straight for a recurrent architecture.

3.6 Network Selection Criteria

We now turn to the question of how to select an appropriate network architecture for the task at hand. As just mentioned, we suggest starting with a simple architecture. When the system to be modelled is known to be chaotic, the methods discussed in Chapter 2 can be applied. The embedding dimension can be used as an indication of the order of the input tap delay line for the TDNN and NARXNN architectures.

The number of layers in the network and the number of hidden neurons in each have always been the subject of much discussion. It has been shown that MLPs with one hidden layer can approximate any continuous function [21]. However, this does not say anything about the ease of

the training process or the generalisation performance. In practise, two different approaches are used: network growing and network pruning. When a network is grown, neurons are added during the training process each time when some criteria is satisfied. With the network pruning technique, a large network is trained until it reaches a good result, and afterwards excess weights are pruned away. This is a form of network regularisation.

3.7 Network Regularisation

Overfitting is a common problem with neural networks and machine learning algorithms in general. It occurs when the network tries to fit training data points exactly. This is problematic, since in order to model data exactly, generalisation performance on unseen test data is often degraded. The best model is one that models the data correctly, and has the minimum number of free parameters. This is known as the minimum description length principle. However, obtaining the correct model may be more difficult when training networks with only the minimum amount of free parameters [34]. A possible solution is to choose the model too large, to remove unimportant weights after training, and then to retrain the model. In this section we discuss a few methods for deciding which weights are important and which are irrelevant.

3.7.1 Optimal Brain Damage

By trading the training error off with network complexity, the best generalisation may be obtained. Thus, by minimising a cost function consisting of the ordinary training error plus a measure of network complexity, generalisation may be improved.

The choice of the complexity measure determines the performance of the regularisation scheme. A crude complexity measure is weight magnitude, but better measures have been proposed. Optimal Brain Damage (OBD) uses a Taylor expansion of the objective function to compute the saliencies of weights [35].

The following pruning algorithm is given:

1. Choose a network architecture

2. Train the network until a "reasonable" solution is obtained

3. Compute the second derivatives h_{kk} for each parameter

4. Compute the saliencies for each parameter: $s_k = h_{kk} w_k^2/2$

5. Sort the parameters according to saliency and delete some low-saliency parameters

6. Iterate to step 2 while network performance is still satisfactory

The pruned network usually needs to be retrained after pruning. Deleting is defined as setting the weight to zero and keeping it zero for all future time. The second derivatives h_{kk} are the diagonal elements of the Hessian matrix H of the objective function E, with respect to the parameter vector W, i.e.

$$h_{kk} = \frac{\partial^2 E}{\partial w_k \partial w_k} \tag{3.9}$$

The values of the h_{kk} can be computed by a procedure similar to error backpropagation, and is described in [35]. The complexity of computing the diagonal Hessian is of the same complexity as computing the gradient.

3.7.2 Optimal Brain Surgeon

The Optimal Brain Surgeon (OBS) improves on OBD by not assuming that the Hessian matrix is diagonal [20]. OBS computes the inverse of the full Hessian matrix. The only approximations it makes is that higher order terms in the Taylor expansion vanish, and that the network response will be close to the desired response for a trained network. A method is provided to update the remaining weights after pruning. This minimises the amount of retraining necessary. The OBS pruning algorithm:

1. Train a "reasonably large" network to minimum error

2. Compute H^{-1}

3. Find the weight index q that gives the smallest saliency $L_q = w_q^2/(2H_{qq}^{-1})$. If this increase

is much smaller than the error E, delete w_q and go to step 4; otherwise go to step 5.

4. *Use q from step 3 to update all weights ($\delta\vec{w} = -w_q H^{-1} \cdot \vec{e}_q / H^{-1}_{qq}$ with $\vec{e}_q$ the unit vector in weight space corresponding to w_q)*
5. *No more weights can be deleted without a significant increase of E.*

The authors of [20] provide a method for computing the inverse of the Hessian matrix. The method has a complexity of $O(Pn^2)$ for computing H^{-1}, where P is the number of examples in the training set, and n is the size of the Hessian matrix. Careful attention must be paid not to include the weights of a disconnected neuron in the calculation of saliencies, as this results in nuisance parameters, which introduce errors. When pruning weights, a neuron becomes disconnected when all its inputs or outputs have been removed.

Many other pruning techniques have been proposed in the literature. Some methods prune whole neurons [63], while others give indications of how much to prune [50]. Other methods, such as partial retraining, try to determine input relevance [68].

3.7.3 Weight Decay

Weight decay is another regularisation method. OBS and OBD regularise the network by pruning weights, and keeping the error small. Weight decay constrains the network by prohibiting weights from growing too large. Weight decay adds an extra term to the cost function which penalises large weights. The new cost function is given by

$$E(w) = E_0(w) + \frac{1}{2}\lambda \Sigma_i w_i^2 \tag{3.10}$$

where E_0 is the normal error measure, and λ is a weight decay parameter. For gradient descent learning, weight decay adds the term $-\lambda w_i$ to the weight update:

$$\Delta w_i \propto -\frac{\partial E_0}{\partial w_i} - \lambda w_i \tag{3.11}$$

Weight decay improves generalisation by suppressing irrelevant components of the weight vector,

and choosing the smallest vector that solves the learning problem [32]. Furthermore, a good choice of λ may suppress some of the effects of static noise in the training set. However, weight decay does add another parameter to the training process, and no general method exists for determining an optimal value for λ.

3.8 Improving the Training Algorithm

Many alterations and improvements on the original error backpropagation algorithm have been proposed. Some of these methods deal with improving the learning capability, training time, and generalisation performance of the network. We discuss three of these methods: improved use of the learning rate, improving the training time, and improving generalisation performance.

3.8.1 Adaptive Learning Rates

The learning rate specifies the amount of change to be applied during weight updates. Adaptive learning rates can enhance network performance in two fundamental ways: The amount of weight update at a particular point on the error surface can be adapted to the shape of the error surface at that point, possibly obtaining faster convergence times. For example, when the error surface is flat, a large learning rate is preferable, but when the gradient descent search is in a ravine, a small learning rate will avoid oscillation, and is preferable. The second reason for using an adaptive learning rate method is that it automates the search for an optimum learning rate, and eliminates the trial-and-error approach.

There are many different schemes for adaptive learning rates. Local learning rate adaption methods have different learning rates for each weight. Global learning rate adaptation methods have a global learning rate for all weights. Some methods have adaptive momentum terms as well. Different principles are used for different methods. These include: numerical optimisation procedures which use second-order information (conjugate gradient, Quasi-Newton, second order calculation

of the step size), stochastic optimisation, and heuristics utilising the sign of the local gradient, the angle between gradient direction, or peak learning rate values. We discuss the delta-bar-delta rule by Jacobs [28], and refer the reader to the literature for a survey of some of the other common techniques employed [45].

The delta-bar-delta adaptive learning rate technique defines a separate learning rate for each weight. It uses an estimation of the slope of the local error function to adjust the learning rate. This estimation is derived using the partial derivative of the error function with respect to the weight at that point, and is obtained on a step to step basis. The local learning rate is increased by a small amount when the partial derivative keeps the same sign. This accelerates learning in shallow regions. When signs change, however, the learning rate is decreased, since the changing signs indicate that a minimum was overshot, and that the previous weight update was too large. In this case the learning rate is exponentially decreased. Each weight maintains the following information:

$$\eta_{ij}^{(t)} = \begin{cases} \kappa + \eta_{ij}^{(t-1)}, & \text{if} \quad \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \beta \cdot \eta_{ij}^{(t-1)}, & \text{if} \quad \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \eta_{ij}^{(t-1)}, & \text{else} \end{cases} \tag{3.12}$$

where $0 < \beta < 1$. The weight update then becomes

$$\Delta w_{ij}(t) = -\eta_{ij}(t)\frac{\partial E}{\partial w_{ij}} + \zeta \cdot \Delta w_{ij}(t-1) \tag{3.13}$$

The initial value η_{ij}, and the parameters κ, β and ζ can be tuned for optimal performance.

3.8.2 Adding Noise to Weight Updates

Noise encountered during the training of a neural network can have a benign effect on network performance. Two types of noise exist: analog and digital noise. Analog noise is impreciseness deliberately built into the training process, whereas digital noise is noise resulting from inaccuracies due to finite bit-length for number storage. It was reported by various authors that analog noise can improve network performance [29] [46]. Noise can be applied in various ways: noisy inputs, noisy weight updates, multiplicative/additive noise and various distribution density functions.

Noise injection can reduce the training time, and at the same time improve generalisation ability. Improved performance can also be obtained by adding noise to the training examples. It was also shown that adding noise to training examples prevents pattern memorisation [43].

Noise has this benign effect on the network because it acts as a regularisation term [29]. The extra noise term changes the error function to favour faster training, and to generate more robust internal representations, resulting in better generalisation performance. Training with noise results in a lower mean weight saliency. What happens in practice, is that weights are forced to be either very small, or very big. This causes the network to be more robust with respect to small variations in the input data. It also has the effect of the weights having a smaller spread of values.

The same results have been obtained for recurrent neural networks. It was shown that training recurrent neural networks with noise improves generalisation performance and convergence simultaneously [29]. The authors also provide performance comparisons between different noise distribution functions, and between applying additive, multiplicative, and cumulative noise.

3.9 Improving Training by Using Prior Knowledge

Many real-world time series are complex, often poorly understood, and non-stationary. It is generally accepted that available expert domain knowledge can improve both training and generalisation performance.

The use of neural networks to obtain rules from data can be divided into two fields: rule extraction and rule discovery [27]. Rule extraction assumes prior theories, and then refines them using training data. Rule discovery depends solely on learning from data, and assumes no prior theory.

Well-known methods for knowledge acquisition are the ID3 and C4.5. decision tree creation methods. These methods can be used to obtain rules for making decisions based on attributes. The rules can then be encoded into a neural network. However, this is not as useful for time series modelling, since it is difficult to assign useful attributes to real valued time series.

Prior knowledge for time series often comes in the form of embedding dimension information, time window sizes or sampling frequencies. This information is extracted from the data by methods such as mutual information and false nearest neighbours. More information on the dynamical process itself may be obtained via methods such as Lyapunov exponent extraction, correlation dimension determination, power spectrum analysis and non-stationarity detection [15]. Prior knowledge about the time series may also be applied to create additional ”virtual” examples such as including examples $f(-x) = -f(x)$ for odd time series.

We now present a method of obtaining rules directly from the time series. We encode these rules into a neural network using the KBANN encoding method.

3.9.1 Knowledge-Based Artificial Neural Networks

Methods for encoding a Boolean rule set into a feedforward network have been proposed [66]. Other methods differ only in the way that they combine the input neurons. The initial network is constructed based on the relationship between rules in the rule set. Rule inputs become input neurons, intermediate results become hidden neurons, final results become output neurons, and dependencies become weighted connections between neurons. All neurons use sigmoidal activation functions.

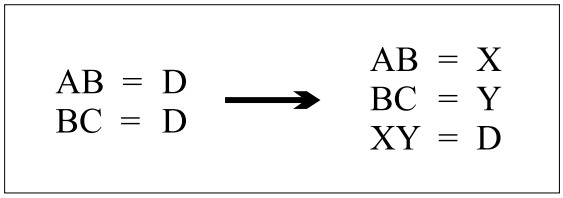

Figure 3.6: An example of a rewritten rule set.

Given a set of Boolean rules, disjunctive rules have to be rewritten in order to avoid accidental activation of rules by wrong combinations. Let two conjunctive rules have the same conclusion, but different inputs. Then, create for each rule a new intermediate result, and use these two intermediate results as input to a new disjunctive rule. The result of the new disjunctive rule is taken as the output

of the original two rules. An example of this procedure is shown in Figure 3.6.

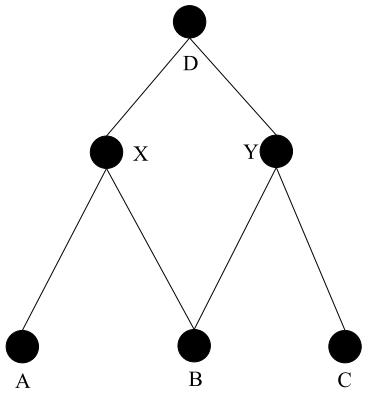

Figure 3.7: Encoding of the Boolean rules shown in Figure 3.6. Uninitialised and bias weights are not shown.

Rules can be encoded into a neural network. Structures for both disjunctive and conjunctive rules are provided. From these two structures, any Boolean rule can be built. A weight in the KBANN network has a value of H when it connects the input of a rule to the output of the same rule. A weight value of $-H$ is assigned if the negative of the input is required. If one uses 1 as the input to the bias weights, the bias weight of disjunctive rules are programmed to $-\frac{H}{2}$, and those of conjunctive rules to $-(P-\frac{1}{2})H$, where P is the number of positive inputs to the rule. These bias weight values guarantee that the output of a disjunctive rule will be high when any of its inputs are high, and that the output of conjunctive rules will only be high when all their inputs are. The encoding of the rules shown in Figure 3.6 is shown in Figure 3.7. The uninitialised and bias weights are not shown.

If the initial domain theory is incomplete, or even not totally correct, the network constructed from the prior knowledge may struggle to perform well. When the prior knowledge is sparse, the network may be too small as well. Adding more learning capability in the form of unprogrammed weights and hidden neurons helps addressing these problems. It has been observed that, in general, networks initialised with correct prior knowledge train faster, and generalise better compared to networks trained without prior knowledge.

3.9.2 KBANN Training

KBANN was designed to encode Boolean rules into a neural network. Since we will be extracting and encoding rules for a real-valued time series, we have to adapt the standard rule encoding algorithm. We encode the extracted rules into a TDNN. For the purposes of this discussion, we will use a one-dimensional time series, but the principle can easily be extended to multidimensional time series.

3.9.3 Rule Extraction from A Time Series

We wish to speed up training of a neural network. We achieve this by encoding prior knowledge into the network. We extract prior knowledge in the form of rules from the time series. We choose rules such that they bias the output of the TDNN high when the correct input sequence is detected. The extracted rules form the first layer of the TDNN. They are then combined by a disjunctive rule at the output. Figure 3.8 shows a network programmed to produce high output values when one of two possible input sequences is detected.

We use a threshold α to determine which data points from the time series will be covered by rules. A rule is generated for every point in the time series for which $x(t) \geq \alpha$. Afterwards, rule duplications are removed, and contradictions are removed. A contradiction occurs when a rule causes the output to be high when the output should have been low. Note that each rule will eventually correspond to a neuron in the hidden layer. Thus, having too many rules will result in an excessively large network. Therefore, we choose α fairly large.

Let a TDNN have n lags in the time delay line. The $x(t)$ denote the data point at which we want the output of the network to be high. Thus, a rule is generated for $x(t)$ using $(x(t-n),$ $x(t-n+1), ..., x(t-1))$. The time series is real valued, but KBANN encodes Boolean rules. Thus, we use another threshold β to divide real values into ones and zeros. A rule is formed by inspecting $(x(t-n), x(t-n+1), ..., x(t-1))$ and requiring each input above β to be one, and each below β to be zero. The choice of β influences the form of the extracted rules. To obtain the

biggest variance in binary values, we suggest using the mean of the time series for β. Note that the size of the rule, i.e. the number of antecedents, is determined by the order of the tap delay line of the TDNN. As an example of rule generation, let $\alpha = 0.9$ and $\beta = 0.4$. Given the time series $(0.3, 0.5, 0.2, 0.3, 0.95, 0.5, 0.2, 0.2, 0.5, 0.45, 0.9)$, two rules will be generated. The input to the rules will be $(0.5, 0.2, 0.3)$ and $(0.2, 0.5, 0.45)$, respectively. Thus, the two rules $\{100\}$ and $\{011\}$ will be generated. These are encoded into a network as shown in Figure 3.8.

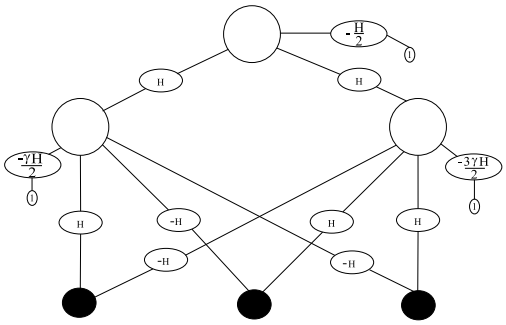

Figure 3.8: The resultant network after the encoding of two rules: $\{100; 011\}$. The output of the network network will only be high when one of the two sequences is detected.

3.9.4 Prior Knowledge Encoding

The extracted rules are encoded as conjunctive rules in the first layer of the TDNN using the normal KBANN method as described in Section 3.9.1. All the conjunctive rules are then combined by one disjunctive rule in the second layer. The conjunctive rule becomes the output.

Since KBANN was designed with binary input values in mind, it expects its inputs to be either one or zero. Real-valued time series that are seldom one will have difficulty activating the extracted rules, since the neuron bias values for the conjunctive rules will be too big to exceed. To compensate, we scale the bias values by a factor γ. Note that the value of γ depends on the choice of β. During experiments, we found the output of the network to be quite sensitive to γ.

Since the network was designed with positive rules in mind, the output of the network prior to training will often be relatively high for most of the data points. To force the output to lower values,

the H value used to program the disjunctive rule can be scaled down by a factor δ. However, in practise we found that error backpropagation quickly adjusts the output to correct values, without the use of a scaling factor.

We designed the network such that the contents of the time delay line have to fit a rule exactly to activate it. Therefore, we cannot randomly initialise weights for which the input should be zero, since the collective effect of many small weights multiplied by large real valued inputs may be enough to activate the rule. This effect is made worse by the downward scaling of the bias, as discussed above. Thus, we set the weight values of zero rule values to $-H$.

It is important to add unprogrammed hidden neurons to the network to allow the network to search for a solution. During experiments, we often found that for larger H values, the rules are kept almost intact, with only small changes to the weights. If too little extra learning capability is available, the network will struggle to train.

3.10 Early Stopping Criteria: Diks' Test

Large training sets do not guarantee good network generalisation performance. Excessive training can result in overfitting, thus causing the generalisation performance to deteriorate. However, it is not clear when to stop training, since more training may in fact yield better results on test data.

One possible stopping criterion is Diks' Test [12]. This test provides a measure of how close the output generated by the model is to the measured data, i.e. how close the generated attractor is to the attractor being modelled. Bakker *et al.* used this test as a stopping criterion for time series prediction training on data generated by a chaotic system [3].

We now give a brief description of Diks' Test. This criterion tests the null hypothesis that two time series originated from the same process, i.e. that the delay vector distributions of two time series are identical. Given two sets of sampled vectors with distributions ρ_1 and ρ_2, Diks *et al.* define two

new distributions ρ_1' and ρ_2' such that

$$\rho_k'(\vec{r}) = \int d\vec{s}\rho_k(\vec{s})\kappa(\vec{r},\vec{s}) \tag{3.14}$$

for $k \in \{1,2\}$ and $\kappa(\vec{r},\vec{s})$ a Gaussian kernel

$$\kappa(\vec{r},\vec{s}) = (\sqrt{2\pi}d)^{-m}e^{-|\vec{r}-\vec{s}|^2/(2d^2)} \tag{3.15}$$

where d is called the bandwidth, as is a fixed length scale. The distance measure Q is then defined as

$$Q = (2d\sqrt{\pi})^m \int d\vec{r}[\rho_1'(\vec{r}) - \rho_2'(\vec{r})]^2 \tag{3.16}$$

For every $d > 0$, $\sqrt{Q}$ defines a distance between the probability distributions ρ_1' and ρ_2' based on the inner product of $(\rho_1' - \rho_2')$ with itself. It follows that $Q = 0$ only for $\rho_1'(\vec{r}) = \rho_2'(\vec{r})$. Thus, by using an estimator $\widehat{Q}$ for Q, and determining whether it is significantly larger than zero, we have a test for the null hypothesis that $\rho_1 = \rho_2$.

Let $\{\vec{X}_i\}_{i=1}^{N_1}$ have distribution ρ_1, and $\{\vec{Y}_i\}_{i=1}^{N_2}$ have distribution ρ_2, then Diks derives the estimator $\widehat{Q}$ as

$$\widehat{Q} = \frac{1}{\binom{N_2}{2}} \sum_{1\le i<j<N_1} \sum h(\vec{X}_i,\vec{X}_j) + \frac{1}{\binom{N_2}{2}} \sum_{1\le i<j<N_2} \sum h(\vec{Y}_i,\vec{Y}_j) - \frac{2}{N_1N_2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} h(\vec{X}_i,\vec{Y}_j) \tag{3.17}$$

with

$$h(\vec{s},\vec{t}) = e^{-|\vec{s}-\vec{t}|^2/(4d^2)} \tag{3.18}$$

Let $N = N_1 + N_2$, then the variance $V_c(\widehat{Q})$ of $\widehat{Q}$ is given as

$$V_c(\widehat{Q}) = \frac{2(N-1)^2)(N-2)}{N_1(N\_1)N_2(N_2-1)(N-3)} \frac{1}{\binom{N_2}{2}} \sum_{1\le i<j<N} \sum \psi_{ij}^2 \tag{3.19}$$

with

$$\psi = H_{ij} - g_i - g_j \tag{3.20}$$

$$H_{ij} = h(\vec{z}_i, \vec{z}_j) - \frac{1}{\binom{N_2}{2}} \sum_{1 \le i < j} \sum_{< N} h(\vec{z}_i, \vec{z}_j) \tag{3.21}$$

$$\vec{z}_i = \begin{cases} \vec{x}_i & \text{for } 1 \le i \le N_1 \\ \vec{y}_{i-N_1} & \text{for } N_1 < i \le N \end{cases} \tag{3.22}$$

$$g_i = \frac{1}{N-2} \sum_{j, j \neq i} H_{ij} \tag{3.23}$$

Finally, the quantity S is defined as

$$S = \frac{\widehat{Q}}{\sqrt{V_c(\widehat{Q})}} \tag{3.24}$$

and has a zero mean and unit variance under the null hypothesis.

For more detail on choosing the optimum values for the bandwidth d, as well as how to deal with correlations within the time series, we refer the reader to the paper by Diks [12]. Diks suggests rejecting the null hypothesis with 95% confidence for S larger than three. In an adaptation of Diks' procedure, Bakker *et al.* used an S averaged over 10 semi-independent realizations. For this case, they found that a value of 3 caused the test to classify two systems as identical too soon. They used a value of 1.4 for S for their application [3].

3.11 Benchmark Problems

We test some of the methods discussed in this chapter on our benchmark chaotic systems.

3.11.1 Network Pruning Demonstrated on the Lorenz Time Series

In Section 2.5.1, we found the embedding dimension of the Lorenz time series to be 3. We trained a TDNN with 3 layers and a tap delay line of order 3. We used 30 sigmoidal hidden neurons, and

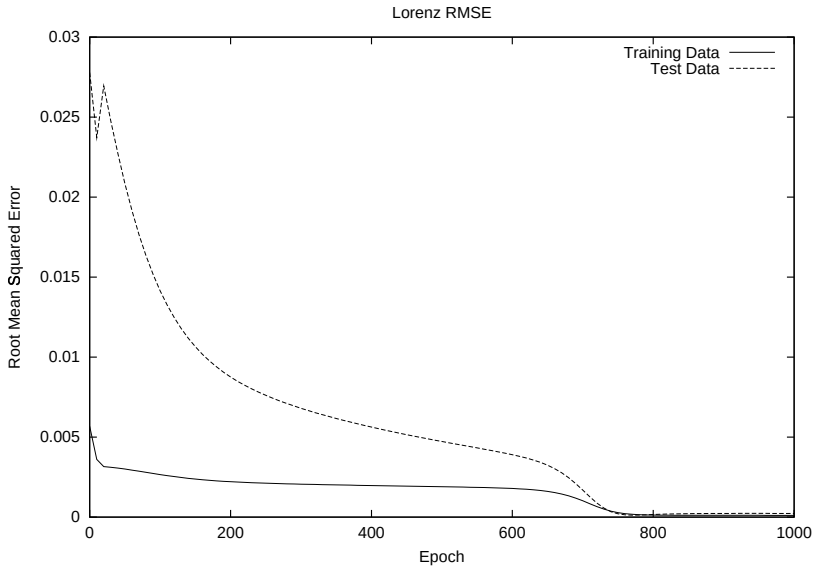

Figure 3.9: The mean squared error of the training and test data for the TDNN trained on the Lorenz time series.

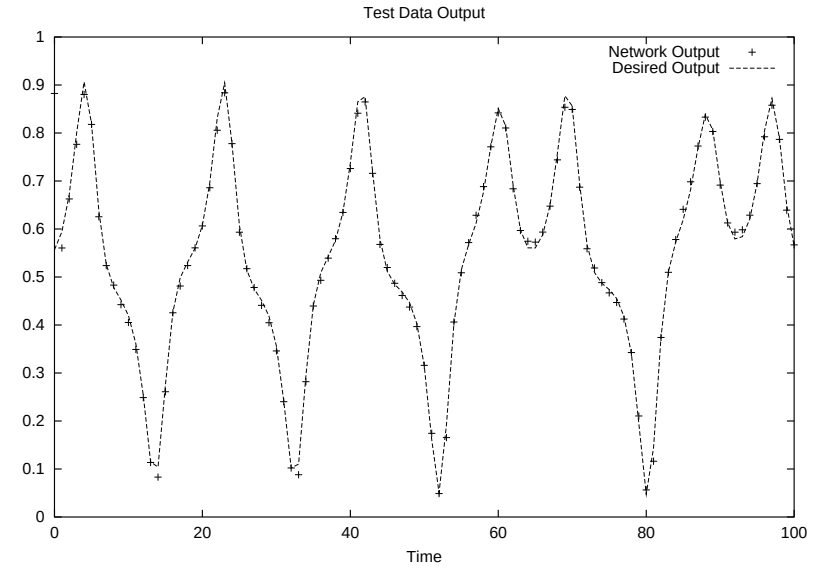

Figure 3.10: TDNN prediction of test data.

a linear output neuron. Thus, including the bias weights, the network consisted of 120 weights in the first layer, and 31 weights in the second layer. Figure 3.9 shows the mean squared error of the training and test data as a function of training time. The network's prediction performance is shown

in Figure 3.10. The network learned the training set, and performed very well on the test set.

We applied OBS to prune 120 weights off the trained network, leaving a network with 31 weights, and 8 remaining hidden neurons. The pruned network had a MSE of 0.3, which is 3 orders of magnitude worse than that of the trained network. However, after retraining the pruned network for 10 more epochs, a MSE of 0.0002 (the same as the un-pruned network) was again obtained, but this time for a network with 31 weights.

3.11.2 The Influence of Noisy Training Data

We use the Rössler time series to demonstrate training with noisy data. The embedding dimension for the Rössler time series was determined to be 4. We again use a TDNN with 30 sigmoidal hidden neurons, and a tap delay line of order 4. The performance of a network trained on noiseless data is shown in Figure 3.11. We also show results for a one-step-ahead, and a five-step-ahead prediction.

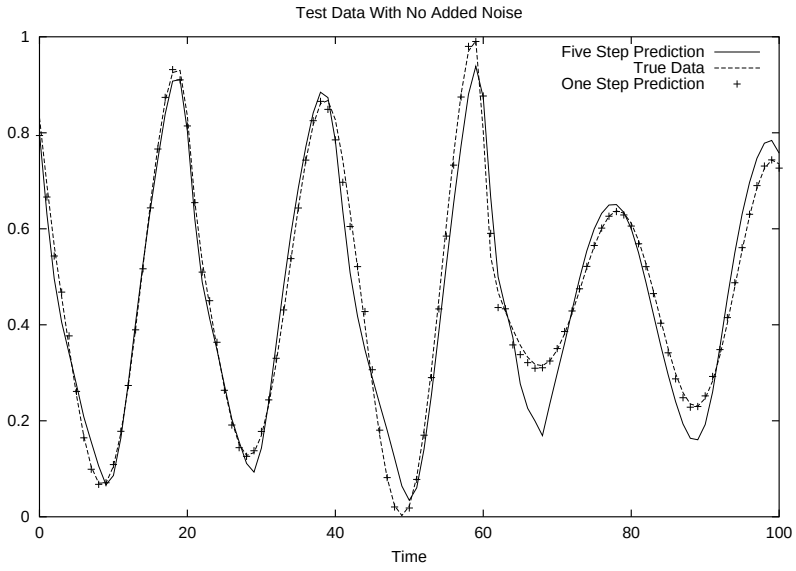

Figure 3.11: Prediction of test data from the Rössler time series. The training data contained no noise.

We added uniformly distributed noise to the training set. Prior to training, all the networks were initialised with the same random weights. Figure 3.12 shows the deterioration of prediction performance when noise is added to the training data. For the noiseless data, we have near perfect

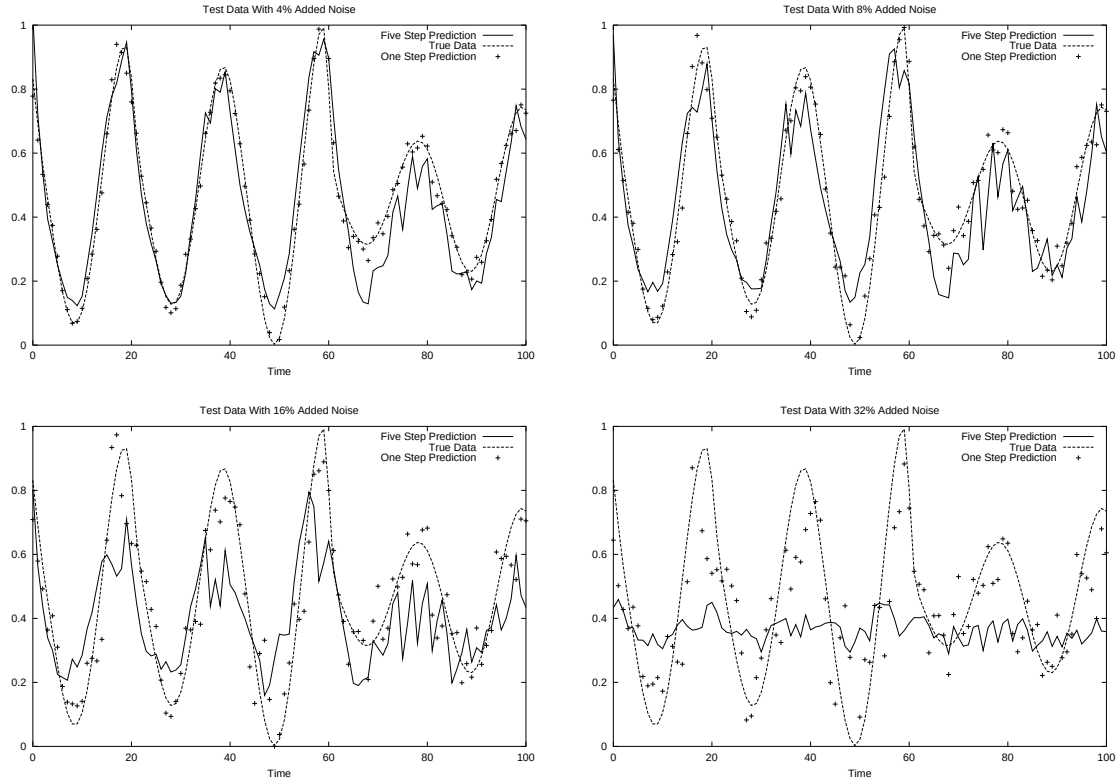

Figure 3.12: Prediction of the Rössler time series when 4%, 8%, 16%, and 32% noise is added to training examples.

one-step prediction, and good five-step prediction. At 16% noise the five step prediction becomes unreliable and at 32% completely unreliable. The one-step prediction becomes unreliable at 32% noise. It is clear, however, that even under quite noisy conditions the networks were still able to extract reasonable models for the system.

3.11.3 Adaptive Learning Rates Demonstrated on the Mackey-Glass Time Series

We use the Mackey-Glass time series to illustrate the beneficial effect that an adaptive learning rate can have on network training and generalisation performance. Using the method of false strands, we determined the embedding dimension for the Mackey-Glass time series to be 6. Therefore we

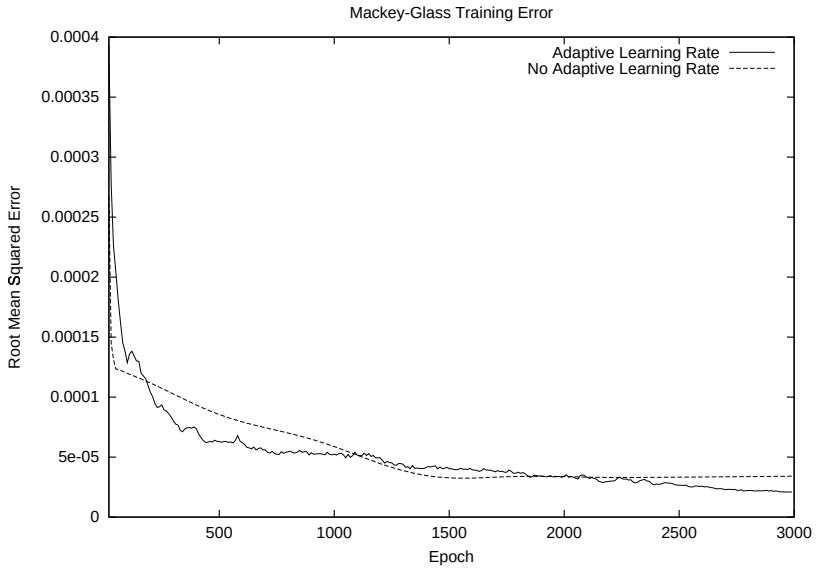

Figure 3.13: Comparison of the MSE on the training data when trained with, and without, adaptive learning rates.

set the input order of the tap delay line to 6, and again used 30 hidden neurons for a TDNN. We used the first 1000 data points of the time series for training, and the next 200 data points for testing. We trained another network with the same architecture and the same random initialised weights, but applied the delta-bar-delta adaptive learning technique, as discussed in Section 3.8.1.

The MSE on the training set for both methods Is shown in Figure 3.13. The adaptive learning rate method and fixed learning rate method performs equally well on the training data. However, when we inspect Figure 3.14, we observe that the generalisation performance of the adaptive learning rate method is significantly better than that of the fixed learning rate method. The comparison of the two methods after 300 epochs of training is shown in Figure 3.15. Unfortunately, this result is not universal, as we have found instances where adaptive learning rates had a detrimental effect on either or both training and generalisation performance.

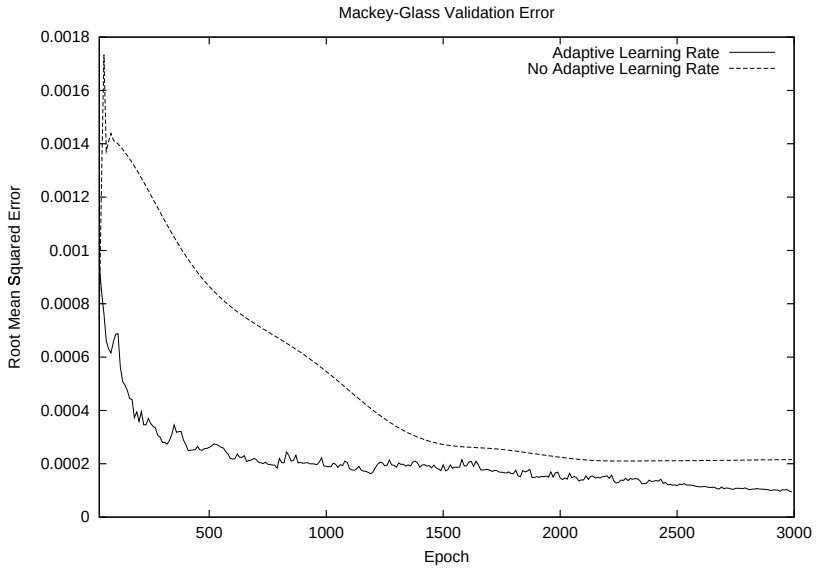

Figure 3.14: Comparison of the MSE on the test data when training with, and without, adaptive learning rates.

3.11.4 Faster Training with Prior Knowledge

We use the Lorenz time series to demonstrate how prior knowledge can speed up training. Although an input order of three was used previously, we choose the input order to be five for the TDNN. An input order of three is too small to extract rules, as it provides only eights possible rules, resulting in too many contradictions. Thus, all the conjunctive rules will have five inputs. We chose $\alpha = 0.9$ and $\beta = 0.5$. We found empirically that $\gamma = 0.3$ yields a good result. We chose not to scale the output, thus we have $\delta = 1$. We extracted 48 rules, five of which remained after validation. The five extracted rules for which the output is to be high are: $\{00111; 01111; 00001; 00011; 11111\}$, where zeros are programmed with $-H$, and ones with H. As discussed in Section 3.9.4, the five conjunctive rules are used as input to a disjunctive rule in the output layer. We chose the value of H as 2.0.

The training time for the initialised network is significantly less than that of an uninitialised network — often half the training time. The network constructed from the extracted rules was augmented with 15 additional hidden neurons. The training times for ten uninitialised networks,

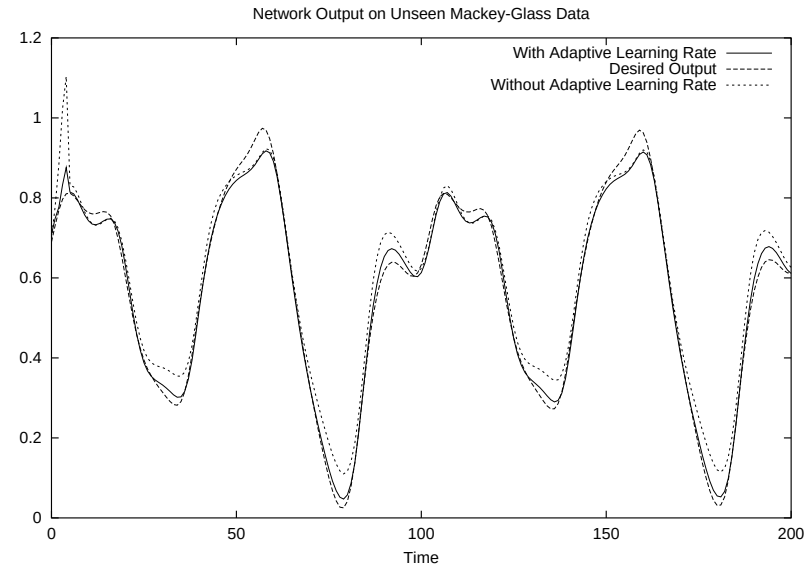

Figure 3.15: Comparison of the TDNN prediction of test data from the Mackey-Glass time series when training with, and without, adaptive learning rates.

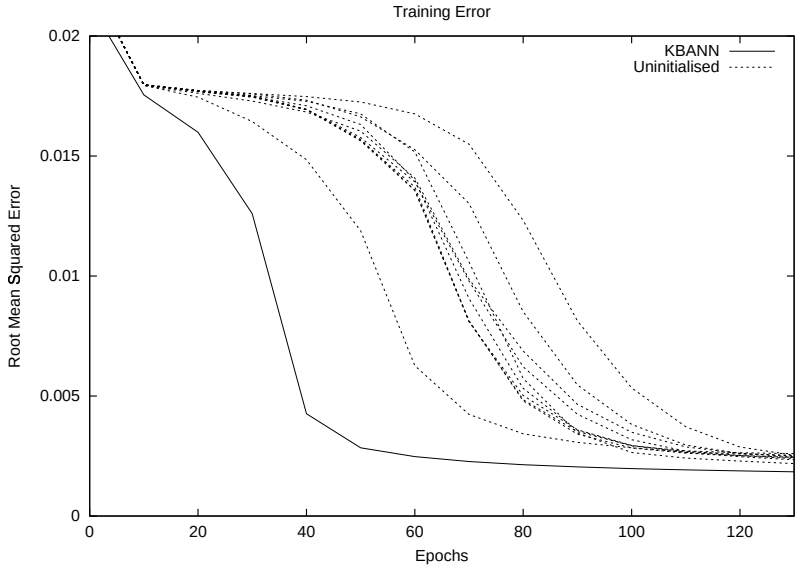

Figure 3.16: Comparison between the MSE performance as a function of training time for an network initialised with prior knowledge, and ten uninitialised networks. The results are shown for training data from the Lorenz time series.

each with five time lags and 20 hidden neurons, are compared to the training time of the constructed network. The results for one-step prediction of training and test data are shown in Figures 3.16 and 3.17, respectively.

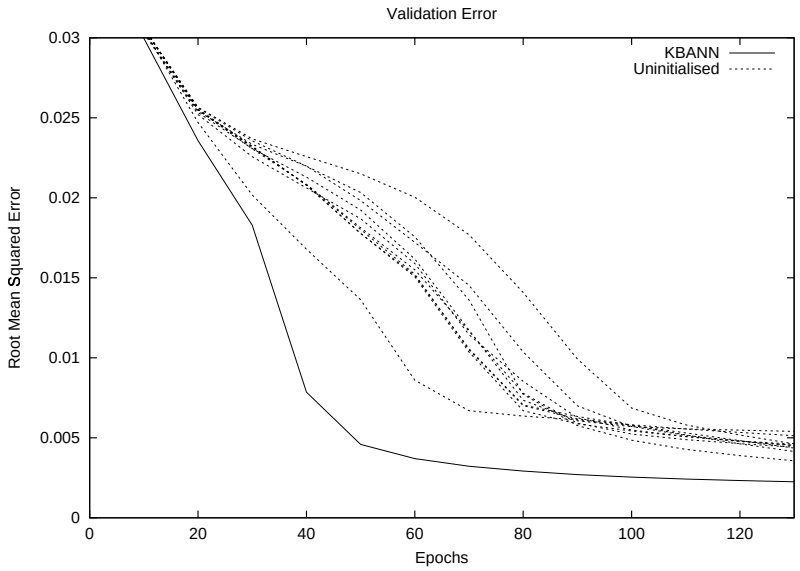

Figure 3.17: Comparison between the MSE performance as a function of training time for an network initialised with prior knowledge, and ten uninitialised networks. The results are shown for test data from the Lorenz time series.

3.11.5 Modelling the Laser Data with FIRNN

We demonstrate the effectiveness of the FIRNN architecture by reproducing the results of the Santa Fé time series competition where this architecture outperformed all other methods of the predicting the behaviour of a chaotic laser [69]. For more detail on the time series itself, please see Section 2.5.2.

We repeated the experiment that won the competition. We trained a FIRNN with two hidden layers, each consisting of twelve neurons with tap delay lines of order five, and an input layer tap delay line of order 25 on the first 1000 data points. We used the temporal error backpropagation method as described in Appendix A.2. We had difficulty obtaining good generalisation performance. How-

ever, when we added Gaussian noise with zero mean and 0.01 variance to the weight updates, we obtained better generalisation performance. This result supports the claims made in Section 3.8.2 that noisy weight updates can improve training and generalisation performance. The results of one-step-ahead prediction for training and test data are shown in Figures 3.18 and 3.19, respectively. The graphs show that good one-step-ahead prediction performance can be obtained with a FIRNN.

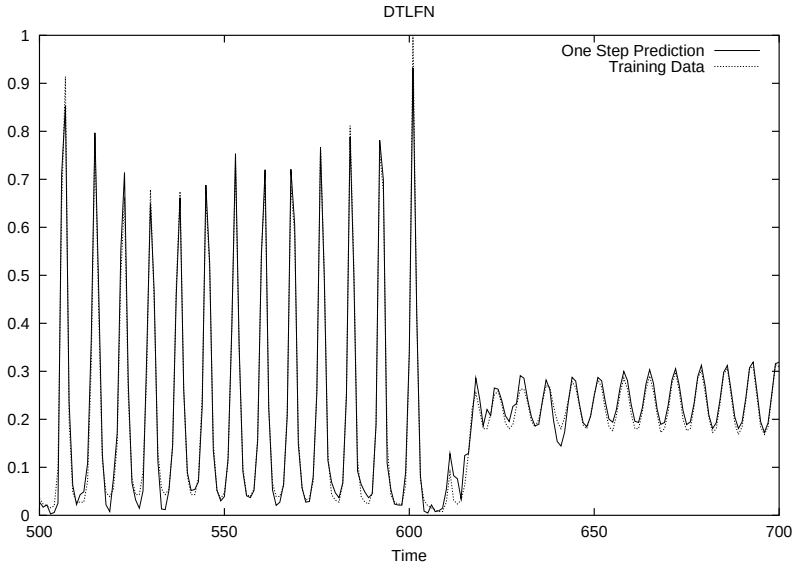

Figure 3.18: One-step prediction of laser training data with a FIRNN.

Closed loop prediction is used when prediction further into the future is required. A cumulative error builds up during closed loop prediction. This error makes long term prediction of chaotic systems impossible. However, the graphs shown in Figure 3.20 show that the FIRNN performed well when predicting the next 200 (data points 1000 - 1200) data points in closed loop.

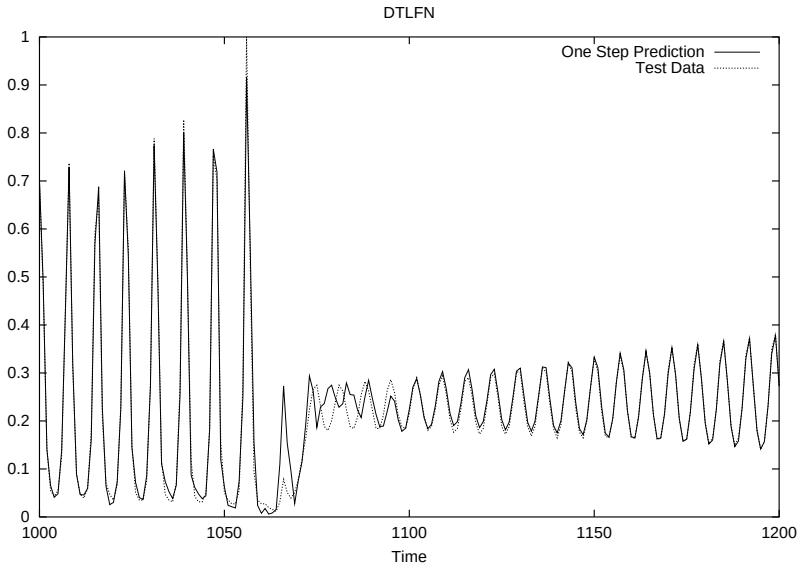

Figure 3.19: One-step prediction of laser test data with a FIRNN.

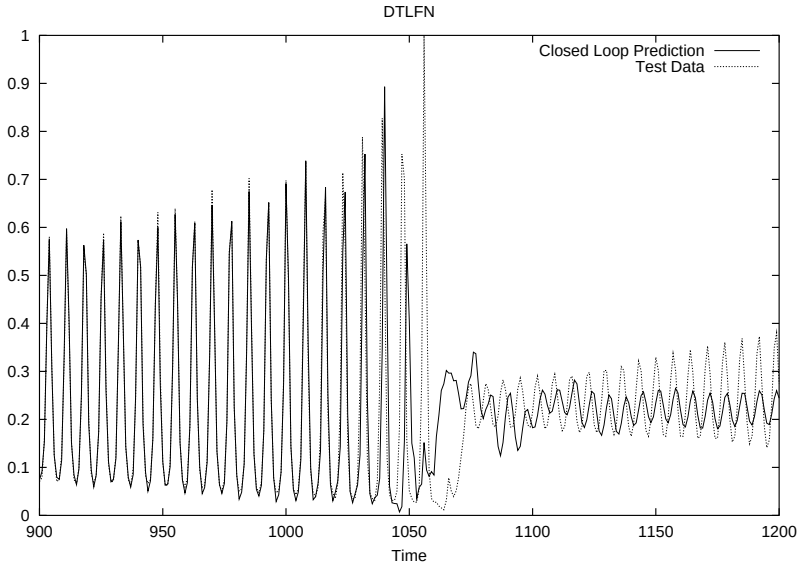

Figure 3.20: Closed loop prediction of laser test data using a FIRNN.

3.12 Summary

In this chapter, we discussed the application of neural networks to the problem of time series prediction. Neural networks are capable of modelling deterministic, non-chaotic systems with relative ease. However, there exist many mathematical and statistical tools that perform as well, or even better (e.g. when the differential equations are known). Neural networks really show their strength in chaotic applications. When applied to experimental time series, of which the dynamics are uncertain or unknown, they have little equal. We discussed various neural network architectures, and various techniques to improve their performance. We demonstrated the techniques on the chaotic time series that we analysed in the previous chapter. However, even the most complex of these time series pale in comparison to the time series we discussed next : seismic event time series.

Chapter 4

Seismic Monitoring and Prediction

4.1 Introduction

Geophysics is concerned with the physical properties of the earth, and therefore also with the study and modelling of the flow of rock. Seismic monitoring in mines was introduced more than 30 years ago. The two primary reasons for monitoring seismic activity were to locate large seismic events to aid in rescue operations, and to predict large rock mass instabilities. It has long been only a dream to understand the physical laws governing rock flow to such a degree that adequate models can be developed to predict rock bursts. However, several factors hindered the achievement of this ambitious objective. Until recently, seismic monitoring systems were based on analog technology, which are noisy and often provide poorly calibrated data. Furthermore, seismic events were, at best, described by their local magnitude. These factors resulted in a non-quantitative analysis of seismic activity. Seismic analysis was based on statistical methods, with only limited physical interpretation.

Recent advances in seismic monitoring technology have improved the accuracy of seismic source parameter measurements in hard rock mines [42]. This presents an opportunity for modelling the rock dynamics, thus predicting seismic events. The process of monitoring seismic events compro-

mises the five stages shown in Figure 4.1: (1) continuously monitoring sensors to decide when the signal becomes significant (triggering); (2) determination whether the signal represents a seismic event (validation); (3) assignment of sensor recording to events (association); (4) extraction of source parameters (seismic processing); (5) construction of a model from historical data (interpretation). In this chapter, we discuss the methods and models used by professional geophysicists to extract and model seismic events. We use the excellent book by Mendecki *et al.* as our primary source of information regarding this subject [42]. We will discuss our methods for modelling the data in Chapter 5.

4.2 The Five Stages of Information Acquisition

We already mentioned the five stages through which seismic data passes in Section 4.1: triggering, validation, association, seismological processing, and interpretation. Figure 4.1 gives a pictorial representation of this process. We will now briefly discuss these stages.

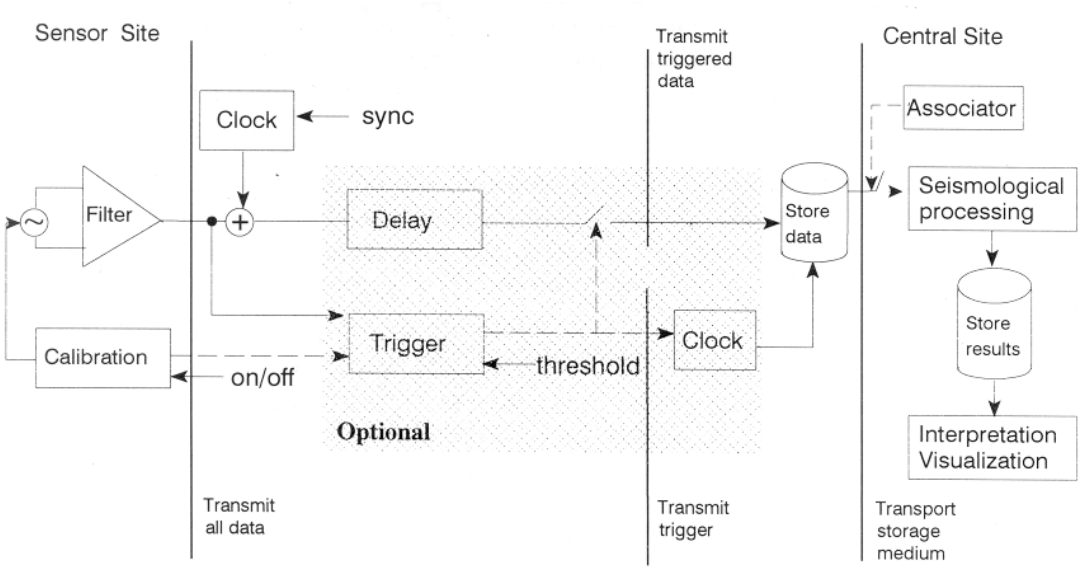

Figure 4.1: The stages of seismic monitoring.

4.2.1 Triggering

Triggering is the automatic detection of a seismic signal in real-time which initiates further action. Ground motion is sampled at a high resolution. Thus, continuous recording results in huge amounts of data. Triggering is used to filter events from non-events and to initiate the process of generating a report.

The fixed threshold method is the simplest triggering method. The signal is compared to a fixed threshold; an event is declared if the signal exceeds the threshold. This method does not work well in the presence of noise. An improved method compares the long-term average (LTA) of the signal to the short-term average (STA). An event is declared when this ratio exceeds the triggering ratio (R_t):

$$\frac{STA}{LTA} > R_t \tag{4.1}$$

Since an event is only detected after it has begun, the most recent history is stored in short-term memory. This can also be used to determine when an event ends.

4.2.2 Validation

A validation stage determines the validity and suitability of a trigger event for further processing. Neural networks have been used for validation and obtained a success rate of 90%, which is comparable with that of trained operators [9]. These networks consisted of 200 input neurons, 10 hidden neurons, and 2 output neurons. They used the squared amplitudes of the recorded signal as input, and trained it on 500 data sets which had been visually classified as good or bad.

4.2.3 Association

The associator identifies data originating from the same event. This decision is based on the time difference of triggering between two stations. The time difference Δ_{ij} must be less or equal to the

time that a seismic wave takes to travel between the two stations:

$$|T_i - T_j| \leq \Delta t_{ij} = \frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}{V_p} \tag{4.2}$$

where T_i is the trigger time of the i^{th} station at position (x_i, y_i, z_i), and V_p is the wave propagation velocity. Equation 4.2 can be used to associate pairs of triggers; further rules are used to find all the triggers pertaining to an event. One strategy, sometimes referred to as a "network trigger", searches for the start trigger. The first station that recorded the event is set as the start trigger. A time window from the trigger time of the start trigger, during which all associated triggers must arrive, is used to validate the start trigger. If an insufficient amount of stations are associated with the event, the trigger is discarded, and the next station is tested for being the start trigger. The length of the time window is usually equal to the longest wave travel time between any two stations in the network.

4.2.4 Seismic Processing and Interpretation

All the processing of acquired data is done at a central site. This site must perform many tasks in parallel. The data acquisition part of the central processing site is vital for maintaining continuous monitoring, and is the most developed, and best understood part of the whole system. The interpretation of the data is done by human operators according to the models that we will discuss in Section 4.7.

4.3 The Seismic Event

Mining in hard rock can induce two kinds of deformations: elastic reversible and inelastic non-reversible deformations. Elastic deformation causes no new micro defects within the rock; all existing micro defects convect with the mass, without growing in size. Inelastic deformation is mainly caused by fracturing and frictional sliding. The potential energy that accumulates during elastic deformation may be unloaded either gradually, or be released suddenly during the process of inelastic deformation.

During inelastic deformation, fracturing and frictional sliding of the rock radiate seismic waves. The strength, state of stress, size, and rate of deformation of the rock determine the frequency and magnitude of the radiated seismic waves. With all other parameters held constant, the following general rules apply:

1. The amplitude and frequency increase with an increase in rock strength and stress,
2. the frequency at which the most energy is radiated, called the predominant frequency, decreases with an increasing source size, and
3. the amplitude and frequency increase with increasing deformation rate.

A seismic event can now be defined as a sudden inelastic deformation within a given volume of rock, such that detectable seismic waves are radiated. The following source parameters characterise a seismic event: time of the event t, location $X = (x, y, z)$ of the event, seismic moment M, seismic energy E radiated from the source of the event , and size and stress estimates.

Seismic events can range from the fracturing of a cubic meter of rock, with cracks in the order of a meter, to the fracturing of a cubic kilometre of rock, with source dimensions of a few hundred meters. The average velocity of rock deformation varies between a few centimetres per second to a few meters per second. In general, the velocity is lower within softer, less homogeneous rock that has a lower differential stress.

4.4 A Simple View of Rock Dynamics

4.4.1 Seismic Moment

The distribution of the forces, or moments, at the seismic source V provides the most general description of the processes within the rock. The inelastic processes at the source can be described as the stress-free change of size and shape of an elastic body without alteration of the elastic

properties of the region [2]. The change in strain $\Delta\epsilon_{kl}$ provides a measure for the change in size and shape. The equivalent stress change, or change in moment per unit volume, can be given by:

$$\Delta\sigma_{ij} = c_{ijkl}\Delta\epsilon_{kl} \tag{4.3}$$

with c_{ijkl} elastic constants. $\Delta\sigma_{ij}$ is called the seismic moment density tensor, or stress glut. The stress glut is the internal stress necessary to cancel the strain produced by the internal inelastic process. Integration of the total moment over the source volume defines the seismic moment tensor M_{ij},

$$M_{ij} = \int_V \Delta\sigma_{ij} dV \tag{4.4}$$

We define the scalar seismic moment M for the region with the largest inelastic shear strain drop $\Delta\epsilon$. In this region the strain drop is of the order of the stress drop. Note that the stress drop is not limited to the source volume, whereas the stress glut is. Using this region as the source volume V, we can express M as

$$M = \mu\Delta\epsilon V = \Delta\sigma V \tag{4.5}$$

where μ is a rigidity constant. The seismic moment is often expressed in terms of magnitude:

$$m_M = \frac{2}{3}\log M - 6.1 \tag{4.6}$$

The seismic moment is proportional to the integral of the far field displacement pulse of the seismic radiation. This can be used to determine seismic moment from body wave observations. Unfortunately, heterogeneities within the rock cause more complex radiated pulse shapes. Unbroken, homogeneous rock causes much less wave attenuation whereas heterogeneous structures can lead to an underestimation of event sizes.

4.4.2 Radiated Seismic Energy

Radiated seismic energy is another important seismic event source parameter. Fracturing and frictional sliding results in the transformation of elastic strain into inelastic strain, and causes energy to

be released. The speed, or average velocity, with which the transformation takes place determines the frequency of the radiated seismic waves. For slower movement, the predominant frequency tends to be lower than for faster movement. Furthermore, slower ruptures tend to radiate less energy, whereas quasi-static ruptures radiate almost no energy.

The energy radiated from a single source fracture is given in terms of the source parameters:

$$E = -2\gamma_{eff}A + \frac{1}{2}\int_A \Delta\sigma_{ij}u_i n_j dA + \int_0^{t_{cs}} dt \int_{A(t)} \dot{\sigma}_{ij}u_i n_j dA \tag{4.7}$$

where γ_{eff} is the effective surface energy which includes the total loss of mechanical energy, A is the area of the fracture with displacement u_i, $\Delta\sigma_{ij}$ is the difference between the final and initial stress, n_j is the unit vector normal to the fracture plane, t_{cs} is the source duration, and $\dot{\sigma}_{ij}$ is the traction rate. The relative contribution of the fracture work to seismic energy increases with decreasing fracture size. For small fractures, the first term of Equation 4.7 cancels out the second, suppressing the acoustic emission, thus giving rise to a ”silent” fracture. The second term of Equation 4.7 contains the static quantities stress drop and final slip. The final term with the traction rate depends on the fracture propagation, and correlates with slip. Due to the presence of the time derivative of stress, faster stress oscillations contribute more to the radiated energy. Radiational friction is the high frequency waves radiated by the acceleration and deceleration of rock during a rupture. Radiational friction normally occurs when the moving zone of slip pule reaches regions of differing resistance to deformation. The third term of Equation 4.7 will vanish when the traction rate and slip are uncorrelated.

In practice, the strength and distribution of barriers together with asperities cause ruptures to propagate in a discontinuous matter. However, the contribution to the total radiated energy from such incoherent ruptures is unknown. Equation 4.7 illustrates that it is not possible to obtain an unambiguous expression for radiated energy only in terms of the initial and final equilibrium states of the rock.

4.5 Transducers

Seismic waves are measured by transducers placed throughout the volume of interest. A transducer is a device used to measure and convert ground motion to an electric signal. This analog electric signal can then be sampled by an analog to digital converter, and fed to a central processing system.

The amplitude and frequency range of the seismic waves to be measured determines the sensitivity required of the transducer. Typically, the largest events measured range from magnitude $m_M = 3$ to $m_M = 5$, whereas the smallest events, measured during quiet periods, range from $m_M = -3$ to $m_M = -4$. The frequencies at which measures must be made is dependent on the corner (predominant) frequency, which in turn depends on the rock mass to be monitored.

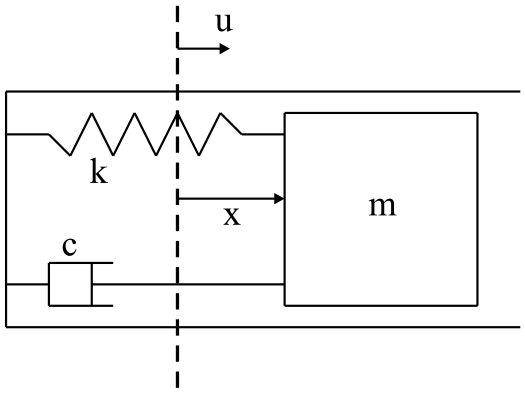

Figure 4.2: An inertial damped spring-mass system. This system is a simplistic transducer, used to measure ground motion.

Most seismic transducers measure ground motion relative to an inertial mass. The mass is balanced by a damped spring system, as shown in Figure 4.2. Newton's laws for such a system prescribes:

$$m(\ddot{u} + \ddot{x}) = -kx - c\dot{x} \tag{4.8}$$

where m is the inertial mass, u is the ground displacement to be measured, x is the displacement of the mass relative to the ground and casing, k is the spring constant, and c is the damping coefficient. The following transfer function is derived by applying the Laplace transform to the system of

Equation 4.8,

$$\frac{\bar{x}(s)}{\bar{u}(s)} = \frac{-s^2}{s^2 + 2b\omega_n s + \omega_n^2} \tag{4.9}$$

where $\bar{x}(s)$ and $\bar{u}(s)$ are the Laplace transforms of $x(t)$ and $u(t)$ respectively, s is the complex frequency variable, $\omega_n = 2\pi f_n$ is the natural frequency ($\omega_n^2 = k/m$), and b is the relative damping factor ($2b\omega_n = c/m$). The bode plots for such a system are shown in Figure 4.3. The graphs show that the natural frequency and damping factor play a central role in the impulse response. A lightly damped sensor ($b << 1$) will have an exaggerated response at the natural frequency, and is thus undesirable. A damping value of $b = 0.7$ will give the flattest frequency response.

When $s >> \omega_n$, the transfer function is essentially equal to -1, and the mass mirrors the ground motion. This is the classic seismometer mode. On the other hand, when $s << \omega_n$, the relative motion of the mass to the ground and casing is small, and proportional to s^2. As this represents a double time differentiation, the mass moves proportional to the ground acceleration. Therefore this mode is used in accelerometers.

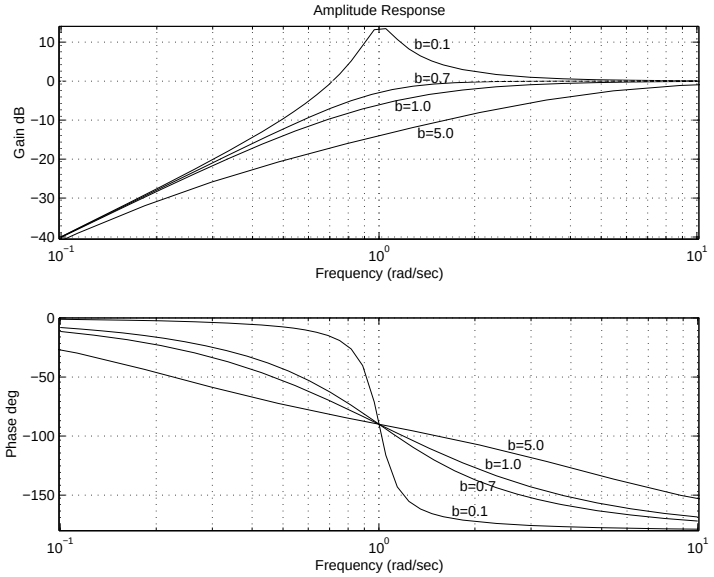

Figure 4.3: The normalised frequency response for the system depicted in Figure 4.2. Note that for under-damped systems, the gain is positive. The response is the flattest for $b = 0.7$, called critical damping.

4.6 Seismic Parameters Used for Event Prediction

The following relations between source parameters are used by the event prediction models that we discuss in Section 4.7.

4.6.1 Energy Index

The energy index EI is defined as the ratio of the observed radiated seismic energy of an event to the average energy radiated by events of the same seismic moment $\bar{E}(M)$, taken from the $\log E$ - $\log M$ relation for the area of interest. It is given by the relation:

$$EI = \frac{E}{\bar{E}(M)}, \quad \bar{E}(M) = 10^{c_3 \log M + c_4} \tag{4.10}$$

where c_3 and c_4 are constants for the given area.

4.6.2 Apparent Volume

The source volume can be estimated from the relation $V = \frac{M}{\Delta\sigma}$. The apparent volume is the measure of the volume of rock with coseismic strain, and is given by the relation:

$$V_A = \frac{M}{2\sigma_A} = \frac{M^2}{2\mu E} \tag{4.11}$$

Apparent volume depends on seismic moment, and can easily be expressed in the form of cumulative plots. It provides insight into the rate and distribution of coseismic deformation and stress transfer in the rock mass. Apparent volume has been shown to show precursory behaviour prior to instabilities in the form of a considerable increase in the cumulative volume prior to the event.

4.6.3 Seismic Viscosity

Seismic viscosity is the resistance of a rock mass to the flow coseismic inelastic deformation within a volume ΔV over time Δt. Seismic viscosity is given by the ratio of seismic stress to seismic

strain:

$$\eta_s(\Delta V, \Delta t) = \frac{\sigma_s}{\dot{\epsilon}_s} = \frac{4\mu^2 \Delta V \Delta t \sum_{t_1}^{t_2} E}{(\sum_{t_1}^{t_2} M_{ij})^2} \tag{4.12}$$

4.6.4 Relaxation Time

The period for which past data is useful for the prediction of future events is referred to as the relaxation time τ_s. It is dependent on the seismic viscosity:

$$\tau_s(\Delta V, \Delta t) = \frac{\eta_s}{\mu} \tag{4.13}$$

4.6.5 Deborah Number

The Deborah number is given by the ratio of the relaxation time to the time of observation Δt:

$$De_s(\Delta V, \Delta t) = \frac{\tau_s}{\Delta t} = \frac{4\mu \Delta V \sum_{t_1}^{t_2} E}{(\sum_{t_1}^{t_2} M_{ij})^2} \tag{4.14}$$

The Deborah number can be interpreted as the ratio of elastic to viscous forces, with De_s going to infinity for perfectly elastic mediums. For $De_s < 1$, the viscous forces dominates, whereas for large De_s, the system will essentially behave as an elastic solid.

4.6.6 Seismic Diffusion

Average seismic diffusion is given by the following relation:

$$D_s(\Delta V, \Delta t) = \frac{L^2}{\tau_s} = \frac{L^2 \mu}{\eta_s} = \frac{(\sum_{t_1}^{t_2} M_{ij})^2}{4\mu \Delta V \sum_{t_1}^{t_2} E} \tag{4.15}$$

for a cube of rock with volume L^3. Seismic Diffusion increases when same size events become softer, and decreases with an increase in the viscosity of the seismic flow of the rock. It is a function of space and time.

4.6.7 Seismic Schmidt Number

The Schmidt number gives the ratio of kinematic viscosity ν_s to diffusion. It measures the turbulence in a given flow, higher numbers indicating less turbulence. The seismic Schmidt number is given by:

$$Sc_{sd}(\Delta V, \Delta t) = \frac{\nu_s}{\Delta t} = \frac{4\mu^2 \Delta V \Delta t(\bar{t}) \sum_{t_1}^{t_2} E}{\rho(\bar{X})^2 (\sum_{t_1}^{t_2} M_{ij})^2} \tag{4.16}$$

where $\bar{t}$ is the average time between consecutive events, and ρ is the rock density dependent on $\bar{X}$, the average distance between consecutive sources of interacting seismic events. The seismic Schmidt number gives the spatiotemporal complexity of the seismic flow of rock, with lower numbers indicating less stable flow.

4.6.8 Seismic Softening

The seismic softening value S_s gives an indication of the average seismic softening or hardening within a volume ΔV between t_0 and t_2. The seismic softening is given by:

$$S_s(\Delta V, \Delta t_1, \Delta t_2) = \frac{(\sum_{t_1}^{t_2} E \sum_{t_0}^{t_2} M_{ij} - \sum_{t_0}^{t_1} E \sum_{t_1}^{t_2} M_{ij}) \sum_{t_0}^{t_2} M_{ij}}{\sum_{t_1}^{t_2} M_{ij} \sum_{t_0}^{t_1} M_{ij}} \tag{4.17}$$

where $\Delta t_1 = t_1 - t_0$ and $\Delta t_2 = t_2 - t_0$.

4.6.9 Time to Failure

The time to failure t_f is estimated by utilising the behaviour of materials under constant stress and temperature:

$$\dot{\Omega}^{-\gamma} \ddot{\Omega} = c \tag{4.18}$$

where γ and c are constants, and Ω is the measured strain. Equation 4.18 is used for estimating the time to failure:

$$\frac{\Sigma V_A}{\bar{t}\bar{E}I\Delta V} = \frac{k}{(t_f - t)^\alpha} \tag{4.19}$$

where $\bar{t}$ and $\overline{EI}$ are running means, and k and α are constants.

Figure 4.4 shows graphs[1] leading to an event of size $\log E = 9.43$.

4.7 Current Seismic Event Prediction Methods

We qualitatively discuss the models used by current event prediction systems for predicting the danger of continuing mining operations. The growth of the deformation process up to the point of instability is called nucleation. The breakdown instability cannot take place before quasi-static or quasi-dynamic inelastic deformation has occurred within the rock. Experimental data shows that the nucleation process includes substantial crack growth, as well as dynamic instabilities of local to small scale. Sub-critical crack growth starts only above certain stress levels, which can be in the order of 50% of the rupture stress. During this process, crack-advance occurs by discrete, individually dynamic events, but the average rupture velocity is of small scale. The individual micro-seismic events can be modelled as a continuous, quasi-static process, because they release only small amounts of stress or seismic moment compared to the breakdown instability. Larger events have greater influence on the stress and strain environment in the area, and are more likely to influence subsequent events.

The development of the nucleation zone is associated with strain softening. Thus, sources of seismic events located within the nucleation zone are on average of a lower, softer nature. Experiments have shown that crack branching, and associated distributed damage, are observed only when the strength variation in the source region is greater than the stress concentration. Outside and at the interface of the nucleation zone, seismic events of a harder nature characterised by higher energy indices can be expected. As the size of the seismic moment of the potential instability increases, the overall nucleation zone also increases in size.

A phase of accelerating deformation precedes breakdown instability. This increase in the rate of co-seismic deformation prior to instability can be attributed to the increased rate of micro-seismicity,

[1] Figures from [42] reproduced with the kind permission from Kluwer Academic Publishers

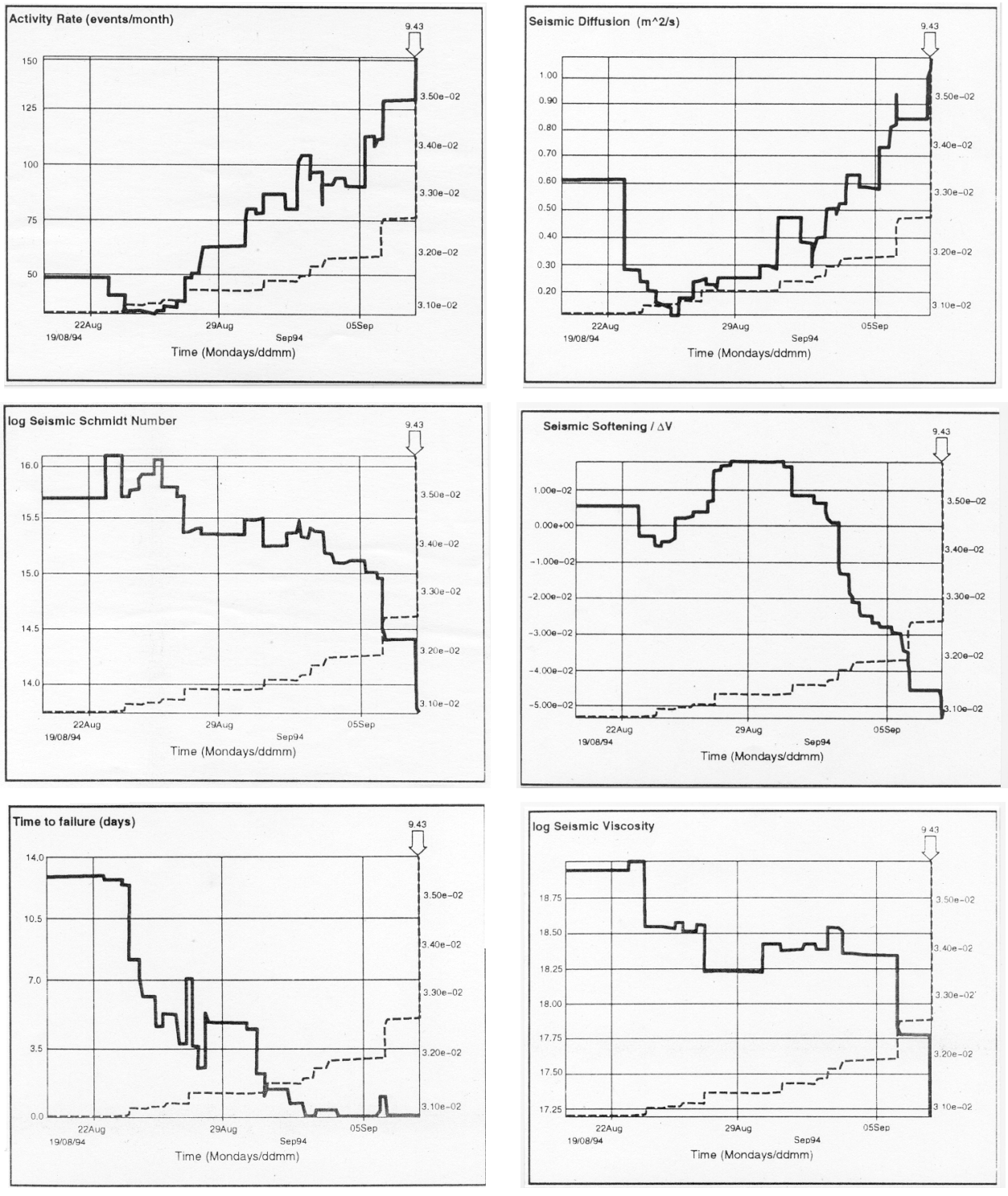

Figure 4.4: Seismic Parameter reactions before a $\log E = 9.43$ sized event

or to the softer nature of individual events occurring during nucleation. A small decrease in seismic activity has also been observed at the beginning of the strain softening stage. This is followed by an increase just before the instability occurs.

The following behaviour is a good indication of an impending breakdown. Firstly, an overall decrease in the average value of the energy index EI, seismic softening S_s, and seismic viscosity ν_s with time is observed in seismic events close to the hypocentre of the impending instability. The softened zone is characterised by a low seismic Deborah number. However, short periods of increase and/or oscillations in these parameters have also been observed. Secondly, an increased rate of coseismic deformation as measured by the cumulative apparent volume ΣV_A, and seismic diffusion D_s, can be observed. This behaviour is caused by the increased seismic activity rate, as well as the softer nature of events occurring within the already fractured zone. Seismic softening coinciding with accelerating deformation cause a decrease in seismic viscosity ν_s, and an increase in seismic diffusion D_s, with a resultant increase in the seismic Schmidt number Sc_{sd}. In general, the sensitivity of the seismic monitoring system is a limiting factor in the detection of nucleation processes. Stronger and more homogeneous rock mass have smaller nucleation zones, and are therefore harder to detect.

4.8 Summary

Seismic monitoring, event prediction, and risk management have become a very important part of modern mining operation. Advances made in digital technology provide the means to make measurements with accuracy levels which were unthinkable only a few years ago. The emphasis has now shifted from accurate measuring to accurate modelling of the processes governing the elastic and inelastic rock deformations. State-of-the-art event prediction methods can predict up to one out of every three inelastic deformations of magnitude two and higher. Thus, there is still much room for improvement.

Neural networks have already been used in the validation phase of the data flow. In the next chapter, we will discuss the use of neural networks for the modelling of the seismic time series itself.

Chapter 5

Prediction and Modelling of Seismic Time Series

5.1 Introduction

Neural networks have been used for seismic discrimination (e.g. to distinguish natural earthquakes from man-made explosions [59]), and for the interpretation of seismic data for gas and oil exploration [1]. To the best of our knowledge, no work has been done on the use of neural networks for seismic event prediction in hardrock mines. Modelling and prediction of seismic time series are very difficult, because they originate from high-dimensional chaotic attractors. The high degree of noise present in the measurements complicates matters further. Fortunately, neural networks have been shown to be robust in their capability to cope with noise [6]. As we have demonstrated in Chapter 3, neural networks are capable of extracting accurate attractor models from chaotic time series.

In Chapter 4, we discussed the different source parameters that characterise seismic events. They are: the time of occurrence, the location of the event, the location measurement error, the number of monitoring stations involved in the measurement, the moment, the energy, and the estimated

error in the moment and energy measurements. Any of these parameters can be used to construct a time series, but for the purposes of this thesis, we will only focus on the energy and moment.

We had available seismic event data from a hard rock gold mine in South Africa. The influence of events originating in one part of the mine on events in distant parts of the mine is unclear. Thus we used a data set containing localised events. As an illustration for the process of time series characterisation, we will examine two time series in detail: an energy and moment time series, containing seismic events from a section of the mine called *polygon k*.

In this chapter, we discuss the application of the methods discussed in previous chapters to a real world time series. We show that modelling and predicting time series from real world applications are considerably more difficult than our benchmark systems. One of the reasons for this increased difficulty is the presence of noise in the data. In the next section we discuss our method for pre-processing the raw data into useful time series. We discuss the characterisation of two example time series. Next, we show that the methods of false nearest neighbours and false strands find it much harder to extract embedding dimensions from the time series. We then show our results for TDNN and FIRNN. These results are representative of many experiments performed. This section is followed by a section on regularisation. We conclude the chapter with a discussion of our results.

5.1.1 Data Preprocessing

For time series prediction, neural networks need to represent time. This is normally done by sampling a continuous signal at a fixed frequency. The seismic events are recorded as they occur, and are therefore discrete, and irregularly spaced in time. We address this problem by dividing the time series into fixed size time intervals. We chose the biggest event in the interval to represent the interval. Thus, one-step ahead prediction will predict the biggest event that will occur within the next interval.

Our neural networks use sigmoidal activation functions. Thus, training data that is much bigger than one, or much smaller than zero, will drive the neurons into saturation. Training with neurons in

saturation mode is difficult, since the gradient in this region is small. The true energy and moment values of seismic events range from 10^2 to 10^{14}. A linear scaling would have dwarfed the smaller events. Thus, we take the logarithm, and then scale the data linearly to lie between zero and one. The time series are shown in Figures 5.1 and 5.2. The time series have 4000 data points, and a binning period of four hours. Thus, the time series covers a period of about a year and 10 months.

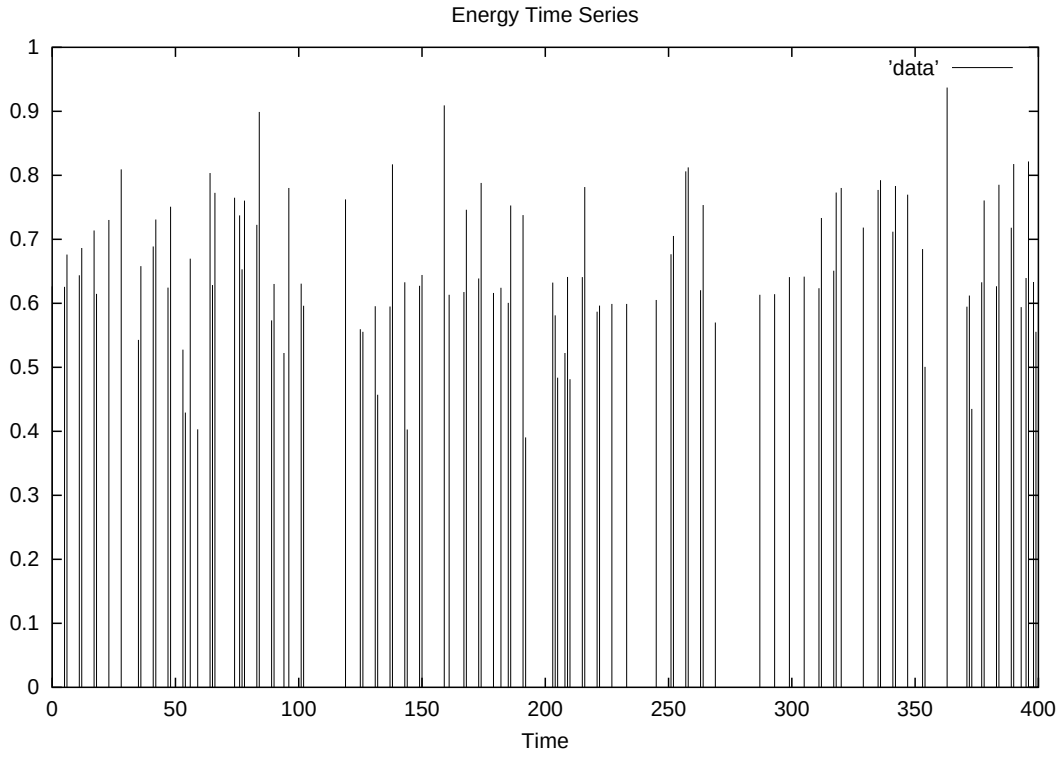

Figure 5.1: The energy time series. The binning period is 4 hours, and thus each data point on the x-axis represents a 4 hour interval. As discussed in Section 5.1.1, the data is logscaled to lay between zero and one.

5.2 System Characterisation

In this section, we follow the same route for system characterisation as Chapter 2. We calculate the power spectrum, correlation dimension, Lyapunov spectrum and Poincaré map for each of the time series.

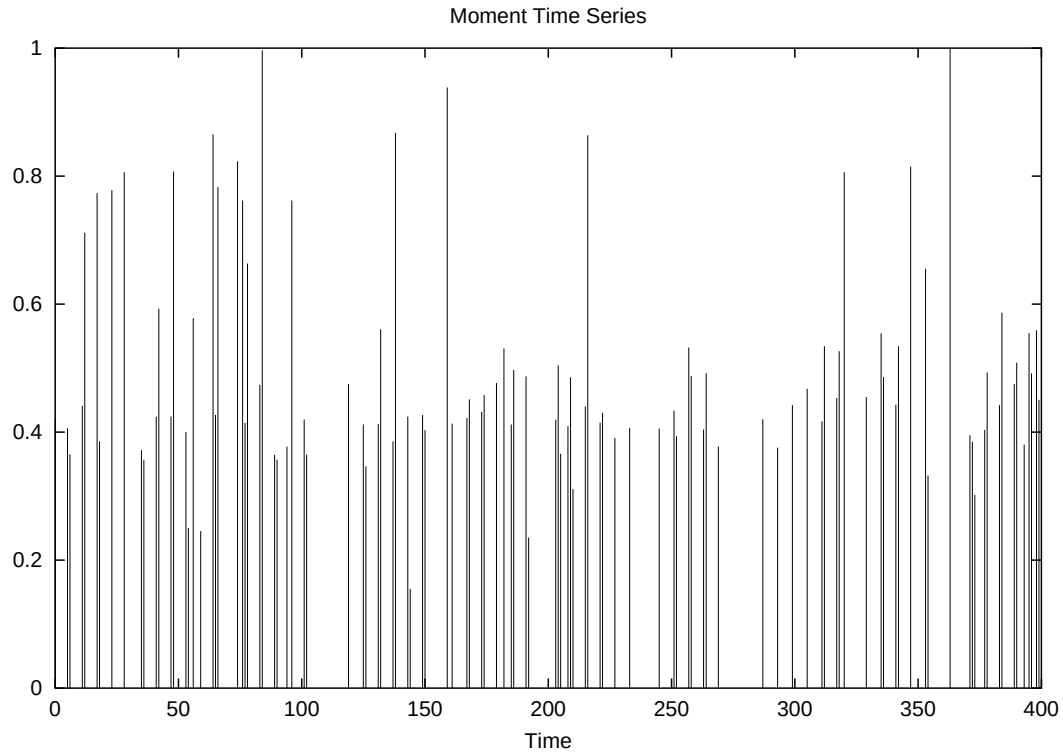

Figure 5.2: The moment time series. The binning period is 4 hours, and thus each data point on the x-axis represents a 4 hour interval. As discussed in Section 5.1.1, the data is logscaled to lay between zero and one.

5.2.1 Power Spectrum

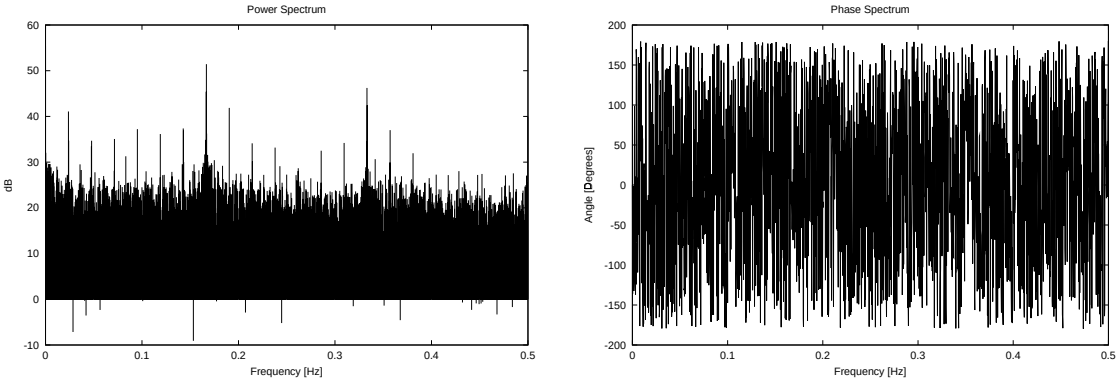

Figure 5.3: The log-magnitude and phase angle graphs for the energy time series

We begin by examining the power spectrum of both time series, as shown in Figures 5.3 and 5.4. From the rich spectra in both the magnitude and angle plots, it is clear that neither time series stem

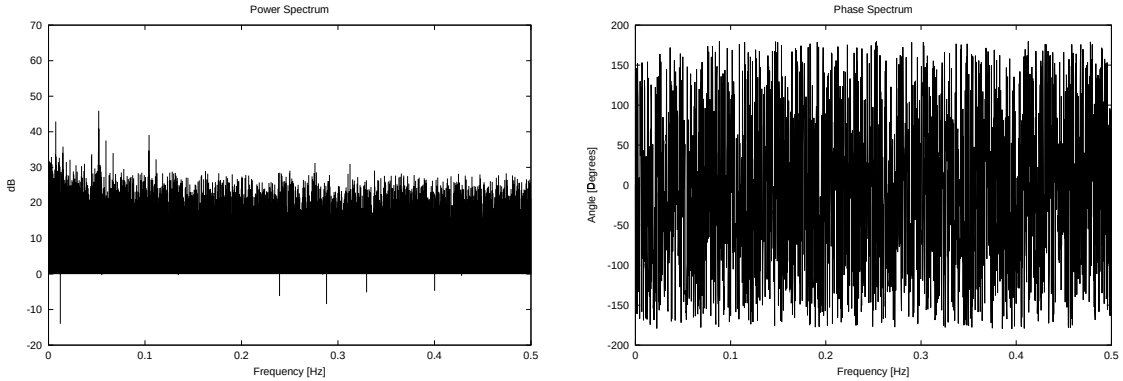

Figure 5.4: The log-magnitude and phase angle graphs for the moment time series

from a periodic attractor. Thus, we continue and calculate the correlation dimensions.

5.2.2 Correlation Dimension

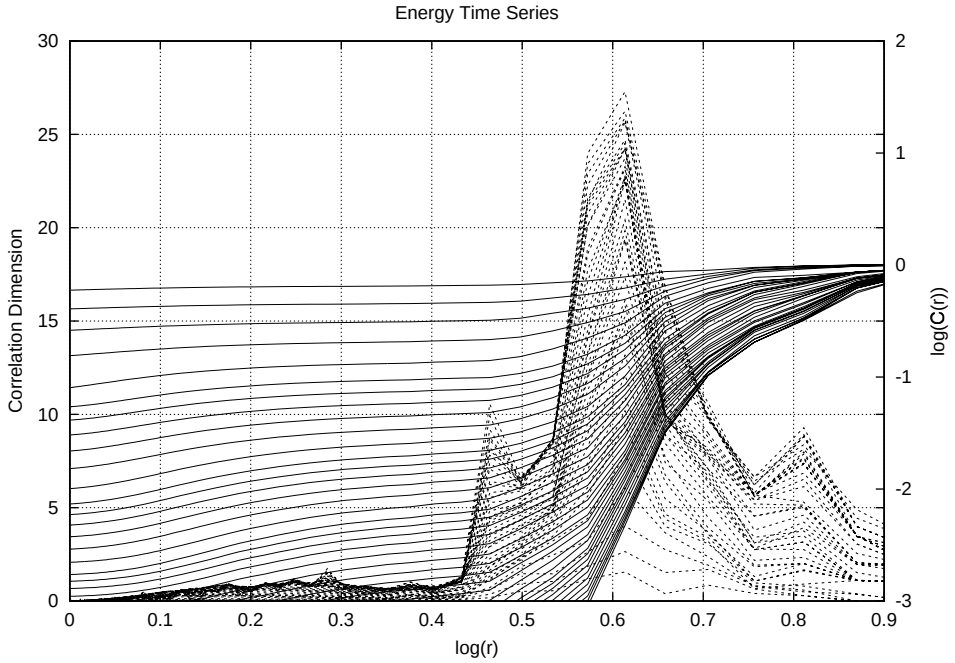

Figure 5.5: The correlation dimension and correlation sum graphs for the energy time series. The correlation sum has a skewed S-shape, and its scale is shown on the right border.

The correlation sums and correlation dimensions for the first 40 embedding dimensions are shown

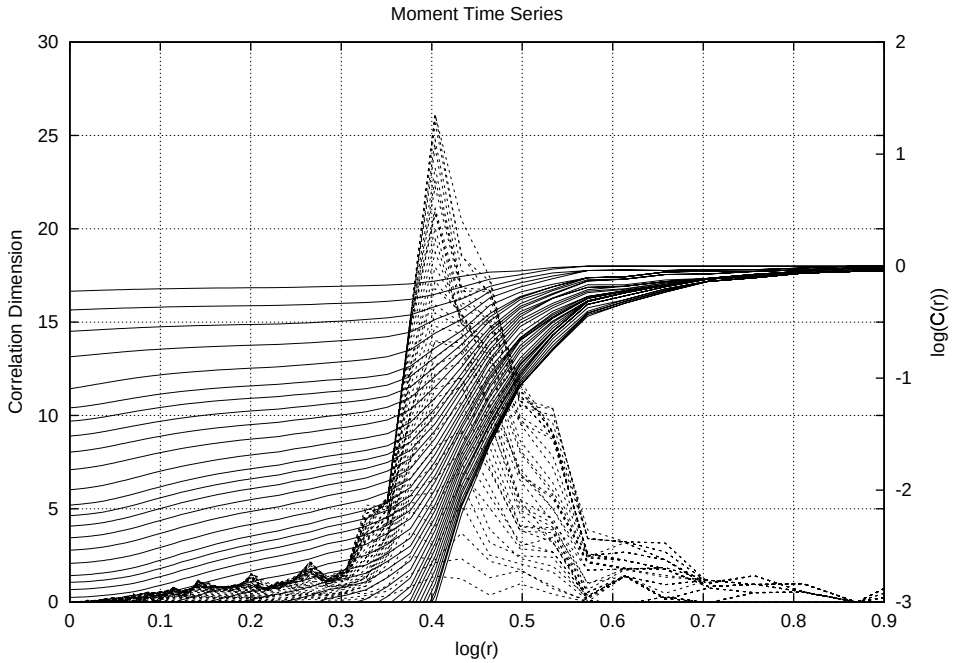

Figure 5.6: The correlation dimension and correlation sum graphs for the moment time series. The correlation sum has a skewed S-shape, and its scale is shown on the right border.

for the energy and moment time series in Figure 5.5 and Figure 5.6, respectively. The correlation dimension is obtained from the gradient of the correlation sum at its most linear point. We observe that the correlation dimensions are much higher than those of the Lorenz, Rössler, Mackey-Glass or laser systems. This suggests that these time series will be more difficult to predict.

5.2.3 Lyapunov Spectrum

To confirm our suspicion that the time series are chaotic, we proceed to calculate the Lyapunov spectra. Not surprising, both time series contain positive Lyapunov exponents, and the spectra are shown in Figures 5.7 and 5.8. Note that largest Lyapunov exponent of the moment time series is higher than that of the energy time series. We expect the moment time series to be more difficult to model.

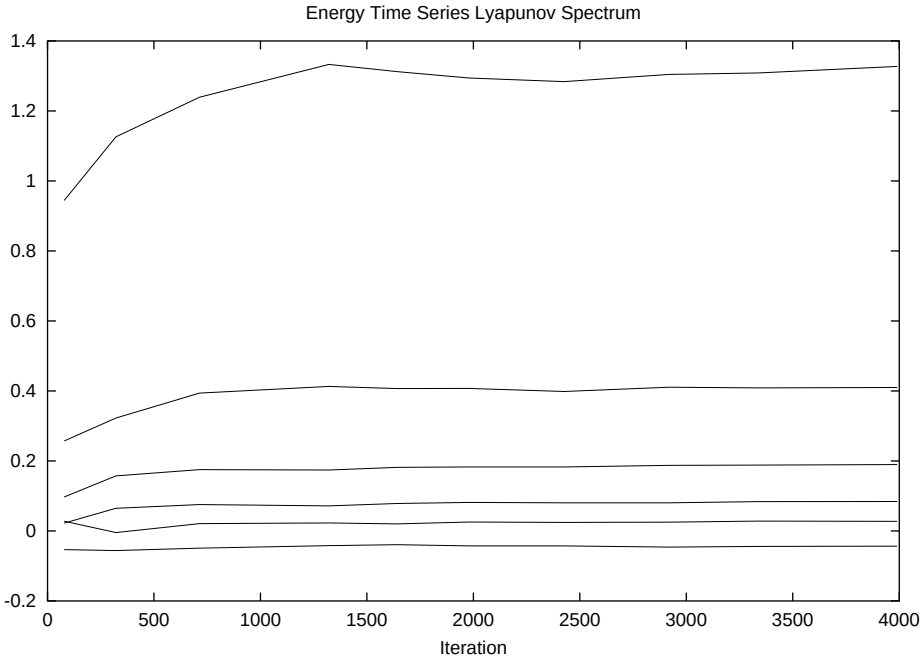

Figure 5.7: The Lyapunov spectrum of the energy time series

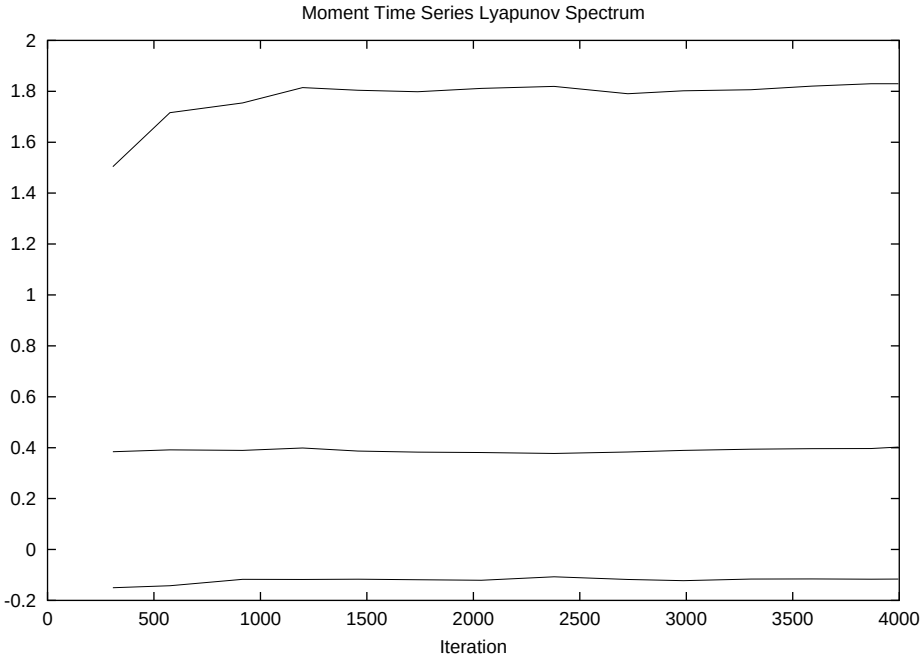

Figure 5.8: The Lyapunov spectrum of the moment time series

5.2.4 Poincaré Map

Finally, we draw the Poincaré maps for the moment and energy time series. The graphs shown in Figures 5.9 and 5.10 show the attractors less clear than that of our benchmark systems. However, some underlying structure is evident, as opposed to no distinguishable structure such as that of the random time series that we discussed in Section 2.5.2.

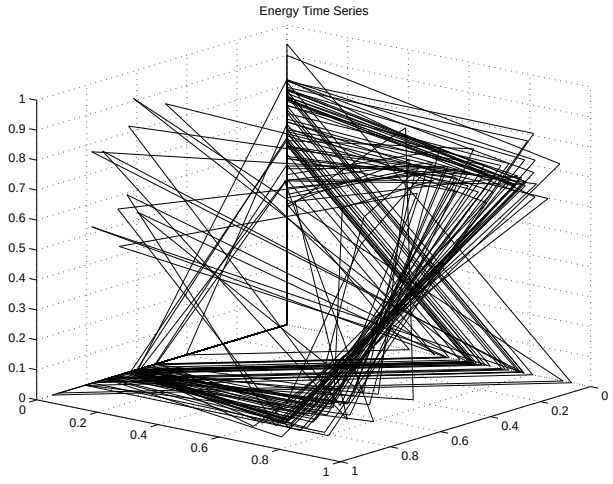

Figure 5.9: The energy time series map. The figure shows the development of $(x(t), x(t-1), x(t-2))$ as t increases.

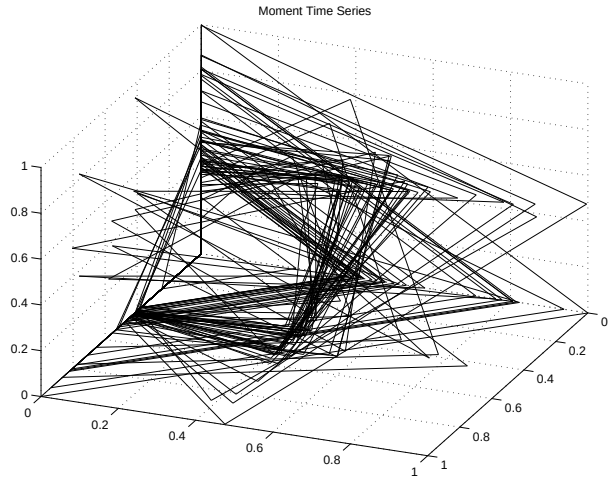

Figure 5.10: The moment time series map. The figure shows the development of $(x(t), x(t-1), x(t-2))$ as t increases.

5.3 False Nearest Neighbours and False Strands

The seismic time series contains discrete events, and therefore we do not perform time lag estimations. We proceed to estimate the embedding dimension using the methods of false nearest neighbours and false strand methods.

Figure 5.11 shows the results from both methods for the energy time series. The method of false nearest neighbours seems to converge, and suggests embedding dimensions of 37 and higher. The method of false strands never reach low false to true strand ratios. At embedding dimension 43, the average suddenly drops, and seems to stay lower. This gives some support to the estimation of the false nearest neighbours method. Figure 5.12 shows the results for the moment time series. Again, the method of false strands fails to provide any good indication of a good embedding dimension. The method of false nearest neighbours reaches its smallest value at an embedding dimension of 43. However, the method of false strands is an improvement on the method of false nearest neighbours. Thus, the failure of the method of false strands casts some suspicion on the contradicting result obtained with the method of false nearest neighbours.

As we discussed in Chapter 4, the seismic moment and energy are not independent parameters. Taking all results into account, we conclude that embedding dimensions of ± 40 should be used for both time series.

5.4 Performance Measures

Before we discuss the modelling of the time series, we pause to reflect on performance measures. A common performance measure is the mean squared error (MSE):

$$MSE_t(\vec{x}) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - d_i)^2 \tag{5.1}$$

where y_i and d_i are the output and desired output for pattern i, respectively, for a total of N input patterns.

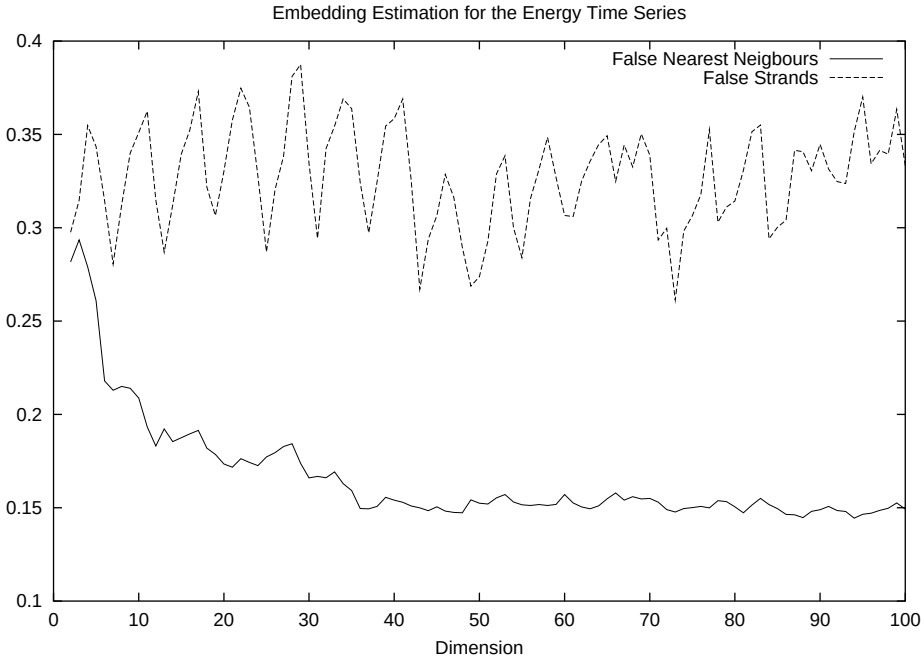

Figure 5.11: The methods of false strands and false nearest neighbours performed on the energy time series. Both methods give the ratio of false strands/nearest neighbours to true strands/nearest neighbours.

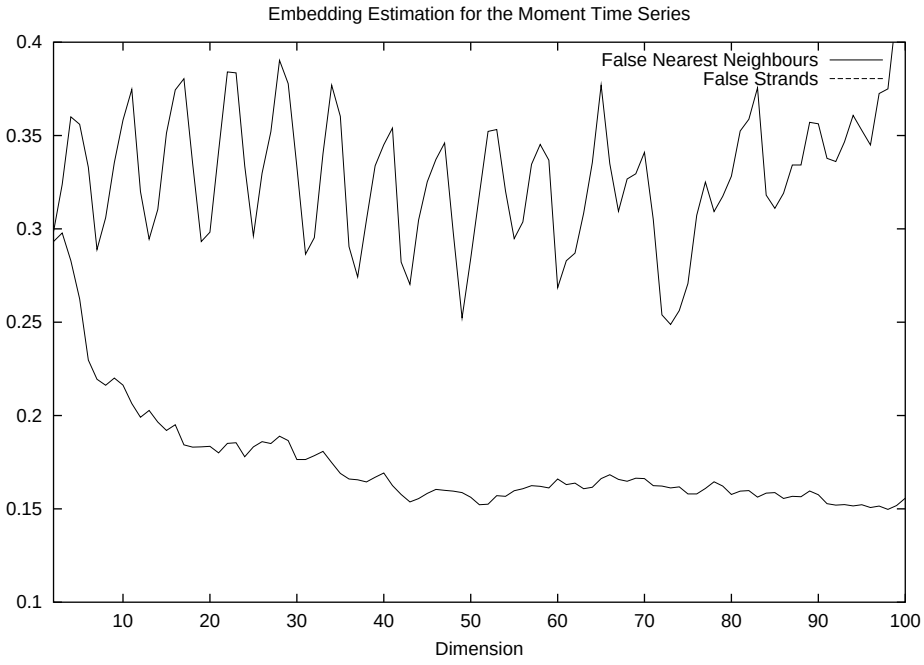

Figure 5.12: The methods of false strands and false nearest neighbours performed on the moment time series. Both methods give the ratio of false strands/nearest neighbours to true strands/nearest neighbours.

The mean squared error gives a general idea of how well the model fits the given data. Keep in mind that the error on both the training set and test set should be used for evaluation. Using only the training MSE will give an overly optimistic picture of the degree to which the network models the underlying attractor.

MSE may place too much emphases on big differences. Another measure is the mean absolute error (MAE), given by:

$$MAE_t(\vec{x}) = \frac{1}{2N}\sum_{i=1}^{N}|y_i - d_i| \tag{5.2}$$

For the specific application of event prediction, both the MSE and MAE lack one important functionality: the ability to discriminate between more and less important events. It is important to be able to track smaller sized events, but it is even more important to predict possible disasters.

Thus, the use of receiver operating characteristic (ROC) curves is warranted. This method employs a threshold; predictions above the threshold are regarded as disaster warnings, and predictions below the threshold are ignored. Let a big event be an event above the threshold. We define the probability of detection (P_d) as the ratio of detected big events to the number of big events. The probability of detection is used in conjunction with another measure, called the probability of false alarms (P_f). The probability of false alarms is defined as the ratio of false big event predictions to the number of big event predictions. Thus, a P_d of one and a P_f of zero is the ideal.

By varying the threshold, and drawing P_d on the y-axis and P_f on the x-axis, a ROC curve is drawn. An operating point can then be chosen according to the value placed on predicting disasters as opposed to the cost on acting on false alarms. We will explicitely show the influence of increasing the threshold by plotting both P_d and P_f as functions of the threshold.

Another performance measure is to draw the histogram of prediction errors. This method counts the number of points lying within an error interval. Thus, the desired response is to have all the errors lying within a narrow region around zero. In essence, the error histogram method provides the same information as the MSE and MAE, but it is more descriptive, visual and intuitive.

Network performance can be viewed from a mining engineering, or from a nonlinear dynamical

systems point of view. For the former, ROC curves for performance evaluation is recommended, as accurate tracking of the chaotic time series is not necessary. From the dynamical systems point of view, we are interested in how closely the attractor is modelled. Here, the histogram method might be of more interest, since it is not biased to event sizes.

5.4.1 Stopping Criterion for Seismic Time Series

We investigated the use of Diks' Test as a performance measure for the seismic time series. Unfortunately, we found no conclusive evidence that the test can be used as a stopping criterion for the seismic data. We believe the test failed for our application, because the seismic time series are too noisy. The validation error never reaches values that would necessitate the use of Diks' Test to discriminate whether the model is truly an accurate representation of the original attractor. Therefore, the results of the test never reach acceptable values. We also found no evidence that the trend of the test can be used as a stopping criterion. We often found that better P_d values are reached with more training than that suggested by Diks' Test.

We reverted to using P_d and P_f as stopping criteria. We continued training while the P_d for the test set kept increasing and the P_f stayed at acceptable levels.

5.5 Modelling and Predicting Seismic Time Series

In this section we show the modelling capability of neural networks when applied to seismic time series prediction. The results presented in this section are representative of many experiments performed.

We first present results for the radiated seismic energy, and then for the seismic moment. Throughout this section, whenever we show predicted data, we will use stippled lines for the desired response, and solid lines for network output.

5.5.1 Radiated Seismic Energy

All the input data presented in this section is scaled between zero and one according to the method discussed in Section 5.1.1. A one-to-one map of the logscaled values to the real values is shown in Figure 5.27.

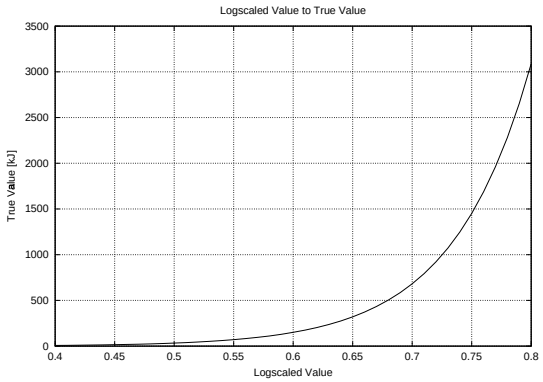

Figure 5.13: Mapping of logscaled values to actual values in kJ.

Modelling and Prediction Using TDNN

We show results for a TDNN trained on the first 1000 data points of the energy time series. We used a TDNN with 40 input neurons, as indicated by the methods of false nearest neighbours and false strands. The network had 30 neurons in its first hidden layer and 3 in its second hidden layer. It had one neuron with a linear activation function in the output layer. We trained the network for 2500 epochs. As shown in Figure 5.14, the network performed well when predicting the training data, and obtained a mean squared error of 0.023 on the training data. Results for predicting the unseen test data are shown in Figure 5.15. The network output can be larger than one because the output neuron is linear. The mean squared error on the test data was 0.074.

We evaluate the network performance using the methods discussed in Section 5.4. The ROC curves for the training data are shown in Figure 5.16. Note that we split the normal ROC curve into two curves, explicitly showing the influence of the threshold. An operating point can be chosen by specifying a desired threshold and obtaining the probability of detection and probability of

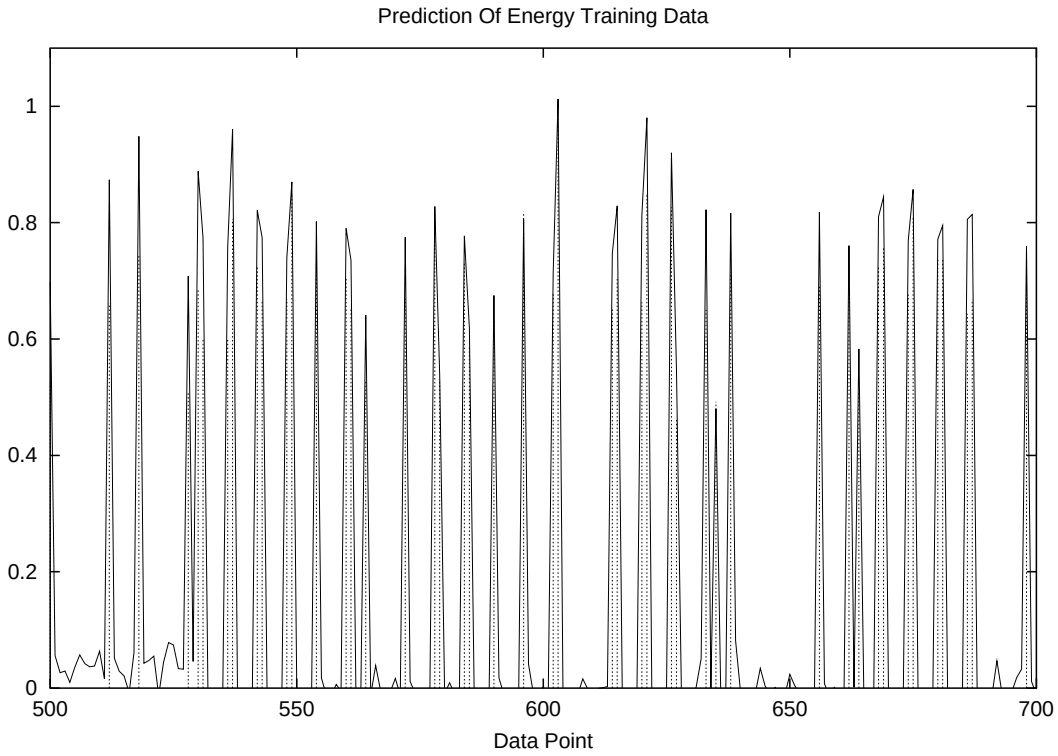

Figure 5.14: TDNN Prediction of energy training data.

false alarm from the graph. We observe that accurate prediction becomes increasingly hard as the threshold increases. The ROC curves for prediction of the next 350 test data points are shown in Figure 5.17. From the positive false alarm probability for small thresholds, we deduce that sometimes events are predicted when there are none.

We chose the criteria for event detection and false alarm prediction such that an event is only marked as predicted if the prediction is above the threshold. Thus, if events of size 0.6 and larger are to be detected, and the prediction is 0.59, it does not count as a prediction. Similarly, when an event is of size 0.59, the threshold 0.6, and the prediction 0.6 as well, it is designated as a false alarm.

The histogram of errors for both the training and test set are shown in Figure 5.18 and Figure 5.19, respectively. We do not count correct predictions of non-events when calculating the histogram. Including these predictions causes the histogram scoring method to be overly optimistic.

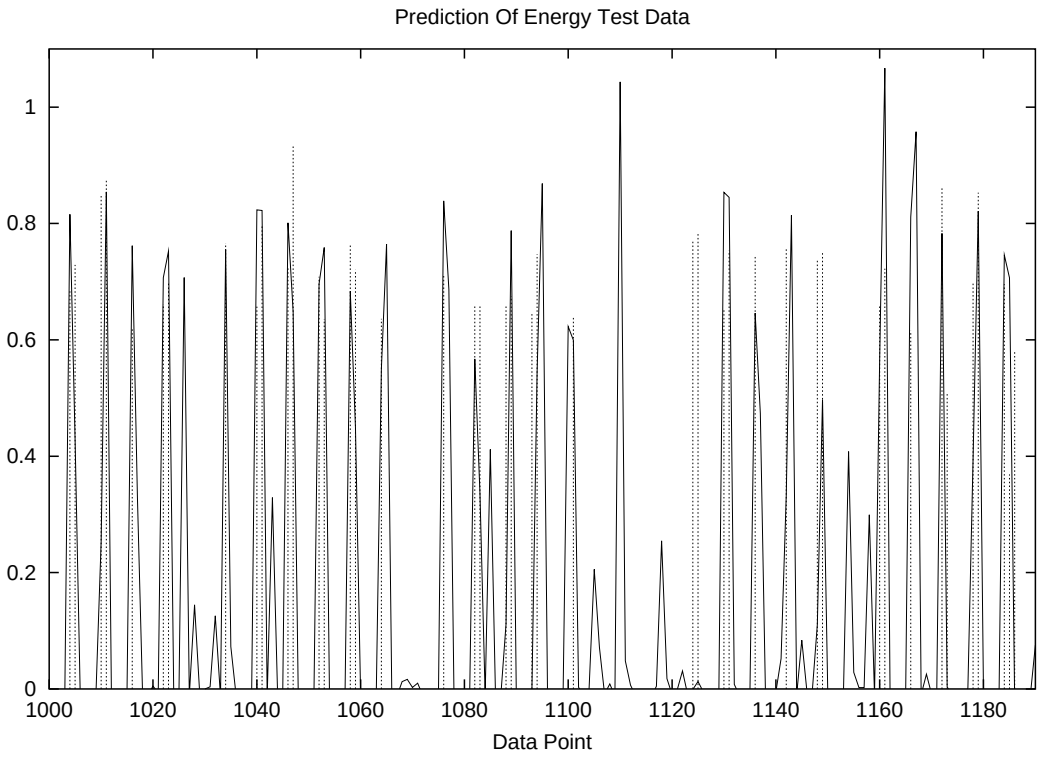

Figure 5.15: TDNN Prediction of energy test data.

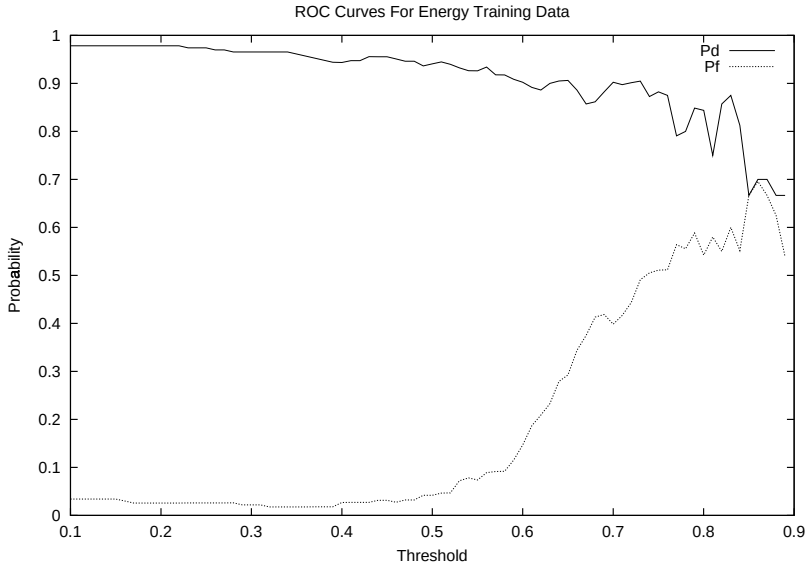

Figure 5.16: The probability of detection and probability of false alarm curves for energy training data.

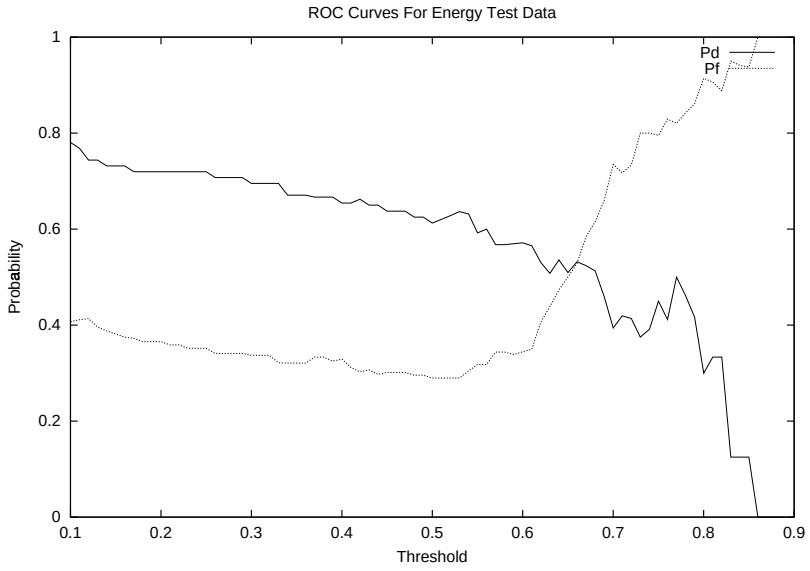

Figure 5.17: The probability of detection and probability of false alarm curves for energy test data.

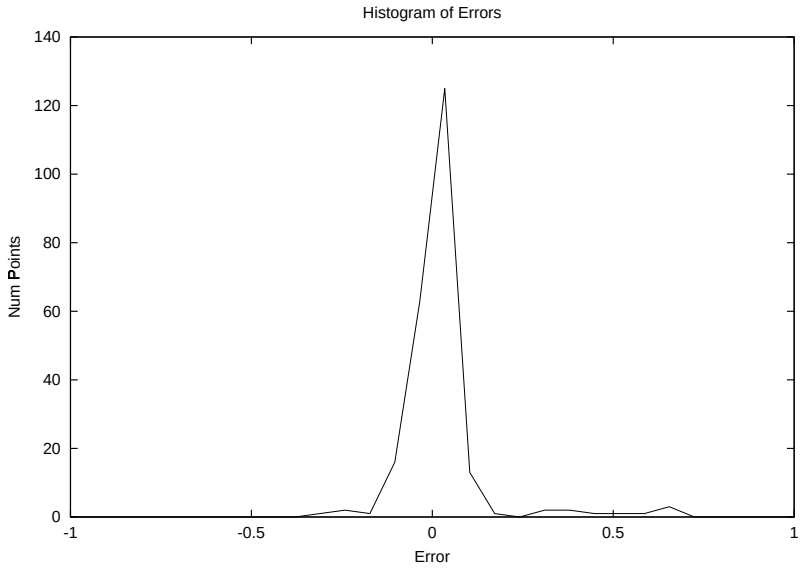

Figure 5.18: The histogram of errors made on the prediction of training data.

Next we show the results for training up to 4000 data points. The results for the training set were similar to those already shown, thus we only show results for the test set. Figure 5.20 shows the first 200 predicted points.

The ROC curves for the first 400 predicted points are shown in Figure 5.21. The probability of

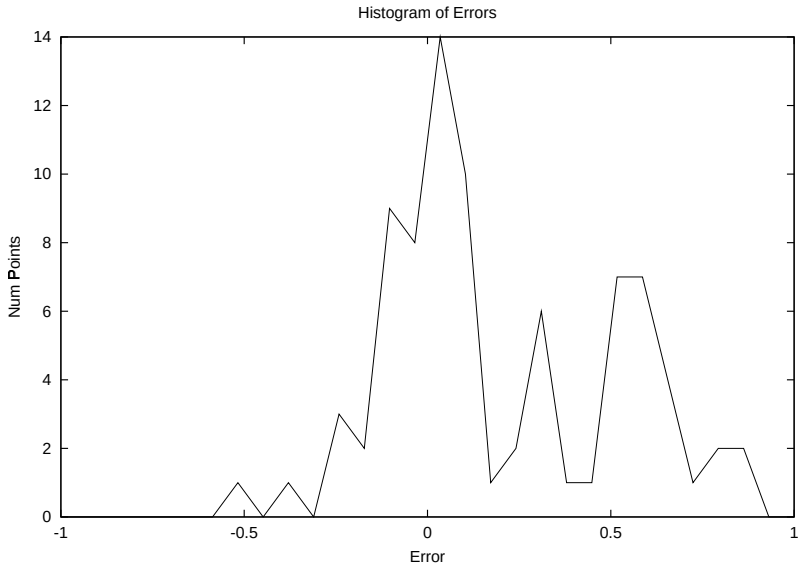

Figure 5.19: The histogram of errors made on the prediction of test data.

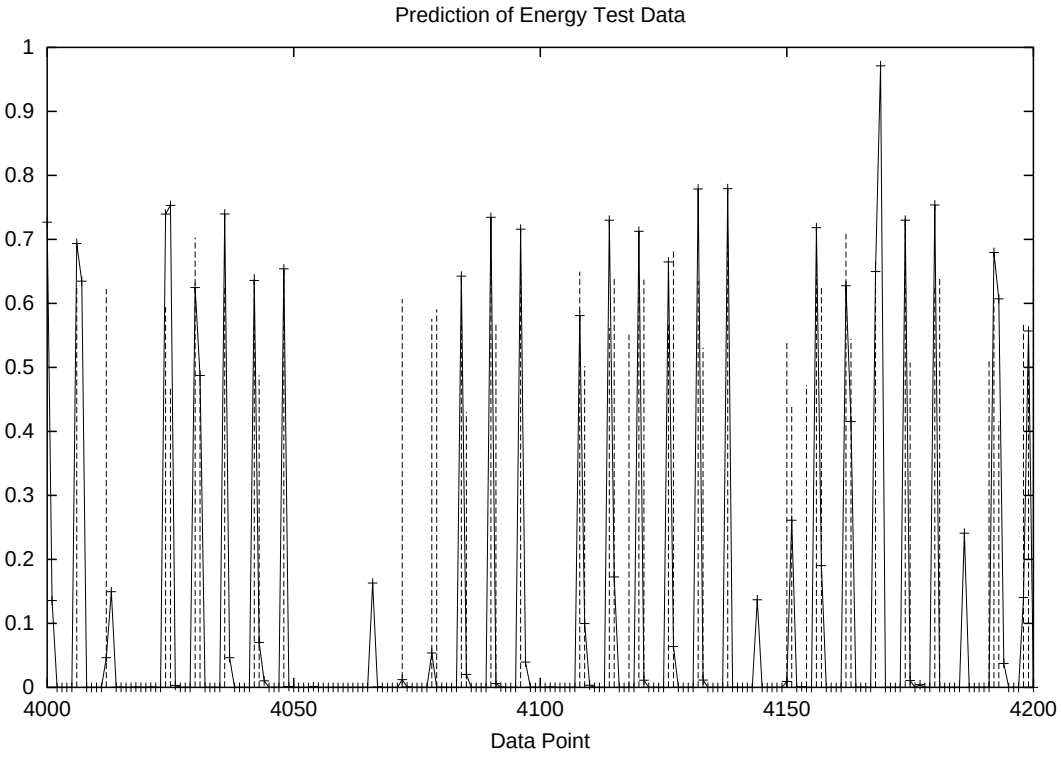

Figure 5.20: Prediction of energy test data when using the first 4000 data points as training data.

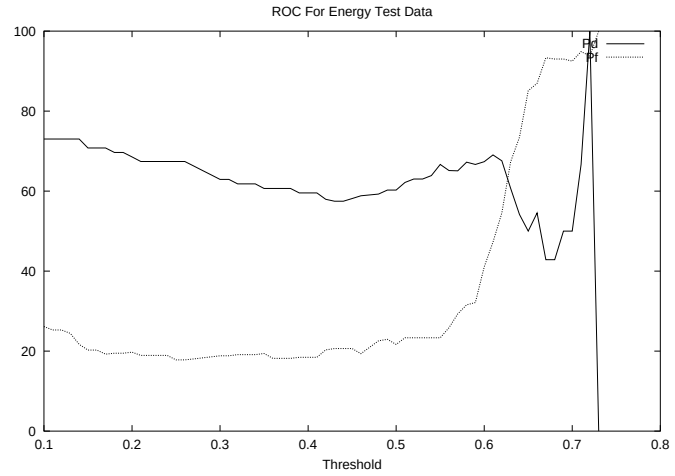

Figure 5.21: The probability of detection and false alarm curves for the energy test data.

detection for events up to size 0.7 is reasonably good at above 0.5. At a threshold value of 0.72 the probability of detection is 1 for a short period, and then drops to 0. This effect is the result of the ever decreasing number of big events remaining as the threshold is increased. Thus, the one or two events above 0.72 are all detected, giving a probability of detection of 1, and a number of false alarms are made.

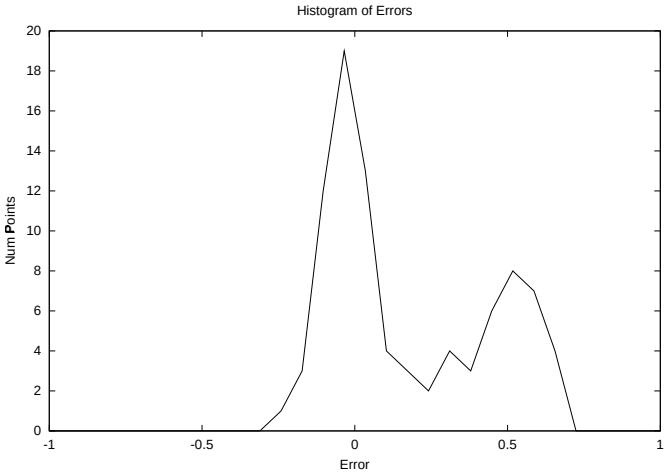

Figure 5.22: A histogram of the errors made on the prediction of energy test data

Figure 5.22 shows the histogram of errors on the test data. It shows a concentration around zero, and a smaller concentration about 0.5. This hump was present in Figure 5.19 as well, and is the

manifestation of event predictions where there are none.

Modelling and Prediction Using FIRNN

We show results for a FIRNN trained for 7400 epochs. A section of the prediction of training data is shown in Figure 5.23. The FIRNN did not predict training data as well as the TDNN. However, it performed better on the prediction of test data, as shown in Figure 5.24.

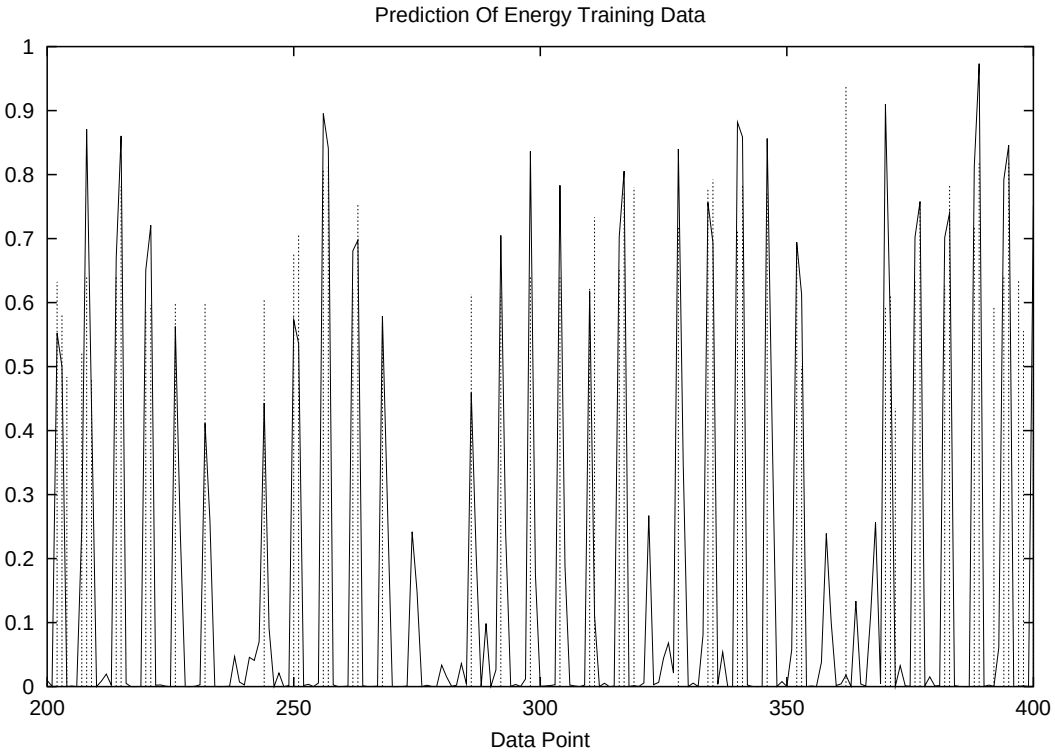

Figure 5.23: Prediction of energy training data with a FIRNN.

When comparing the ROC curves for training data shown in Figures 5.16 and 5.25, respectively, we observe that the FIRNN performed worse than the TDNN. However, from Figures 5.17 and 5.26 we observe that the FIRNN gave better generalisation performance.

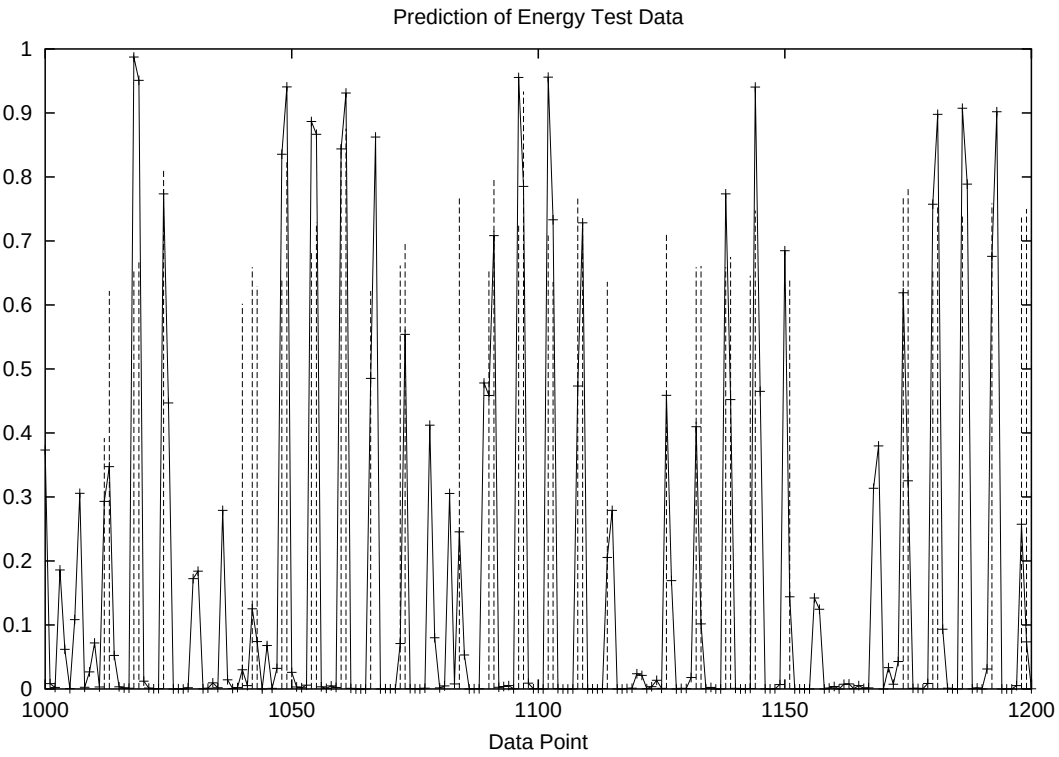

Figure 5.24: Prediction of energy test data with a FIRNN.

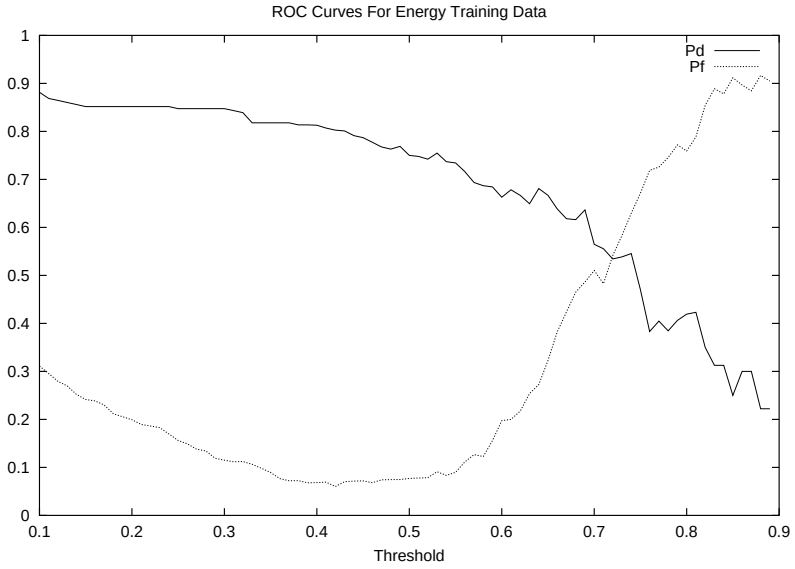

Figure 5.25: The probability of detection and false alarm curves for energy training data for the FIRNN.

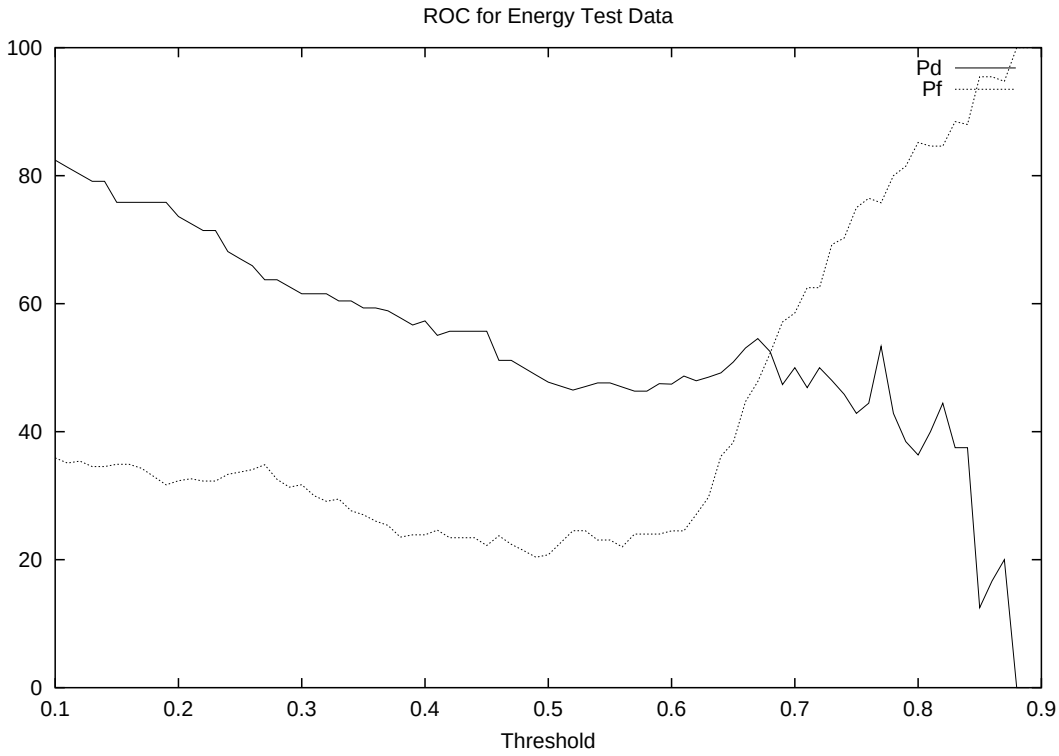

Figure 5.26: The probability of detection and false alarm curves for energy test data for the FIRNN.

5.5.2 Seismic Moment

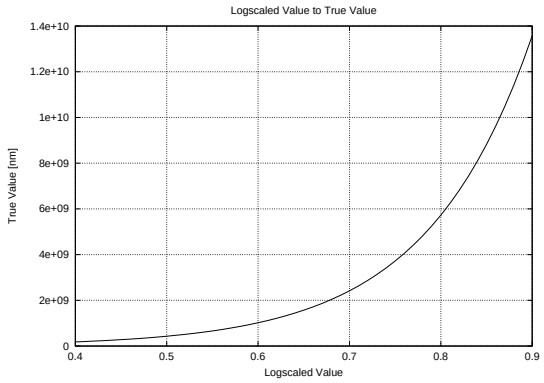

Figure 5.27: Mapping of logscaled values to actual values in nm.

In this section we model and predict seismic moment. We found predicting seismic moment to be much harder than predicting seismic energy. The mapping of training data values to real values is shown in Figure 5.27.

Modelling and Prediction Using TDNN

For the seismic moment we used a TDNN that had 40 neurons in the input tap delay line, 30 hidden neurons, and a linear output neuron. We trained the network on the first 1000 data points of the moment time series. The results for the training data are shown in Figure 5.28. The results for the test data are shown in Figure 5.29. We obtained a MSE of 0.026 for the training data, and 0.065 for the test data.

Figure 5.28 suggests that the training data is not learned very well. However, the stopping criterion focus on the P_d for the test data. Further training resulted in worse P_d and P_f values for the test set. The ROC curves for training data are shown in Figure 5.30, and that of the test data are shown in Figure 5.31. The ROC curves clearly show that the moment time series is much more difficult to predict. However, when studying Figures 5.28 and 5.29 we observe that compared to the energy time series, the moment time series has much less values above 0.6. Thus, training data for values above 0.6 is sparse.

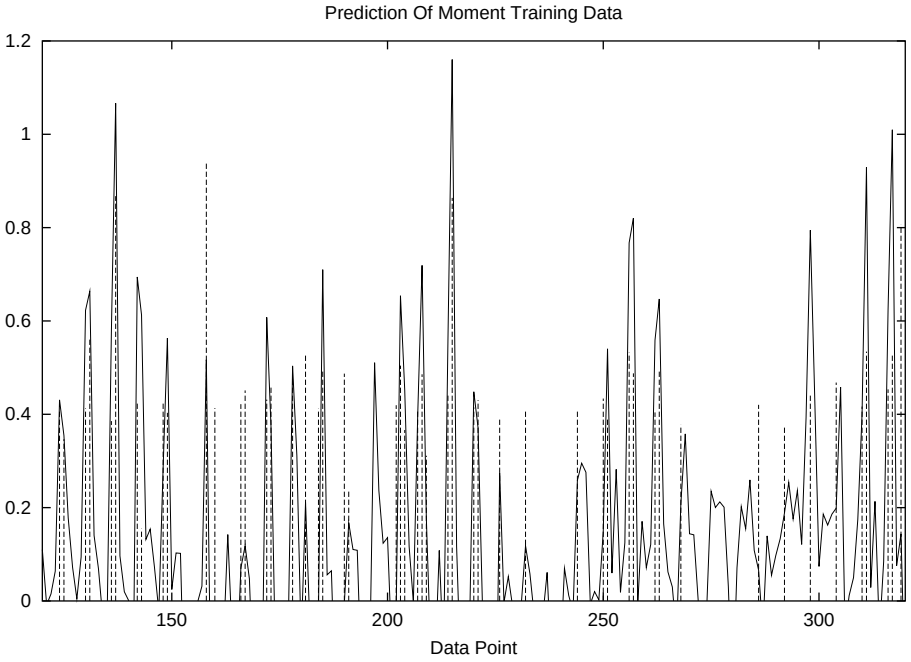

Figure 5.28: TDNN Prediction of seismic moment training data.

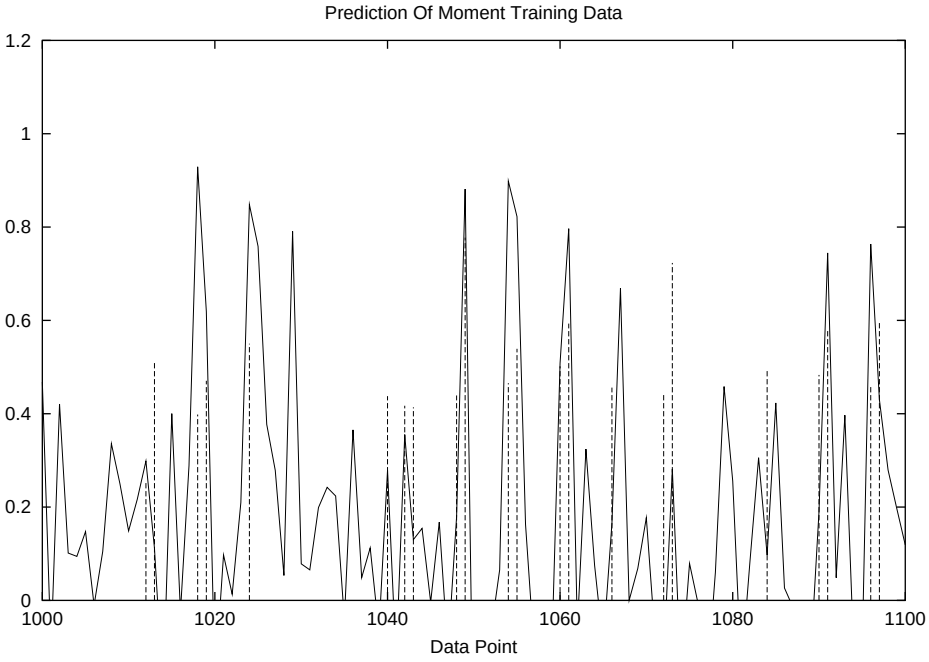

Figure 5.29: TDNN Prediction of seismic moment test data.

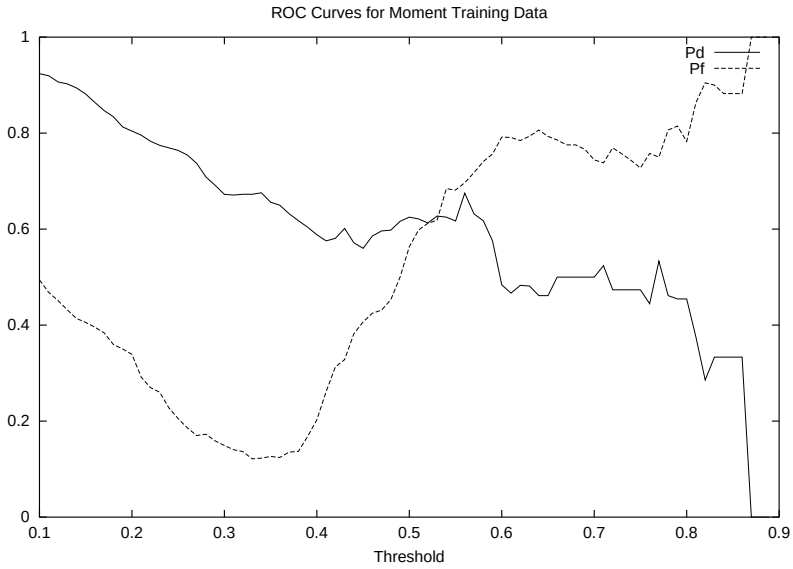

Figure 5.30: The probability of detection and false alarm curves for moment training data.

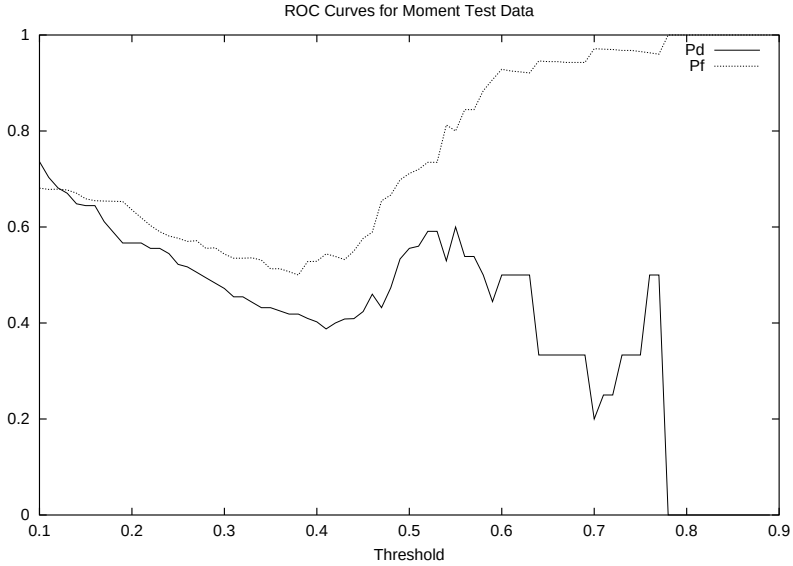

Figure 5.31: The probability of detection and false alarm curves for moment test data.

Finally, we show the error histograms for the training data in Figure 5.32, and for the test data in Figure 5.33. Again, these are not as narrow as those obtained for the energy data.

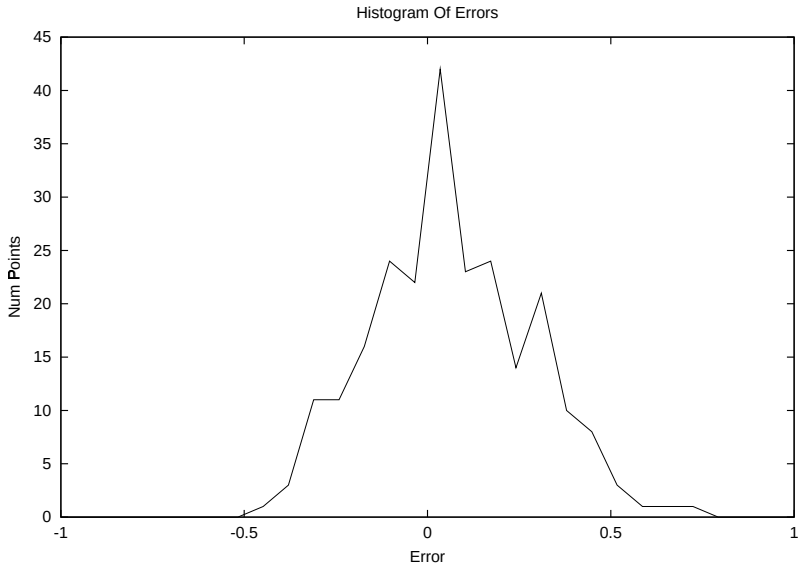

Figure 5.32: The histogram of errors for the moment training data

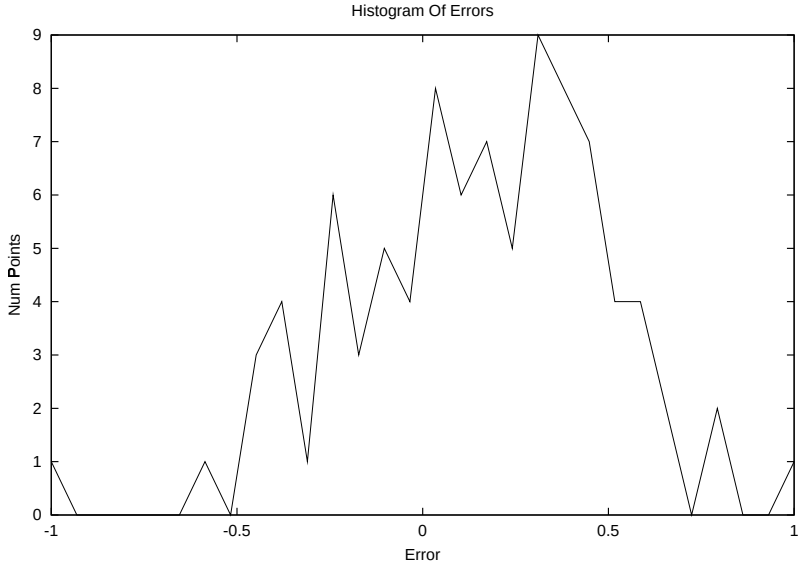

Figure 5.33: The histogram of errors for the moment test data

Modelling and Prediction Using FIRNN

For the moment data, we found that FIRNNs are much harder to train than TDNNs. However, in general, when we could train a FIRNN to model the moment data, it performed better than TDNNs.

We show the results of a 4 layer FIRNN. The input layer had 40 taps in the tap delay line. The first hidden layer had 5 neurons with 10 tap delays each, and the second hidden layer had 10 neurons with 20 taps each. We trained the network on the first 1000 data points, and predicted the next 200. We show the training data after 300 and 5300 epochs in Figures 5.34 and 5.35, respectively. Figure 5.36 shows the prediction after 300 epochs, and Figure 5.37 shows the prediction after 5300 epochs.

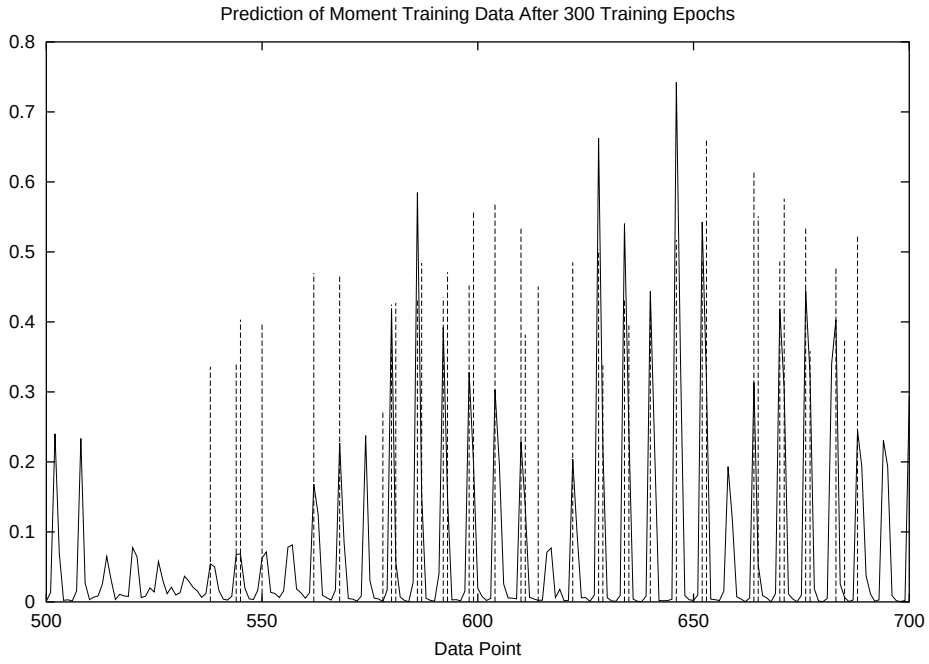

Figure 5.34: FIRNN prediction of moment training data after training for 300 epochs.

These two figures demonstrate a trend observed in general. The model is refined quite quickly to fit the data reasonably well. Improving results further usually turns out to be quite difficult. We often observed that overshooting occurs in the test data, and therefore the probability of false alarms increases beyond acceptable levels. Thus, with an increase in the probability of detection, one also observes an increase in the probability of false alarms as training progresses.

We show the training and test data ROC curves for the FIRNN after training for 300 epochs in Figures 5.38 and 5.39, respectively. We conclude this section by showing the training and test data error histograms in Figures 5.40 and 5.41, respectively.

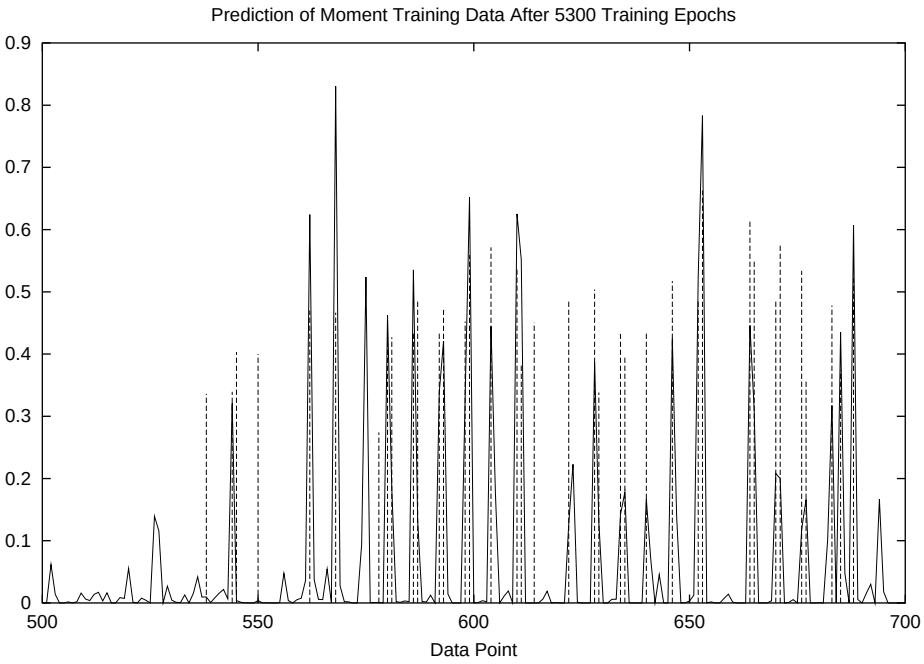

Figure 5.35: FIRNN prediction of moment training data after training for 5300 epochs.

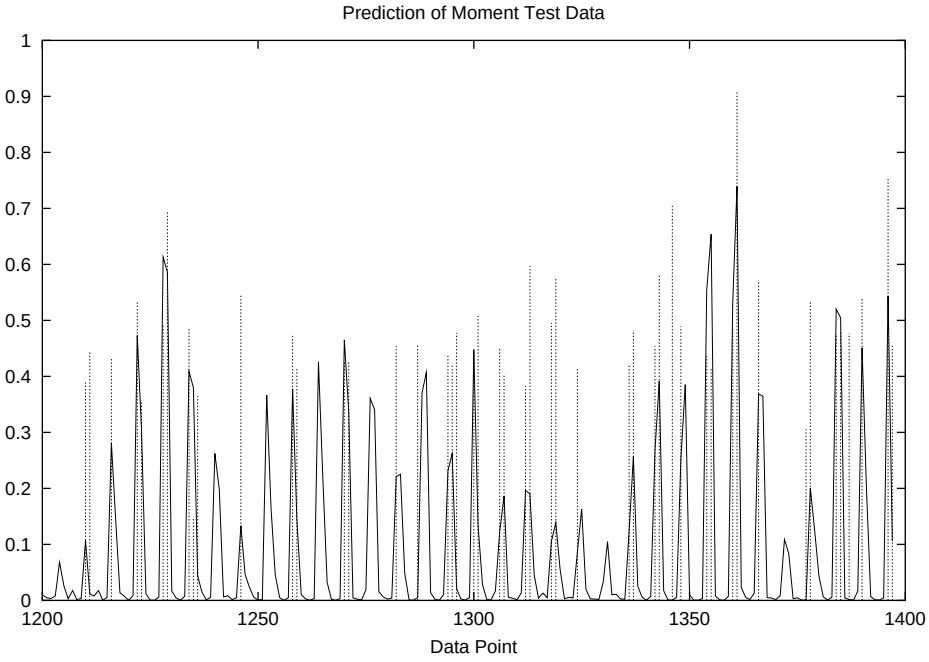

Figure 5.36: FIRNN prediction of moment test data after training for 300 epochs.

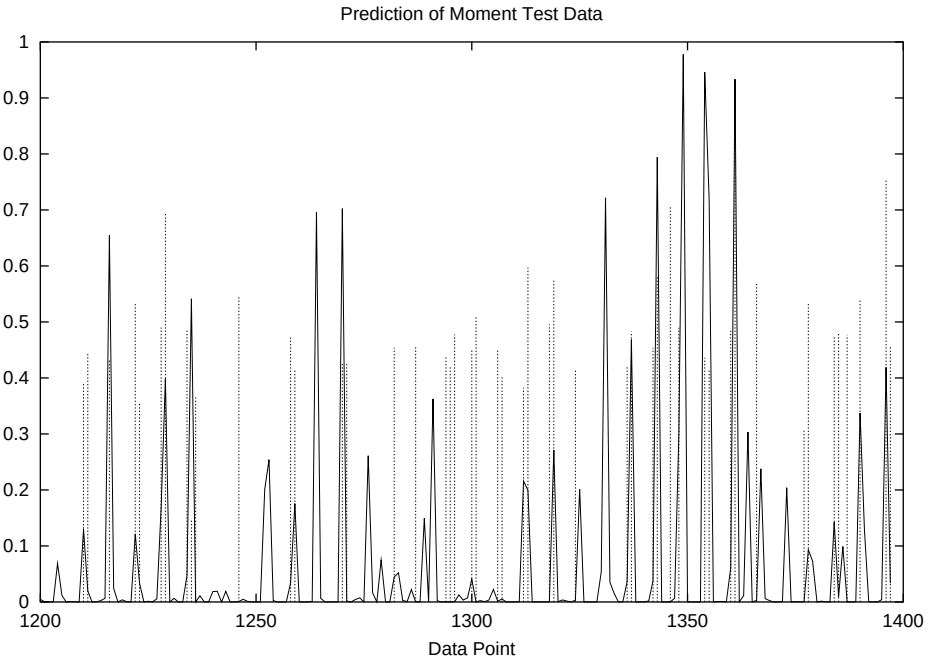

Figure 5.37: FIRNN prediction of moment test data after training for 5300 epochs.

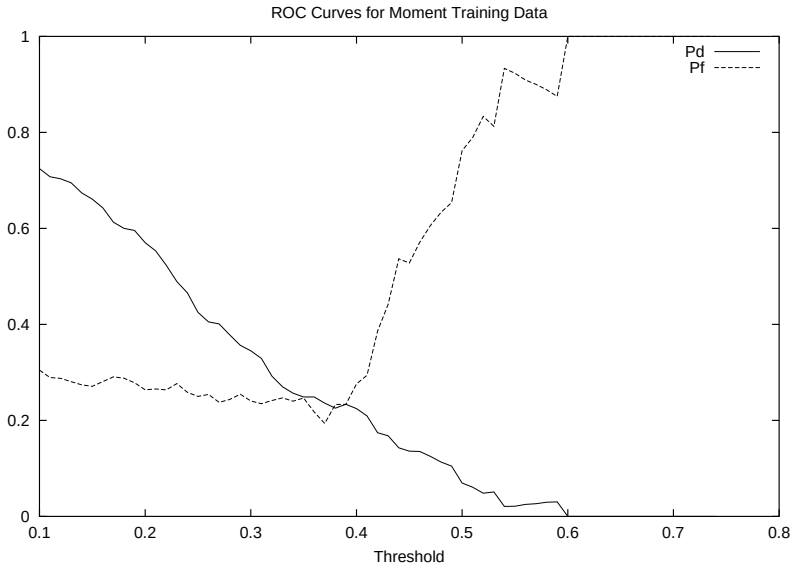

Figure 5.38: The probability of detection and false alarm for the moment training data after 300 epochs.

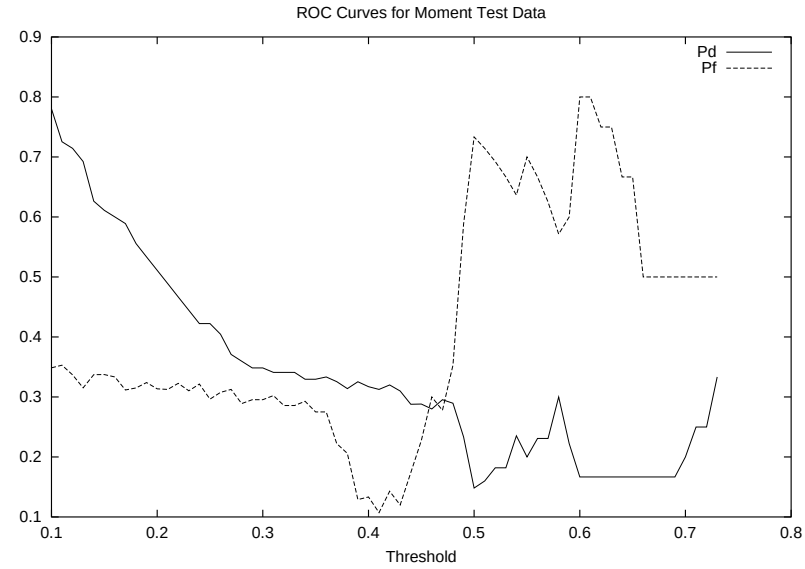

Figure 5.39: The probability of detection and false alarm for the moment test data after 300 epochs.

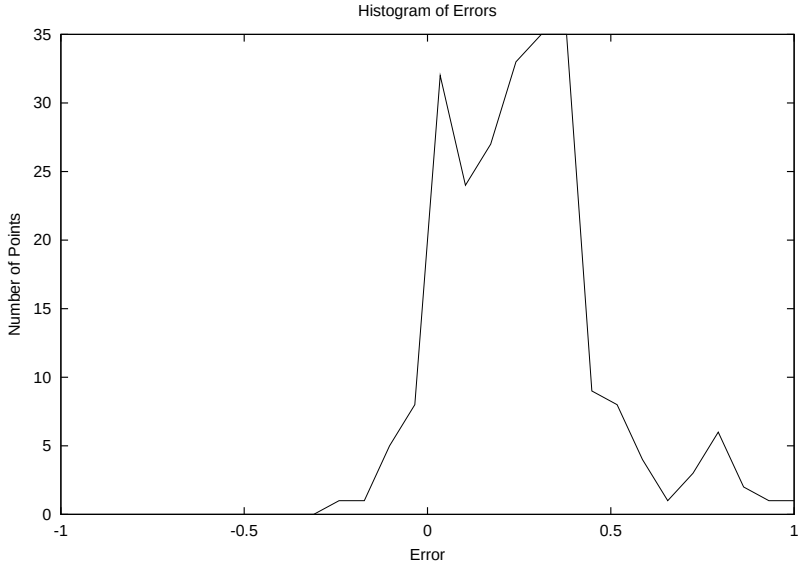

Figure 5.40: The histogram of errors for the moment training data after 300 epochs.

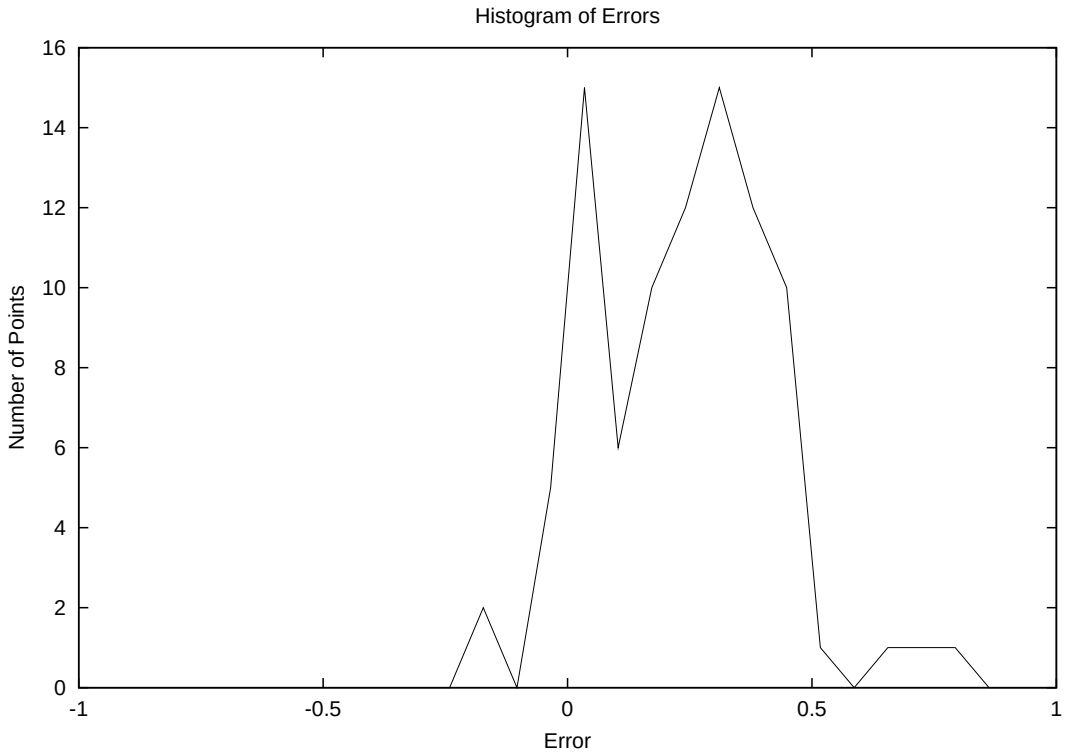

Figure 5.41: The histogram of errors for the moment test data after 300 epochs.

5.5.3 Employing the Optimal Brain Surgeon

The optimal brain surgeon method (described in detail in Section 3.7.2) provides a means to remove redundancy in the neural network. It prunes weights that are of little importance in the computation of the output. Thus, it can be used as an indication of how many hidden neurons are needed for a good model.

A good model, i.e. one which provides good performance generalisation, is not necessarily easy to obtain by starting to train on a model with only the optimal amount of hidden neurons. The reason for this is that the training process might be easier with more neurons and more weights than with just the optimal amount needed for computational purposes. However, since OBS does provide an indication of how many hidden neurons are needed for model computation, it does help getting a rough idea of how many hidden neurons are needed for training purposes. If nothing else, it puts a lower boundary on the number of hidden neurons.

Since OBS has such severe computational needs, we did not prune all the resultant networks. When we did apply OBS, we found that it removed many of the weights while maintaining the generalisation performance. However, we seldom found OBS to increase generalisation performance.

5.5.4 Discussion

The results presented in the previous section leaves us with a few questions. Which network architecture should be used? Does FIRNN provide any more strength than TDNN? Why is the moment time series more difficult to model, and what are acceptable P_d and P_f values?

We discuss the question of FIRNN versus TDNN first. We found that it is easier to train TDNNs than FIRNNs for both time series. Comparing the two predictions of the same data sets for the energy time series, we observe that the FIRNN seems to be less prone to make false alarms than the TDNNs, but the P_d for the FIRNN was smaller. We observe the same behaviour for the moment data, where the P_f of the FIRNN is less than half that of the TDNN. Thus, the challenge from the FIRNN point of view is to increase the P_d without an accompanying increase in P_f, where as the

challenge from the TDNN point of view is to decrease the P_f while maintaining good P_d values.

One could ask exactly what good P_d and P_f values are. Values of 0.75 for P_f sound bad. However, when big events are few in number, then a relatively high P_f could be tolerable. For example, let there be one big event in four months. For a P_d of .8, and a P_f of .75, one prediction will be made each month, and the true event would have a 80% chance of being detected. If the costs saved from detecting the big event is more than the money lost from false alarms, having models with this accuracy will still be useful.

One last question remains: the question of why modelling seismic energy seems to be easier than modelling seismic moment. The Lyapunov exponents hinted at this behaviour. The biggest Lyapunov exponent for the energy time series is about 1.3, and that of the moment time series about 1.8. However, the biggest Lyapunov exponent is only a measure of the period during which valid predictions can be made. The reason might lie in the inherit properties of the seismic moment, which although correlated with seismic energy, is a different physical property of the seismic event. Another factor inhibiting the accurate prediction of large seismic moment events is the relatively sparsity of big seismic moments, as opposed to the more frequently occurring bigger seismic energy events.

The statistic of one death per day in South African mines, which are of the most modern in the world, gives an indication of just how difficult it is to predict seismic events of size. In this chapter we presented a method for extracting neural network training and test data for raw seismic data. We discussed the application of the methods from Chapters 2 and 3 to seismic time series. We also discussed performance measures suitable for the evaluation of seismic event predictors. Finally, we showed some of our best results. We were able to extract models capable of tracking the dynamics of the data. Given the high-dimensionality of the attractor, and the large noise component present in the time series, the neural networks obtained results that were beyond expectation. The prediction of (only) very big events, especially events with a big moment, were not as good as we hoped for, but could have been much worse. This study indicates that, pertaining to seismic event prediction, neural networks are definitely worthy of further investigation.

Chapter 6

Conclusions and Directions for Future Research

6.1 Accomplishments and Open Problems

The overall objective of this study was to investigate the use of neural networks to predict seismic event series. In achieving this goal we followed a specific route. We first characterise the time series, obtaining information about the underlying attractor. We then extract parameters useful for time series prediction. We then proceed to model and predict the time series, using neural networks. We first demonstrated the validity of this path by showing its effectiveness when applied to well-known systems. Then we applied this methodology to the seismic time series. We first showed how to preprocess the raw seismic data into a useful time series. We then showed that this time series is chaotic. We showed that standard methods such as false nearest neighbours and false strands have difficulty extracting embedding dimensions for these time series. We continued to show results when predicting the seismic time series using TDNNs and FIRNNs. We showed that the networks were able to model and predict both the energy and the moment time series.

This being said, we certainly do not claim to have solved all the worlds problems when it comes to

predicting disasters in mines. Several problems still remain. We found that predicting larger events — which is the ultimate goal from the mine engineering perspective — is much more difficult than predicting smaller events. Furthermore, models that do have a reasonable probability of detecting large events, almost always had unacceptably high probability of false alarms.

Another problem yet to be solved is how to improve the results of a model beyond a certain point. We found that a model capable of large event prediction, looses this ability with further training, aimed at reducing the probability of false alarms. Our largest concern is the lack of good training data — i.e. highly populated data with acceptable noise levels. Noise results from two sources, accuracy of measurements and irrelevant data. Irrelevant measurements are measurements from unpredictable sources, for example blasting, and explosions such as methane gas explosions that do not occur due to seismic buildup. The latter problem may be addressed by marking, and removing irrelevant data from the time series. The first concern may also be addressed in the near future by advances made in seismic monitoring technology. Recently, a new kind of seismic transducer, called the accelerometers, have been developed. These devices are capable of much more accurate measurements than those currently used in mines — those from which we extracted our time series. When data from these devices, spanning an adequate period of time, becomes available, neural networks may well be able to obtain results advancing the current state of the art in seismic hazard prediction.

6.2 Information from Knowledge Discovery

Neural networks are well suited for many modelling purposes, but unfortunately it is not obvious what they have learnt. Thus, no new information is gained by obtaining a model for the time series, and it is also not possible to verify the model. This ”black box” characteristic often dissuades domain experts from using neural network models. However, methods for extracting information from neural networks exist. One such method, called TREPAN, can extract decision trees that approximate the concept represented by networks trained on time series data [10]. Applying tech-

niques such as TREPAN to extract understandable information from neural networks trained on seismic time series should prove interesting. It may even lead to improving current models for event prediction.

6.3 Detection of Novelties

Seismic monitoring constantly provides more data. It may be beneficial to train the existing neural networks on the new data as well. However, if the new data contains no new information, more training could result in overfitting. The flow of rock is a dynamical process, and involves various parameters that may slowly change as time progresses. Thus, we are presented with the question: when do we need to retrain the neural networks? Novelty detection poses a possible solution. Novelty detection uses the idea of negative selection found in immunology, and has been used to detect anomalities in time series [11]. These methods may be used to detect behaviour that was not present in the training data. When such behaviour is detected, it might be wise to retrain the network to include the new data.

6.4 Long Short-Term Memory Recurrent Neural Networks

We experimented only briefly with recurrent neural network architectures. Recurrent neural networks of substantial size require large amounts of computing power, and are notoriously difficult to train. However, as we pointed out in Section 3.5, TDNNs and FIRNNs have only limited context. Thus, relevant information may be lost to the network. Recurrent networks have, in principle, unlimited context. However, this comes at the price of difficulty during the training process. The reason is that most recurrent neural networks find it extremely hard to learn long-term dependencies. However, long short-term memory neural networks claim to be able to cope with long-term dependencies without losing short-term accuracy [23]. Thus, an investigation of long short-term memory for seismic event prediction is warranted.

Appendix A

Error Backpropagation

A.1 Error Backpropagation for Feedforward Networks

The derivation for the feedforward error backpropagation algorithm can be obtained from references such as [21]. Here, we only summarise the algorithm for weight updates.

The algorithm consists of two passes; a forward and a backward pass. During the forward pass, input is applied to the input neurons, and the result is propagated through the network, until it reaches the output. During the backward pass, the weights are adapted to fit the desired response better. During the forward pass, the net input to each neuron j at time step n is computed as follows:

$$v_j(n) = \sum_i w_{ji}(n) y_i(n) \tag{A.1}$$

where weight w_{ji} connects the output of neuron i, y_i, to neuron j. The error made at the output neuron is given by

$$e_j(n) = d_j(n) - y_j(n) \tag{A.2}$$

where d_j is the desired output for output neuron j. During the backward pass the local error signals

for each neuron are computed:

$$\delta_j = \begin{cases} e_j(n)\phi'(v_j(n)) & \text{for neuron } j \text{ in the output layer} \\ \phi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) & \text{for neuron } j \text{ in a hidden layer} \end{cases} \tag{A.3}$$

where $\phi'(x)$ is the derivative of the activation function with respect to its input, and weight w_{kj} connects the output of neuron j to neuron k in the next layer. The update to weight w_{ji} is then given by

$$w_{ji}(n+1) = w_{ji}(n) + \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{A.4}$$

where α is the momentum constant and η the learning rate parameter. The activation function is often chosen to be sigmoidal:

$$\phi(x) = \frac{1}{1+e^{-x}} \tag{A.5}$$

For the sigmoidal case, the derivative can be shown to be computable from only the output y_j:

$$\phi'(x) = y_j(1-y_j) \tag{A.6}$$

A.2 Temporal Error Backpropagation

The temporal error backpropagation algorithm also consists of a forward and a backward phase. During the forward pass, the input is propagated through the network until the output for each neuron is computed. As discussed in Section 3.4, each weight in the feedforward network makes place for a weight vector in the FIRNN such that

$$v_j(n) = \sum_{i=1}^{m_0} \vec{w}_{ji}^T \vec{x}_i(n) + b_j \tag{A.7}$$

$$y_j(n) = \phi(v_j(n)) \tag{A.8}$$

where $\vec{x}_i(n)$ is the state vector, $\vec{w}_{ji}$ the weight vector or synapse connected to the output of neuron j, b_j the externally applied bias, and m_0 the number of inputs in the previous layer.

The error e_j at the output neuron is again the difference between the desired response and the network output. For the temporal error backpropagation algorithm, the state vector for each synapse must be stored. For neuron j in the output layer, the local gradient is computed as

$$\delta_j(n) = e_j(n)\phi'_j(n) \tag{A.9}$$

and the weight updates computed as

$$\vec{w}_{ji}(n+1) = \vec{w}_{ji} + \eta\delta_j(n)\vec{x}_i(n) \tag{A.10}$$

where $\vec{x}_i(n)$ is the state vector at time n. The gradient for neuron j in a hidden layer is computed as

$$\delta_j(n-lp) = \phi'(v_j(n-lp)) \sum_{r\in A} \vec{\Delta}_r^T(n-lp)\vec{w}_{rj} \tag{A.11}$$

In this equation, l is the depth of the hidden layer, with $l = 0$ the output layer. The set A for a neuron j is the set of all neurons whose inputs are fed by the output of neuron j. The order of each synaptic FIR filter is designated by p, and the vector $\vec{\Delta}(n-p)$ stores previous local gradients and is given by

$$\vec{\Delta}_r(n-p) = [\delta_r(n-p), \delta_r(n+1-p), ..., \delta_r(n)]^T \tag{A.12}$$

Finally, the weight updates are computed by

$$\vec{w}_{ji}(n+1) = \vec{w}_{ji} + \eta\delta_j(n-lp)\vec{x}_i(n-lp) \tag{A.13}$$

where η is the learning rate.

Bibliography

[1] S. Addy, "Neural net processed seismic facies maps and their enhancement using attributes and inversion data." http://www.denvergeo.org/lunch299.htm.

[2] G. Backus and M. Mulcahy, "Moment tensor and other phenomenological descriptions of seismic sources. I. continuous displacements," *Geophysics, J.R. Astronomical Society*, vol. 47, pp. 301–329, 1976.

[3] R. Bakker, J. Schouten, C. Giles, F. Takens, and C. van den Bleek, "Learning chaotic attractors by neural networks," *Neural Computation*, vol. 12, no. 10, 2000.

[4] A. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, pp. 930–945, 1993.

[5] M. Boker, T. Schreiber, B. Pompe, and B. Bertenthal, "Nonlinear analysis of perceptual-motor coupling in the development of postural control," in *Nonlinear Techniques in Physiological Time Series Analysis* (H. H. G. Meyer-Kress and G. Kurths, eds.), Berlin: Springer-Verlag, 1997.

[6] G. Bolt, "Investigating fault tolerance in artificial neural networks," Tech. Rep. YCS 154, University of York, Heslington, York, England, 1991.

[7] D. Broomhead and G. King, "Extracting qualitative dynamics from experimental data," *Physica D*, vol. 20, pp. 217–236, 1986.

[8] Brunner, Emerson, Ferguson, and Suddarth, *Textbook of Medical Nursing*. New York: Lippincoff, 1970.

[9] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. New York: John Wiley and Sons, 1993.

[10] M. Craven and J. Shavlik, "Understanding time-series networks: A case study in rule extraction," *International Journal of Neural Systems*, vol. 8, no. 4, pp. 373–384, 1997.

[11] D. Dasgupta and S. Forrest, "Novelty detection in time series data using ideas from immunology," *Neural Information Processing Systems (NIPS) Conference*, 1995.

[12] C. Diks, W. Zwet, F. Takens, and J. DeGoede, "Detecting differences between delay vector distributions," *Physical Review E*, vol. 53, no. 3, pp. 2169–2176, 1996.

[13] M. Ding, C. Grebogi, E. Ott, T. Sauer, and J. Yorke, "Estimating correlation dimension from a chaotic time series: When does plateau onset occur?," *Physica D*, vol. 69, pp. 404–424, 1993.

[14] W. Ditto, M. Spano, H. Savage, S. Rauseo, J. Heagy, and E. Ott, "Experimental observation of a strange nonchaotic attractor," *Physical Review Letters*, vol. 65, no. 5, pp. 533–536, 1990.

[15] R. Drossu and Z. Obradović, "Datamining techniques for designing neural network time series predictors," in *Knowledge-Based Neurocomputing* (I. Cloete and J. Zurada, eds.), pp. 325–367, London, UK: MIT Press, 2000.

[16] J. Eckmann, S. O. Kamphorst, D. Ruelle, and S. Ciliberto, "Liapunov exponents from time series," *Physical Review A*, vol. 34, no. 6, pp. 4971–4979, 1986.

[17] J. Farmer, E. Ott, and J. Yorke, "The dimension of chaotic attractors," *Physica D*, vol. 7, pp. 153–180, 1983.

[18] A. Fraser and H. Swinney, "Independent coordinates for strange attractors from mutual information," *Physical Review A*, vol. 33, no. 2, pp. 1134–1140, 1986.

[19] P. Grassberger and I. Procaccia, "Characterization of strange attractors," *Physical Review Letters*, vol. 50, no. 5, pp. 346–349, 1983.

[20] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," *Advances in Neural Information Processing Systems*, vol. 5, pp. 164–171, 1993.

[21] S. Hayken, *Neural Networks, A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice-Hall, 1999.

[22] H. Herzel, J. Holzfuss, Z. Kowalik, B. Pompe, and R. Reuter, "Detecting bifurcations in voice signals," in *Nonlinear Techniques in Physiological Time Series Analysis* (H. H. G. Meyer-Kress and G. Kurths, eds.), Berlin: Springer-Verlag, 1997.

[23] S. Hochreiter and J. Schmidhuber, "Long short term memory," Tech. Rep. FKI-207-95, Technische Universität München, 80290 München, Germany, 1995.

[24] D. Hoyer, R. Bauer, B. Pompe, M. Palus, J. Zebrowski, M. Rosenblum, and U. Zwiener, "Nonlinear analysis of the cardiorespiratory coordination in a newborn piglet," in *Nonlinear Techniques in Physiological Time Series Analysis* (H. H. G. Meyer-Kress and G. Kurths, eds.), Berlin: Springer-Verlag, 1997.

[25] U. Hübner, N. Abraham, and C. Weiss, "Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared NH_3 laser," *Physical Review A*, vol. 40, no. 11, pp. 6354–6365, 1989.

[26] L. Iasemidis and J. Sackellares, "The evolution with time of the spatial distribution of the largest Lyapunov exponent on the human epileptic cortex," in *Measuring Chaos in the Human Brain* (D. Duke and W. Pritchard, eds.), Singapore: World Scientific, 1991.

[27] M. Ishikawa, "Structural learning and rule discovery," in *Knowledge-Based Neurocomputing* (I. Cloete and J. Zurada, eds.), pp. 153–206, London, UK: MIT Press, 2000.

[28] R. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.

[29] K. Jim, C. Giles, and B. Horne, "An analysis of noise in recurrent neural networks: Convergence and generalization," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 996–1006, 1996.

[30] M. Kennel and H. Abarbanel, "False neighbors and false strands: A reliable minimum embedding dimension algorithm," tech. rep., Institute for Nonlinear Science and Department of Physics, University of California, San Diego, Mail Code 0402, La Jolla, CA 92093-0402, 1994.

[31] M. Kennel, R. Brown, and H. Abarbanel, "Determining embedding dimension for phase space reconstruction using the method of false nearest neighbors," tech. rep., Institute for Nonlinear Science and Department of Physics, University of California, San Diego, Mail Code R-002, La Jolla, CA 92093-0402, 1992.

[32] A. Krogh and J. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems 4* (J. Moody, S. Hanson, and R. Lippmann, eds.), pp. 950–957, San Mateo, CA: Morgan Kaufmann Publishers, 1992.

[33] D. Kugiumtzis, "State space reconstruction parameters in the analysis of chaotic time series - the role of the time window length," *Physica D*, vol. 95, pp. 13–28, 1996.

[34] S. Lawrence, C. Giles, and A. Tsoi, "What size neural network gives optimal generalization?," Tech. Rep. UMIACS-TR-22, Institute for Advanced Computer Studies, University of Maryland, College Park, MD, 1996.

[35] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), pp. 396–404, San Mateo, CA: Morgan Kaufmann, 1990.

[36] D. Lerner, "Monitoring changing dynamics with correlation integrals: Case study of an epileptic seizure," *Physica D*, vol. 97, pp. 563–576, 1996.

[37] W. Liebert and H. Schuster, "Proper choice of the time delay for the analysis of chaotic time series," *Physical Letters A*, vol. 143, no. 2, pp. 107–111, 1989.

[38] R. Lynch. Integrated Seismic Systems International, Stellenbosch, South Africa, *Personal Communication*.

[39] M. Nørgaard and O. Ravn and N.K. Poulsen, *Neural Networks for Modelling and Control of Dynamical Systems*. London, UK: Springer-Verlag, 2000.

[40] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, p. 287, 1977.

[41] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[42] A. Mendecki, *Seismic Monitoring in Mines*. London, UK: Chapman and Hall, 1997.

[43] J. Minnix, "Fault tolerance of the backpropagation neural network trained on noisy inputs," *International Joint Conference on Neural Networks*, vol. 1, pp. 847–852, 1992.

[44] R. Moddemeijer, "A statistic to estimate the variance of the histogram based mutual information estimator based on dependent pairs of observations," *Signal Processing*, vol. 75, no. 1, pp. 51–63, 1999.

[45] M. Moreira and E. Fiesler, "Neural networks with adaptive learning rate and momentum terms," Tech. Rep. 95-04, Institut Dalle Molle D'Intelligence Artificielle Perceptive, Case Postale 609, 1920 Martigny, Valais, Suisse, 1995.

[46] A. Murray and P. Edwards, "Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training," *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 792–802, 1994.

[47] K. Ogata, *Modern Control Engineering*. Upper Saddle River, New Jersey: Prentice Hall, 1997.

[48] E. Ott, C. Grebogi, and J. Yorke, "Controlling chaos," *Physical Review Letters*, vol. 64, no. 11, pp. 1196–1199, 1990.

[49] L. Pecora and T. Carrol, "Synchronization in chaotic systems.," *Physical Review Letters*, vol. 64, pp. 904–907, 1990.

[50] L. Prechelt, "Connection pruning with static and adaptive pruning schedules," *Neurocomputing*, vol. 16, pp. 49–61, 1997.

[51] J. Proakis and D. Monolakis, *Digital Signal Processing*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[52] K. Pyragas, "Transmission of signals via synchronization of chaotic time-delay systems," *International Journal of Bifurcation and Chaos*, vol. 8, no. 9, pp. 1839–1842, 1998.

[53] C. Reyl, L.Flepp, R. Badii, and E. Brun, "Control of NMR-laser chaos in high-dimensional embedding space," *Physical Review E*, vol. 47, no. 1, pp. 267–272, 1993.

[54] O. Rössler, "An equation for continuous chaos," *Physics Letters*, vol. 57A, no. 5, pp. 397–398, 1976.

[55] D. Ruelle, "Deterministic chaos: The science and the fiction," *Proceedings of the Royal Society*, vol. A 427, pp. 241–248, 1990.

[56] D. Rumelhart, G. Hinton, and R. Williams, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*. Cambridge, MA: MIT Press, 1986.

[57] I. Sandberg and L. Xu, "Uniform approximation of multidimensional myopic maps," *IEEE Transactions on Circuits and Signals*, vol. 44, pp. 477–485, 1997.

[58] T. Sauer, J. Yorke, and M. Casdagli, "Embedology," *Journal of Statistical Physics*, vol. 65, no. 3/4, pp. 579–615, 1991.

[59] Y. Shimshoni and N. Intrator, "Classifying seismic signals by integrating ensembles of neural networks," *Proceedings of ICONIP'96. Progress in Neural Information Processing*, vol. 1, pp. 84–90, 1996.

[60] H. Siegelmann, B. Horne, and C. Giles, "Computational capabilities of recurrent NARX neural networks," Tech. Rep. CS-TR-3408, University of Maryland, College Park, MD, 1995.

[61] H. Siegelmann and E. Sontag, "Turing computability with neural nets," *Applied Mathematics Letters*, vol. 4, no. 6, pp. 77–80, 1991.

[62] C. Sparrow, *The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors*. New York: Springer-Verlag, 1982.

[63] A. Stahlberger and M. Riedmiller, "Fast network pruning and feature extraction using the unit-OBS algorithm," in *Advances in Neural Information Processing Systems 9* (M. J. M. Mozer and T. Petsche, eds.), pp. 655–661, Cambridge: MIT Press, 1997.

[64] J. Theiler, "On the evidence for low-dimensional chaos in an epileptic electroencephalogram," *Physical Letters A*, vol. 196, pp. 335–341, 1995.

[65] J. Thomsen, *Vibrations and Stability*. London: McGraw-Hill, 1997.

[66] G. Towell and J. Shavlik, "Knowledge-based artificial neural networks," *Artificial Intelligence*, vol. 70, no. 1,2, pp. 119–160, 1994.

[67] N. Tufillaro, T. Abbott, and J. Reilly, *An Experimental Approach to Nonlinear Dynamics and Chaos*. Reading, MA: Addison-Wesley, 1992.

[68] P. van de Laar, T. Heskes, and S. Gielen, "Partial retraining: A new approach to input relevance determination," *Proceedings of ICANN'97*, 1997.

[69] E. Wan, “Time series prediction by using a connectionist network with internal delay lines,” in *Time Series Prediction. Forecasting the Future and Understanding the Past* (A. Weigend and N. Gerhenfeld, eds.), Reading, MA: Addison-Wesley, 1994.

[70] A. Wolf, J. Swift, H. Swinney, and J. Vastano, “Determining Lyapunov exponents from a time series,” *Physica D*, vol. 16, pp. 285–317, 1985.