Artificial Neural Network Modelling Of Flood Prediction And Early Warning

BY

Ramapulana Nkoana

This dissertation is presented in partial fulfillment of the requirements for the

Master's Degree In Disaster Management

in the

Faculty of Natural and Agricultural Science

DimTec

at the

University of the Free State
BLOEMFONTEIN

Supervisor: Dr Anwar Vahed

May 2011

Declaration of Authorship

I, Ramapulana Nkoana, declare that this thesis titled, *Artificial Neural Network Modelling of Flood Prediction and Early Warning* and the work presented in it are my own. I confirm that:

- This work was done mainly while in candidature for a research degree at this university.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself, jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________

Name: ______________________

Date: ______________________

Abstract

The risk of flood is affected by factors such as land use, meteorological events, hydrology and the topology of the land. In South Africa, the geography of apartheid, as another example, is also a key factor affecting flood vulnerability since it influenced the geographic location of population sectors. Poverty encouraged settlement patterns that neglected flood risk in certain areas, especially in rural areas. The Msundusi area is one such region which has been vulnerable to flood disasters. This research explores the use of artificial neural network models to predict the onset of floods. Rainfall is considered as the primary factor influencing the likelihood of flood, and a number of artificial neural network architectures were evaluated as flood prediction models.

The mean percentage accuracy and correlation coefficient were used to evaluate the performance of trained neural networks. Training simulation results indicate that a feed-forward model with six input neuronal units, four hidden units and one output unit produced the best predictive results for this type of neural network. However, recurrent network models are shown to perform better than feed-forward models. The best flood prediction result was obtained for the recurrent Elman network, with a mean prediction percentage of 58.8%.

While the results obtained are not as good as some published results, those techniques involved a number of predictive variables, and in some cases required in-depth knowledge of the characteristics (for example hydrological, topographic or land use properties) of the region and physics of these processes. A key advantage of this approach is that no such knowledge is required. Moreover, the method used is parsimonious in that only a single predictive variable, namely precipitation is needed to model flood prediction. Under these conditions, this approach produced acceptable results for the neural network architectures that were studied.

Acknowledgements

I wish to express my deep gratitude to the following persons and departments:

- Dr Anwar Vahed, my supervisor and the research group leader in the CSIR Meraka Institute, for his constructive criticism, unfailing guidance and patience whilst carrying out this research project.

- I gratefully acknowledge the assistance of my lecturer, Mr Andries Jordaan, the Department Head at the Disaster Management Training and Education Centre (DiMTEC).

- My sincere thanks to my colleagues at the University of the Free State, DiMTEC office personnel, Miss Olivia Kunguma, Mrs Alice Ncube and Mrs Annelene Schröder-Groenewald.

- I would also like to acknowledge the help I received from the Council for Scientific and Industrial Research, especially Mr Laurie Bawell and Martie Van Deventer.

- The national Department of Science and Technology and the South African Weather Services for the data and support.

- My deepest gratitude to my parents (Mr L.S and Mrs M.M Nkoana), my brothers (Aubner, Ketjatje, Malegodi, Legaloga and Tshwaledi).

- Dr K.M Thobye and colleagues for their patience, kindness and understanding during my graduate studies.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

List of Figures viii

Introduction 11

1.1 Effects of Floods 11

1.1.1 Floods in South Africa 12

1.2 Flood Forecasting Models 13

1.2.1 Study area 14

1.3 Aims of Study 16

1.3.1 Problem statement 16

1.3.2 Significance of study 17

1.4 Research Objectives 17

1.4.1 Sub-objectives 18

1.4.2 Research questions 18

Flood Prediction Using Neural Networks 20

2. 1 Flooding and Flood Prediction 20

2.1.1 Factors affecting floods 20

2.1.2 Flood prediction techniques 22

2.2 Linear Models for Flood Prediction 23

2.2.1 Linear stochastic auto-regressive moving-average models 23

2.3 Non-Linear Models for Flood Prediction 24

2.3.1 K-Nearest neighbour method 24

2.3.2 Artificial neural networks 26

2.3.3 ANN development 27

2.3.4 Models of artificial neural network 28

2.3.5 Neural network architectures 31

2.3.6 Advantages and disadvantages of neural network models 38

Neural Networks and 39

Experimental Setup 39

3.1 Introduction 39
3.2 Neural Network Learning 41
3.2.1 Neural network topology 42
3.3 Neural Network Training Software 44
3.3.1 Training simulators 45
3.3.2 Neural network training 47
3.3.3 Learning rate 50
3.3.4 Momentum 50
3.3.5 Epoch size 51
3.3.6 Error function 52
3.4 Neural Network Time Series Modelling 53
3.5 Data Preparation 54
3.5.1 Data set generation 54
3.5.2 Data pre-processing 56
Results:Experimental Results 61
4.1 Introduction 61
4.2 Model design 63
4.2.1 Data normalization 63
4.2.2 Model parameters 64
4.3 Elman and Feed-Forward Networks Architecture 65
4.3.1 Feed-forward network architecture 66
4.3.2 Recurrent networks 66
4.4 Training Procedure and Pattern Set 67
4.4.1 Pattern set composition 69
4.5 Experimental Performance 71
4.5.1 Performance of neural network 71
4.5.2 Overall performance 74
Analysis and Discussion 81
5.1 Hypotheses of the Study 81
5.1.1 Methodology 81
5.1.2 Pre-processing and Post-processing 82
5.2 Findings 83
5.2.1 Feed-forward neural network performance 83
5.2.2 Elman recurrent network performance 84
5.3 Comparison of Models 85

5.3.1 Feed-Forward model comparison 86
5.3.2 Elman model comparison 86
5.4 Overall Model Performance 86
Conclusion and Extension 88
6.1 Conclusion 88
6.2 Extensions 89
Bibliography 90

List of Figures

FIGURE 1-1 MAP OF SOUTH AFRICA SHOWING THE MSUNDUSI REGION, WHICH IS THE AREA OF STUDY. ACCESSED AT HTTP://WWW.CSIR.CO.ZA/NRE/ 15

FIGURE 1-2 MAP OF MSUNDUSI CATCHMENT SHOWING DETAILED OF THE STUDY AREA. FROM HTTP://WWW.CSIR.CO.ZA/NRE/ 15

FIGURE 2-1EXAMPLE OF K-NN CLASSIFICATION [WIKIPEDIA, 2007]........ 25

FIGURE 2-2 NEURAL NETWORK LAYERS........ 30

FIGURE 2-3 SCHEMATIC REPRESENTATION OF A NEURAL NETWORK UNIT........ 31

FIGURE 2-4FEED FORWARD NEURAL NETWORK 33

FIGURE 2-5 A SIMPLE RECURRENT NETWORK OR ELMAN NETWORK........ 34

FIGURE 2-6 COMMON NON-LINEAR FUNCTIONS USED FOR ACTIVATION [NEUROAI2007, 2007]........ 35

FIGURE 2-7 SCHEMATIC REPRESENTATIONS OF THE TRANSFORMATION INSIDE A NEURON UNIT 36

FIGURE 3-1 THE GRAPHIC INTERFACE OF THE JAVANNS WORKSPACE........ 46

FIGURE 3-2 THE JAVANNS CONTROL PANEL, ERROR GRAPH AND NETWORK CREATION WINDOWS. THE CREATE LAYERS DIALOGUE BOX ALLOWS ONE TO SPECIFY THE NETWORK UNITS AND TOPOLOGY. 47

FIGURE 3-3 EDITS UNITS WINDOW OF JAVANNS ALLOWS ONE TO SPECIFY PARAMETERS FOR EACH CREATED NEURAL NETWORK UNIT........ 48

FIGURE 3-4 CONNECTIONS BETWEEN NEURAL NETWORK UNITS SHOWN IN THE CREATE NETWORK WINDOW OF JAVANNS........ 49

FIGURE 3-5 SHOWS THE STEEP ERROR SLOPE 51

FIGURE 3-6 JAVANNS CONTROL PANEL INDICATING THE LEARNING, PRUNING, PATTERNS, MOMENTUM, LEARNING FUNCTION AND CYCLES PARAMETERS........ 53

FIGURE 3-7 JAVANNS FILE FORMAT 56

FIGURE 4-1 A SIGMOID WITH A RANGE BETWEEN 0 TO +1 63

FIGURE 4-2 TARGET OUTPUT RANGE OF 0 TO 1 DIVIDED INTO TWO CATEGORIES 71

FIGURE 4-3 TARGET OUTPUT RANGE OF 0 TO 1 DIVIDED INTO THREE CATEGORIES 72

List of Acronyms

ACC	:	Accuracy
ANN	:	Artificial Neural Network
ARMA	:	Autoregressive Moving Average
ARIMA	:	Autoregressive Integrated Moving Average
CSIR	:	Councillor for Scientific and Industrial Research
DPLG	:	Department of Provincial and Local Government
DST	:	Department of Science and Technology
FFN	:	Feed-Forward Neural Network
FN	:	False Negative
FPR	:	False Positive Rate
GIS	:	Geographical Information System
HAS	:	Hydrometer Forecasters
ICT	:	Information and Communication Technology
JavaNNS	:	Java Neural Network Simulator
JRE	:	Java Runtime Environment
KNN	:	K-Nearest Neighbour
MPL	:	Multilayer Perceptron
MSE	:	Mean Square Error
NRE	:	Natural Resources and Environment
QFF	:	Quantitative Flood Forecasting
RNN	:	Recurrent Neural Network
ROC	:	Receiver Operator Characteristic
SAWS	:	South African Weather Service

SE	:	Simulation Experiment
SNNS	:	Stuttgart Neural Network Simulator
TP	:	True Positive
TPR	:	True Positive Rate

Chapter 1

Introduction

Summary

This chapter introduces the problem of flood prediction, the proposed research, the research methodology and the expected output of this research as presented further in this thesis. The research entails the exploration of suitable artificial neural network architectures that can be used to model flood prediction, given prior daily rainfall data for a specific region. The chapter also introduces the concepts and effects of flood as background to the area of study, the problem statement, research objectives and questions to be addressed.

1.1 Effects of Floods

Natural disasters such as flood and tropical cyclones are regarded to be caused by extreme weather conditions as well as changes in global and regional climate. South Africa and other countries are faced with environmental and ecological challenges particularly in view of the impact of climate change. These include the occurrence of natural disasters such as fire, floods, tropical storms, major accidents, drought, epidemic diseases and food shortage [DPLG, 2007].

Floods are the most frequent natural hazards globally [Verdin, 2002], and the hazard of flooding can be divided into primary, secondary and tertiary effects. The primary effects of floods are those due to direct contact with the flood waters, with the water velocities resulting in floods as the discharge velocity increases. Secondary effects, such as disruption of infrastructure and services and health impacts, are primary effects, while tertiary effects are viewed as the long-term changes that occur, for example changes in the position of river channels [Nelson, 2010].

Previous years' flooding have been the most costly disasters in terms of property and human casualties. These floods cause great losses and damage that also have devastating socio-economic, hydrological and climatic tertiary effects [Varoonchotikul, 2003]. For example the flooding of the river Arno in Italy caused serious damage valued at more than 15.5 billion euro [Campolo et al., 2003]. The river crosses the city of Florence as it leaves the mountain region. In the United States between 1989 and 1999, floods caused the loss of at least 988 lives and about 4.5 billion US dollars in economic losses [Barros and Kim, 2001]. Many parts of Europe experienced dramatic summer floods in 1938, 1966, 1981, 1997 and 1998, which placed their economic and environmental situation at risk [Barros and Kim, 2001].

1.1.1 Floods in South Africa

In South Africa, a total of 946 hazardous events were recorded during the period between 1800 and 1995. Of these, the occurrence of floods was the highest on the list of such hazardous events [DPLG, 2007]. In the early 2000, areas of the southern region of Africa, especially the Mozambican coast and northern parts of South Africa were hit by a series of tropical cyclones, namely Astride, Gloria, Hudah and Eline. Cyclone Eline was considered to be the most severe cyclone which caused landfalls in parts of Mozambique and saturated soil in the Limpopo basin [Verdin, 2002]. This type of disaster was described as the most devastating flood disaster experienced in Southern Africa [Verdin, 2002]. Many lives were lost and more than ten million people had no access to potable water. Thousands of people in Mozambique and approximately 200 in South Africa were relocated to refugee camps. The infrastructure such as roads, bridges and buildings damaged within South Africa and Mozambique, were estimated to cost more than one billion rand to repair [Alexander, 2002].

The provinces most frequently affected by floods in South Africa are regarded to be the coastal provinces such as the Western Cape, KwaZulu-Natal and the Eastern Cape [DPLG, 2007]. From historical records, the KwaZulu-Natal province has experienced a number of flood events over the past decades [Kjeldsen et al., 2001] and appears to be more vulnerable to heavy rains, storms and cyclones that cause flash floods. These events affect the economy and lives of the communities, especially in the rural areas of the Msundusi municipality in KwaZulu-Natal. Different studies and models such as statistical analyses for flood magnitude frequency have been undertaken but none of them have accurately predicted the occurrences of floods [Alexander, 2002]. Regional frequency analysis including the use of L-moments together with the index-flood method was applied successfully in a number of case studies from the USA, Australia and the Kwazulu-Natal province in South Africa, for modelling floods [Kjeldsen et al., 2001].

The floods that occurred in the year 2000 over the Southern African region caught South Africa and Mozambique in a state of unpreparedness. In 1995 severe flooding took place in the Msundusi catchment which caused loss of life of around 160 people. Infrastructural damage amounted to R20 million and at least 586 families lost their homes. These flood disasters emphasise the importance of accurate flood prediction methods for South Africa and neighbouring countries. The implementation and improvement of hydrological models in prediction will assist in reducing the impact of flood to human and economic losses in South Africa and other countries [Verdin, 2002, Campolo et al., 2003]. However, hydrological models are often complex and computationally intensive to run, requiring expertise in environmental hydrology and accurate data for many geographical, environmental and meteorological parameters [Barros and Kim, 2001].

1.2 Flood Forecasting Models

Early warning flood systems can be implemented in order to provide effective warning for natural disasters that can be caused by floods. This can be accomplished by the combination of technologies such as Geographical Information Systems (GIS), remote sensing, and Information and Communications Technologies (ICT) that translate data into useful information and make this information accessible to the role players and communities at risk [DPLG, 2007]. Floods cannot be prevented, but damage can be reduced by proper planning. Flood prediction is a complex process influenced by geographic location, rainfall, soil type and size of catchments that affect river water levels. Models such as Quantitative Flood Forecasting (QFF) and Artificial Neural Networks (ANN) have been developed and implemented in different locations to help in weather forecasting over the past decades [Barros and Kim, 2001]. Some of the existing data sources used in flood modelling are:

- Radar information Systems
- Stream and rain-gauge networks and hydrograph analysis
- Linear statistical models
- Non-linear time series analysis and prediction.

This research thesis focuses on a non-linear time series analysis and prediction technique. The non-linear time series regression model is used throughout this research is the Artificial Neural Network (ANN).

1.2.1 Study area

The Msundusi Municipality is located in KwaZulu-Natal province of South Africa within the Umgungundlovu District Municipality. The Msundusi local municipality is located approximately 75 kilometres from the urban centre of Durban and serves a population of over half a million people in Pietermaritzburg and surrounding areas. Figure 1.1 shows the geographical location of this region in South Africa.

The legacy of apartheid is still felt in the uneven development between the urban centres and poorly developed townships and surrounding rural settlements. The Municipality include the Msundusi River catchment, as indicated by Figure 1.2, Pietermaritzburg city and the areas Ashburton, Vulindlela and Claridge. The Msundusi River flow catchment is 540km^2 covering the entire municipality including the rural, peri-urban areas and the city of Pietermaritzburg [Emanuel, 2009].

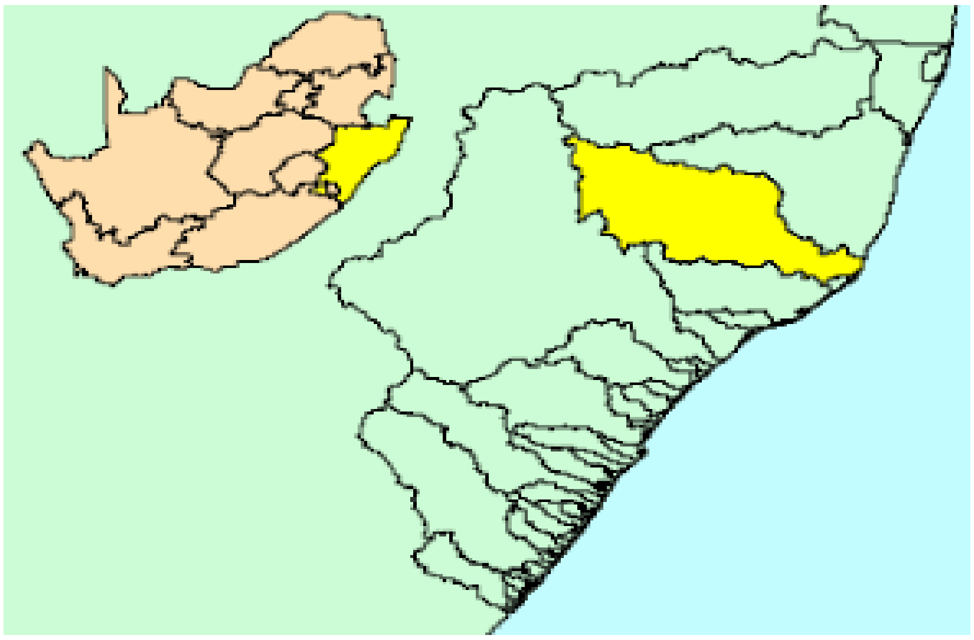


Figure 1-1 Map of South Africa showing the Msundusi region, which is the area of study. Accessed at http://www.csir.co.za/nre/

The Msundusi River is the major river draining the catchment region, and it flows in a west-east direction through the municipality. Parts of its passage through the city have been canalised in order to improve drainage capacity as well as to reduce flooding. This region has been identified for the investigation conducted in this study.

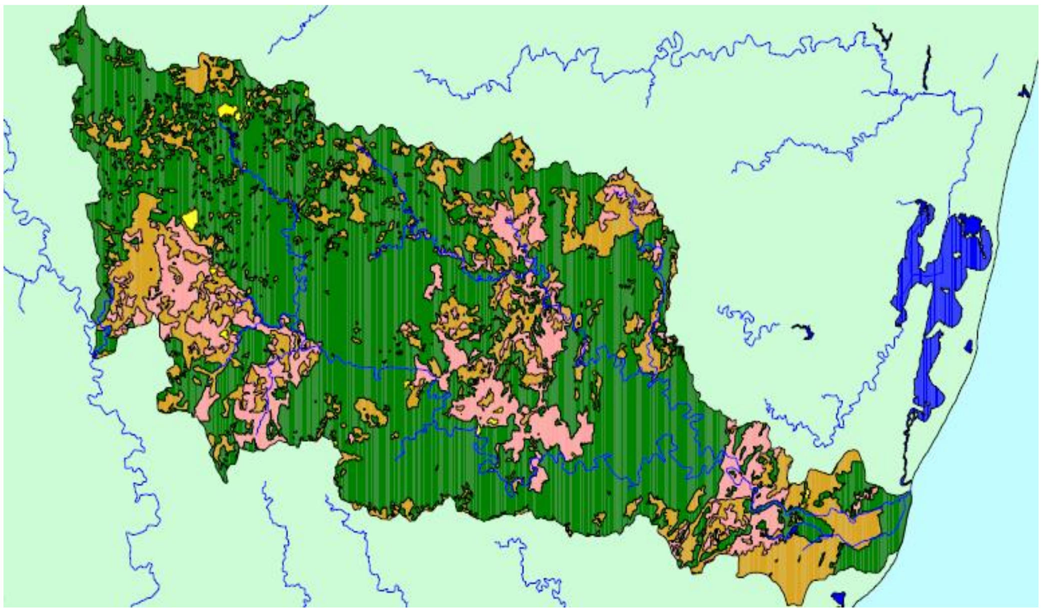


Figure 1-2 Map of Msundusi catchment showing detailed of the study area. From http://www.csir.co.za/nre/

1.3 Aims of Study

This research work entails the research and development of models for improving the prediction of floods in areas which are frequently at risk. We study the application of artificial neural networks, a non-linear auto-regressive machine learning technique trained on patterns of preceding rainfall values in order to predict the likelihood of a flood.

1.3.1 Problem statement

An increase in the risk of a flood hazard can be caused by several factors, including land use changes such as deforestation and rapid urbanization. Demographic pressures also cause the encroachment of informal settlements on hazardous locations in flood plains. Numerous other factors are likely to be the root causes of flood disasters. In South Africa these factors include the geography of apartheid, which led to increased and differentiated trends in flood vulnerability. These trends together with poverty in informal settlements, especially in rural areas, have encouraged settlement patterns that inhabit flood risk areas such as flood plains and areas of poor vegetation cover. KwaZulu-Natal is vulnerable to weather-related natural disaster such as floods affecting the rural and urban areas. A great number, about 4 200 people, in the Msundusi local municipality are living next to the river catchment [IDP, 2002].

The water quality is also very poor due to flash-flooding and bacterial contamination such as e.coli, which poses a serious health risk to the inhabitants. Consequently, the majority of people who are settling in flood plain areas have been severely affected by flood disasters. One of the difficulties is that there are no known accurate flood prediction methods for the Msundusi municipality. The problem is compounded by the lack of communication and effective information transfer in the event of flooding; hence, previous floods caused lot of environmental, economic and social damage.

1.3.1.1 Scope of study

This study does not cover flash floods, because of the resolution of the data needed for rainfall over small areas. It does not cover flooding due to very extreme weather events such as tropical storms and hurricanes, since the region that is studied is not prone to these types of events.

This research is applied to one region, as proof of concept, which is Msundusi area in KwaZulu Natal, but the results intended to obtain may be useful more generally for flood predictions. The research is primarily focused on the use of artificial neural networks trained on historical rainfall values to predict subsequent rainfall values. The advantage of the proposed method is that it requires very few variables and very little knowledge, if any, of related concepts such as hydrology to model the dynamics of flooding.

1.3.2 Significance of study

A flood prediction model can play a key role in providing relevant information of possible impending floods in populated locations. The development of such models can reduce the damage in areas such as the Msundusi municipality by decreasing the economic and environmental impacts of floods. More importantly, a prediction system developed for South Africa, especially the Msundusi area, can effectively lower the risk of harm and loss of life. If artificial neural network (ANN) models can provide sufficiently accurate forecasts, even one day ahead, the lead time for flood warning can be extended and the subsequent flood emergency measures can be better planned and executed.

1.4 Research Objectives

The main objective of this study is to research and develop artificial neural networks that can be used as model to predict the onset of floods in a region such as the Msundusi River catchment. Several types of artificial neural network model are studied, including their architectures and variations of associated learning rules to determine the neural network parameters that will provide the best prediction for impending floods. A number of sub-goals have been identified toward this end.

1.4.1 Sub-objectives

The sub-goals of this research work are:

- To pre-process historical weather data into a form that is suitable for training neural networks and to identify the salient variables for training neural networks.
- To show that artificial neural networks can be used as a valid and effective approach to predict floods from meteorological data.
- To determine the architecture of neural network that will yield the best predictive performance for precipitation.

1.4.2 Research questions

The research questions to be addressed in this study are:

- What is the most effective neural network architecture for predicting floods?
- Which variables (meteorological such as precipitation, wind speed and direction, humidity, hydrological and geographic variables) can be used to most effectively predict impending floods?
- How should these variables be represented in order to derive accurate predictive patterns for an impending flood?
- How can neural networks be trained to learn predictive patterns for floods?

Chapter 2 presents a review of the current literature of flooding, showing previous work done and introduces the general theory of artificial neural networks (ANN).

Chapter 3 the neural network architecture and model data are described, the methodology used in the research, the experimental method and the software used.

Chapter 4 presents preliminary results obtained from training simulations.

Chapter 5 presents the results, an analysis and a discussion of the results obtained from data of a weather station within the Msundusi municipality.

Chapter 6 presents a summary and conclusions of this research as well as recommendations for future research.

Chapter 2

Flood Prediction Using Neural Networks

Summary

This chapter presents the technical concept of flooding and describes findings in literature related to flood prediction. Current flood prediction techniques are presented along with a discussion of their relative advantages and disadvantages. The concept, techniques, structure, types, model description and use of artificial neural networks for flood forecasting are also discussed in this chapter.

2. 1 Flooding and Flood Prediction

Wallingford [Wallingford, 2008] defines flood as a temporary covering by water of land not covered by water. This includes floods from rivers, mountain torrents and floods from the sea in coastal areas. Another definition of flood refers to excessive water run-off or the rise in water level in a particular area which is more than the particular environment can absorb [Nelson, 2009]. Floods are caused either by cloud bursts, continuous rain in the same area, landslides, storm winds or the excessive release of water from dams. Rainfall intensity and duration are the key elements which contribute to flooding. There are different types of floods such as riverine flooding, coastal floods, urban floods and flash floods; the type and factors affecting them are described further in this chapter.

2.1.1 Factors affecting floods

Flooding is influenced not only by meteorological factors, but also by hydrological factors such as terrain slope, land use, vegetation, soil types, soil moisture, as well as hydrological processes related to run-off channels subject to flooding. Various combinations of rainfall intensity and duration may lead to flooding, depending on the hydrologic and hydraulic factors of a watershed [DPLG, 2008, Smith, 2001]. Combinations of these factors can give rise to various types of flooding events, some which are described below:

Riverine flooding: this type of flooding occurs along rivers, usually seasonally during summer in the northern parts of South Africa, and during winter in southern parts of South Africa. Rainfall causes river basins to fill with too much water, too quickly. Torrential rains from tropical cyclones can also produce river flooding as was experienced in the 2000 floods in Mozambique and the northern parts of South Africa.

Coastal floods: this type of flooding can be produced by sea waves called tsunamis which were the case in the 2004 East-Asia tsunami. It normally occurs when seismic sea-bed disruptions or winds generated from tropical storms and cyclones, drive ocean water inland and cause significant flooding.

Urban floods: this type of flooding is caused by urbanisation due to the decrease in run-off from two to six times more than what would occur on natural terrain. In other words, buildings, paved areas and roads reduce run-off capacity. Poor or blocked drainage infrastructure cannot cope with the sudden increase in precipitation, usually resulting in vastly increased volumes of water concentrated in small areas. This normally causes the streets to turn into swift-moving rivers, and buildings can be damaged during urban flooding [DPLG, 2008].

Flash floods: flash flood events are usually characterized by sudden bursts of excessive rainfall events such as thunderstorms, hail storms and hurricanes. More insight into the flash flood dynamics may be obtained from one-dimensional and two-dimensional hydraulic models [Borga et al., 2008]. The monitoring of flash flood events gives the opportunity to observe how catchments respond when most of the surface and subsurface hydrologic flow paths are active. However, flash flood events are difficult to monitor because of the limited spatial extent and time scales over which they develop. Borga [Borga et al., 2008] noted that flash flood monitoring requires rainfall estimates of small spatial scale (1km) and short-time scales (15-30 minutes and even less in urban areas). Therefore, these events are generally better observed by weather radar networks. The rapid rate of urbanisation and consequent residential development in urban centres in South Africa over recent years have caused existing drainage systems to be inadequate to cope with excessive precipitation and hence increased the likelihood of flash flooding in urban areas.

The occurrence of flooding is determined by weather, hydrology, topography, run-off and urban infrastructure such as roads and buildings. Flood forecasting provides a basis for warning and to inform decision-makers and those in the path of floods in order to minimize flood damages, which are normally measured in economic terms. Flood damage refers to all varieties of detrimental effects caused by flooding. Flood damage effects are categorised into direct or primary flood damage, which include damage linked to the immediate physical contact of flood water to humans, property and the environment. Indirect or secondary flood damage occurs as a further consequence of the flood, and the disruptions of economic and social activities [Messner and Meyer, 2005, McCarthy et al., 2007]. Considering the types of floods mentioned, it is clear that all of them impact on economic development and are dangerous to human beings.

2.1.2 Flood prediction techniques

Three main approaches have been used for flood risk prediction, namely, statistical techniques, flood modelling and mapping, and the monitoring of water and ice levels [Sparks et al., 1998]. Statistical techniques have been used to determine the likelihood, frequency and intensity of water discharges causing flooding [Hazarika et al., 1979]. Models and maps can be used to determine and visualise the extent of possible flooding, abnormal amounts of rainfall and sudden large water discharges that can be monitored to provide short-term flood predictions [Alho, 2009].

The use of the Quantitative Precipitation Forecasting model (QPF) in flood forecasting plays an important role, allowing for extension of the lead-time for the river flow forecast, which enables timelier implementation of flood control measures. QPFs were used within hydrologic forecast models to simulate impact on rivers throughout the United States. QPFs produced by the Hydromet Forecasters (HAS) routinely includes 24 hours of forecast rainfall (QPF) throughout the year.

A reliable QPF is not an easy task to obtain due to rainfall being one of the most difficult elements of the hydrological cycle to forecast. Much uncertainty still affects the performance of rainfall prediction models [Reynolds, 2003]. However, numerical weather prediction models such as the timely use of remote sensing observations (for example radar data and satellite images) allows the issue of short-term forecasts [Xue et al., 2000].

Although the output from satellite and radar images provides useful information on precipitation patterns they do not usually provide a satisfactory assessment of rain intensities. Radar detection is difficult in mountainous regions because of elevation and altitude effects [Toth et al., 2000]. Radar imagery forecasting techniques, on the other hand, show higher accuracy than model forecasts within six to seven hours of the time of the radar image.

The QPF model can also be obtained by means of time series analysis techniques. Some examples of time-series analysis techniques which are discussed further, are linear modelling, for example linear stochastic auto-regressive moving-average models (ARMA) and non-linear modelling such as artificial neural networks (ANN) and K-nearest-neighbour (K-NN) methods.

2.2 Linear Models for Flood Prediction

2.2.1 Linear stochastic auto-regressive moving-average models

Linear stochastic processes are among the most widely used time-series technique for modelling future water resources [Toth et al., 2002]. A stochastic process is a process with random outcomes in probability theory, and can be viewed as the counterpart to a deterministic process. A stochastic process considers many probable outcomes instead of dealing with only one possible reality of how the process might evolve over time. In a stochastic process, even if the initial condition is known, there are many possible paths that the process might follow, but some paths may be more probable and others less so [Toth et al., 2002, Papoulis and Unnikrishna, 2001]. In terms of rainfall prediction, these processes express the future rainfall as a linear function of historical data.

The Autoregressive Integrated Moving Average (ARIMA) model is frequently used as predictive model, and simple Autoregressive Moving Average (ARMA) models are referred to as Box-Jenkins models. Most of the time-series techniques traditionally used for modelling floods fall within the framework of the ARMA class of linear stochastic processes [Toth et al., 2002].

In general, the ARMA model is a tool for understanding and predicting future values in a given time series of data, X (t). ARMA models describe each observation of a time-series X (t) as a weighted sum of *p* of previous data, and the current as well as q previous values [Toth et al., 2002]. The notation AR(p) refers to the autoregressive model of order *p*. The AR(p) model is described by Equation 2.1, where ε_t is a variable for white noise, c is a constant value and $\phi_1 \ldots, \phi_p$ are parameters of the model.

$$X_t = c + \sum_{i=1}^{P} \varphi_i X_{t-i} + \epsilon_t \qquad (2.1)$$

The application of low-order ARMA processes to model short-term precipitation values is normally used in flood prediction [Lee, 1996].

2.3 Non-Linear Models for Flood Prediction

This section focuses on non-linear time series analysis and prediction techniques for floods. The K-Nearest Neighbour (K-NN) and Artificial Neural Network models are presented with indications of the differences discussed in some detail.

2.3.1 K-Nearest neighbour method

The nearest neighbour algorithm is a statistical technique used to classify a value of a variable according to the closest training examples in a feature space[1]. The K-nearest

[1] A feature space is an abstract space with as many dimensions (axes) as there are features. For example, classes for the three features of temperature, rainfall and humidity can be defined in a 3-dimensional feature space.

neighbour (k-NN) algorithm considers the *k* examples nearest to the point that has to be classified. Originally a pattern recognition procedure, the algorithm was subsequently extended to time-series and forecasting problems [Karlsson and Yakowits, 1987].

The k-NN prediction technique uses *k* historical data of measurements to predict the value of an outcome variable of a sample based on its 'closeness' to the *k* previous measurements [Baoli et al., 2003]. Usually the Euclidean ('straight line') distance is used as a measure of closeness, as illustrated by Figure 2.1.

The prediction of a time series is based on a local approximation, making use of only the nearby observations. For each forecast instant *t*, a d-dimensional feature vector is defined as the vector (or tuple) of the past *k* observations of the variable *x*. The method thus assumes to be able to summarise statistically, the entire past of the forecast instant *t* [Toth et al., 2000, Toth et al., 2002]. In the case of rainfall prediction, the k-NN algorithm considers all consecutive d-dimensional vectors in the historical rainfall dataset and locates *k* of these vectors, which are closest to the vector of the *d* most recent rainfalls. In this method, the prediction of the next rainfall is then taken to be the average of the rainfall values subsequent to this *k* historical nearest neighbours.

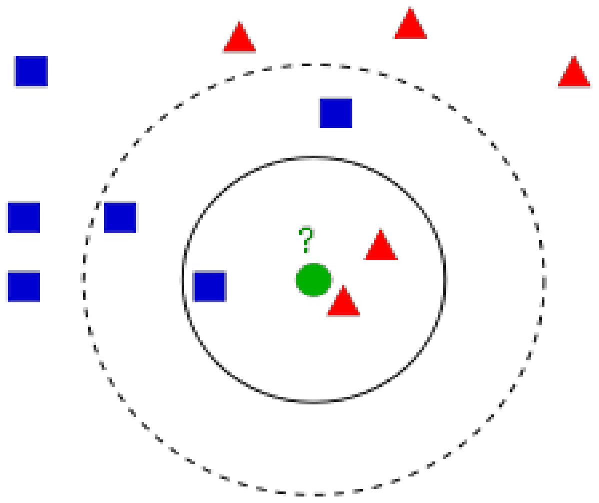


Figure 2-1Example of k-NN classification [Wikipedia, 2007].

The k-NN algorithm generally achieves good performance for different data sets. Hydrology researchers have successfully applied the k-NN method in hydrological field problems [Baoli et al., 2003]. The advantage of k-NN is that the technique does not require the selection of a class of models and the estimation of the models parameters, so that the identification of a specific form of the input-output relationship is not needed [Toth et al., 2000]. Hence this method does not attempt to identify an input/output mapping function and thus is not capable of extrapolating an unfamiliar input vector into the future. In contrast, other non-linear models such as ANNs, attempt to identify a mapping function from input to output and have extrapolation ability.

2.3.2 Artificial neural networks

The artificial neural network (ANN) is an example of a non-linear prediction (NLP) method, which have been extensively studied and applied to a variety of problems, including meteorological simulation and forecasting [Varoonchotikul, 2003]. Tingsanchali [Tingsanchali, 2009] and Campolo et al.[Campolo et al., 2003] have conducted several ANN-based studies directed at the prediction of river flows at a time scale ranging from one year to one day in the Chao Phraya River and Arno River in Italy, with only the use of past flow observations.

The large majority of the ANN hydrologic applications predict future flows based on the knowledge of previous rainfall values along with past observed flows. Results from artificial neural network experimental approaches in daily discharge forecasting show that statistical NPL methods provide more accurate forecasts over a shorter prediction period such as one to six hours, while the ANN method provides more accurate results over prediction periods exceeding 24 hours [Damle, 2003].

The use of artificial neural networks is a popular data-driven technique that has been frequently applied to a broad range of fields. An ANN is able to handle non-linearity and automatically adjusts to new information, while generally requiring little computational effort [Rietjes and de Vos, 2008]. ANNs are widely accepted as powerful ways of modelling complex non-linear and dynamical systems for which there are large amounts of sometimes noisy data [Chen et al., 2002]. ANNs were selected as the applied method for this investigation for the following reasons:

- They are well suited to time series pattern matching problems, and are efficient in terms of information storage for the trained model.

- They can deal with incomplete, noisy and ambiguous data [Haykin, 1994].

- They are pedagogic as opposed to decompositional; for example whereas decompositional methods require knowledge of the domain or the physical characteristics of the problem, a pedagogical method requires only data and does not depend on knowledge of the relationships between factors that affect the problem. Thus it is possible to train ANNs without having intimate knowledge of the hydrological or other aspects of flood forecasting.

The behaviour of a neural network is defined by the way its individual computing elements are connected and by the strength of those connections. These weighted connections are automatically adjusted during training of the network. ANNs with one hidden layer are commonly used in modelling since it has been found that a more than one hidden layer does not yield any significant improvement in performance on a network with a single hidden layer [Chen et al., 2002]. More detail on artificial neural networks is given in the following sections.

2.3.3 ANN development

The artificial neural network model was inspired by the biological nervous system and has allowed scientists and researchers to build mathematical models of neurons in order to simulate neural behaviour [Fu, 1994, Zurada, 1992]. Models of a neuron were introduced in the early 1940s by McCulloch and Pitts by which they described simple logic for neural networks, and was later credited with a learning law, the Perceptron Learning Algorithm [McCulloch and Pitts, 1990, Fu, 1994].

The research on the limits to what one layer perceptrons can compute was demonstrated by Minsky and Papert with the use of elegant mathematics [Minsky and

Papert, 1972, Haykin, 1994]. The back-propagation algorithm developed by McClelland and Rumelhart, is the most popular learning algorithm for the training of multilayer perceptrons [Rumelhart and McClelland, 1987]. In the late 80s, many neural networks research programmes were introduced as is indicated by the rapidly growing number of conferences and journals devoted to the field. Presently, neural networks are widely applied to systems such as:

- support for medical diagnosis
- financial market prediction
- voice and handwriting recognition
- flood prediction, and solar flare forecasting
- a variety of signal processing systems.

ANNs were first introduced to water resources research for their use to predict monthly water consumption and to estimate occurrences of flood. Since then, ANNs have been used for a number of different water resource applications which include time-series prediction for rainfall forecasting, rainfall-runoff processes and river salinity. ANNs have also been used for modelling soil and water table fluctuations, pesticide movement in soils, water table management and water quality management [Parson, 1999].

2.3.4 Models of artificial neural network

There is no single definition for an artificial neural network. Zurada [Zurada, 1992] defined a neural network as follows:

A neural network is simply a class of mathematical algorithms, since a net-work can be regarded essentially as a graphic notation for a large class of algorithms. Such algorithms produce solutions to a number of specific problems.

Another definition by Smith [Smith, 2001] describes a neural network as “a form of multiprocessor computer system” with:

- Simple processing elements
- A high degree of interconnection
- Simple scalar messages
- Adaptive interaction between elements.

According to Haykin [Haykin, 1994] a neural network is a massive l parallel for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

- Knowledge is acquired by the network through a learning process.
- Inter-neuron connection strengths known as synaptic weights are used to store the knowledge.

An ANN can be described as a network of simple but interconnected processing units called neurons (or neuronal units), which are able to automatically adjust to information and learn aspects of this information by storing it in the connection strengths, represented as weights between neurons as shown in Figure 2.3.

The ANN contains a large number of simple neuron-like processing elements and a large number of weighted connections between the elements. The weights of connections encode the knowledge embedded in the network. The “intelligence” of a neural network emerges from the collective behaviour of neurons, each of which performs only very limited operation. Each individual neuron finds a solution by working in parallel. The following list describes the overall tasks involved in constructing an ANN.

1. Determine the network properties or architecture: This includes the network connectivity, the types of connections, the order of the connections (if any), and the weight range values.

2. Determine the system dynamics: this entails the weight initialization method, the activation-calculating formula, and the learning rule.

The topology of a neural network is specified by the number of layers, the number of units per layer and the weighted connections among all the units. These types of layers are the Input layer, the Hidden layer (of which there may be none too many), and the Output layer [Fu, 1994] as shown in Figure 2.2. In a feed-forward network, data flows as indicated by the arrows, from the Input to the Output layer.

The Input layer receives input signals or data from the external world and a node in this layer is called an Input unit. These units represent and encode the data or signal pattern presented to the network for processing [Fu, 1994, Kumar et al., 2004].

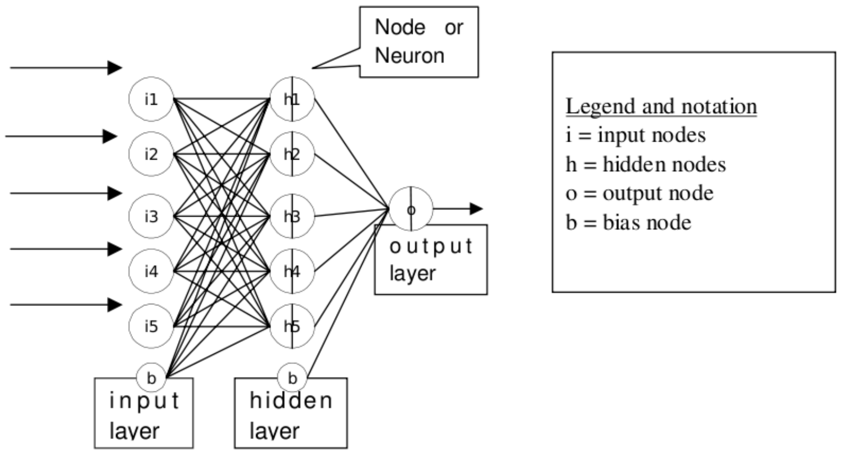


Figure 2-2 Neural network layers.

The layer following the Input layer is the Hidden layer, and the nodes in this layer are called Hidden units. The Hidden layer can consist of one or more layers of neurons with the succeeding layers receiving input from preceding layers in feed-forward architecture. The Output layer is the final layer of the network, and the nodes in this layer are called Output units. These units represent encoded concepts (or values) for the training application under consideration. The neurons present in this layer present the output of the network. Non-input neurons represent linear or non-linear combinations of the input and weights.

2.3.5 Neural network architectures

One class of ANN architecture is the feed-forward network as shown in Figure 2.4, and is discussed in detail further in this chapter. For this class of ANN, data signals always propagate in one direction from the Input layer to the Output layer and without consideration of any time delays. The other class of ANN architecture is the recurrent neural network, which contains feedback connections from units in subsequent layers to units in preceding layers as shown in Figure 2.5.

The neural unit processes the input information into the output information. Each of these units is a simplified model of a neuron and transforms its input information into a neuronal output response. This transformation involves the activation of the neuron as computed by the weighted sum of its inputs. This activation is then transformed into a response by using a transfer function. Figure 2.3 depicts an example of this process.

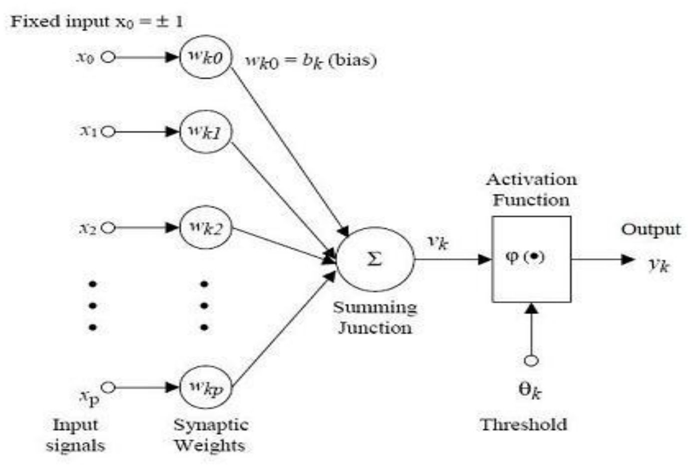


Figure 2-3 Schematic representation of a neural network unit.

The network architecture determines the number of connection weights and also the way information flows through the network. The determination of the best network architecture is one of the difficult tasks in the model building process but one of the most important step to be taken. The ANN models employed in this study are feed-forward networks and recurrent networks with hidden and special hidden layers of neurons. The following neural networks are commonly used for flood prediction.

Feed-forward networks-The initial ANN models used in this study were feed-forward networks with one hidden layer of neurons as shown in Figure 2.4. The simplest way to define feed-forward network, all connections point in one direction from the input towards the output layer. Multi-layered perceptrons are feed-forward structures with one or more layers between the input and output nodes. The advantage of multilayer perceptrons (with one or more hidden layer) is that the number of nodes in the hidden layer can be varied to adapt to the complexity of the relationships between input and output variables. One of the experimental objectives of this research was to determine the size of the hidden layer that produces the best predictive performance. Feed-forward neural networks (FFN) are found to perform best for one time-step forecasting, when applied to data for which the sampling or measuring time interval is less than or equal to time in 24 hours [Varoonchotikul, 2003].

Recurrent Neural Networks (RNNs)-These are models with bi-directional data flow. While feed-forward network propagates data from input to output, RNNs propagate data from 'downstream' processing units to earlier units. Thus RNNs, have feedback connections between units of different layers or loop type self-connections [Kumar et al., 204, Ahmad and Ismail, 2004]. This implies that the output of the network not only depends on the external inputs, but also on the state of the network in the previous time step as is shown in Figure 2.5. The model shown employs full feedback and interconnections between all nodes. There are several advantages of RNNS over FFNs, the first one being that RNNs have the capability to retain values from previous cycles of processing, which can be used in current computations. This advantage allows RNNs to produce complex, time varying outputs in response to simple static inputs.

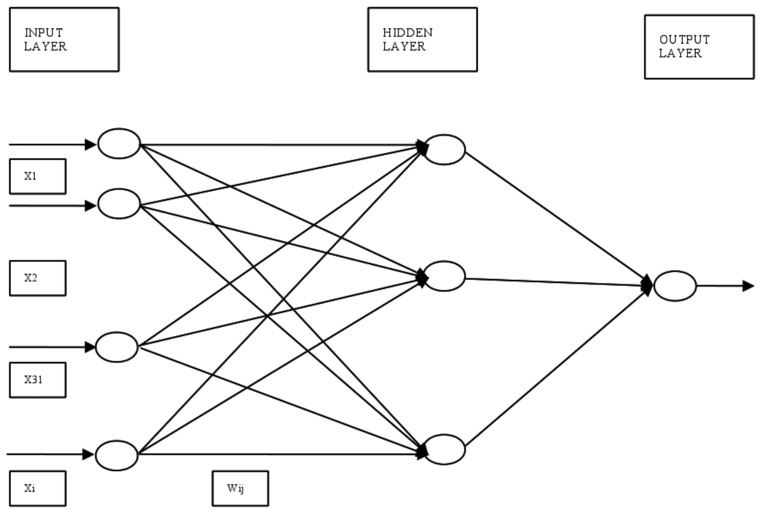


Figure 2-4 Feed Forward Neural Network

Since a RNN can have connections between units of any of the layers, the output of each unit has to be identified in terms of time steps, for example outputs at time step $t - 1$ can be inputs at time step t. It is also possible with some RNN architectures, to preserve a copy of outputs at a previous time step by means of context units. Context units retain these outputs, which can be re-used as inputs in subsequent training steps. Both feed-forward and recurrent architectures were studied as prediction models.

2.3.5.1 Activation functions

Inputs to an ANN can be binary or real valued. In order to compute the single value output for each neuron, the weighted sum of the inputs to a neuron is used in an equation called a transfer or activation function.

The input nodes of a neural network can be completely connected to the hidden units in the hidden layer and each connection has an independent weight attached to it. The term connectivity in this context refers to the extent to which a node is connected to other nodes in the adjacent layer.

A unit that is connected to all units in the preceding layer is said to be fully forward connected; a unit that is connected to all the units in a subsequent layer (as for RNNs), is said to be fully backward connected. In other cases a unit, and hence the neural network, is said to be partially connected. Connection weights may be real or integer valued.

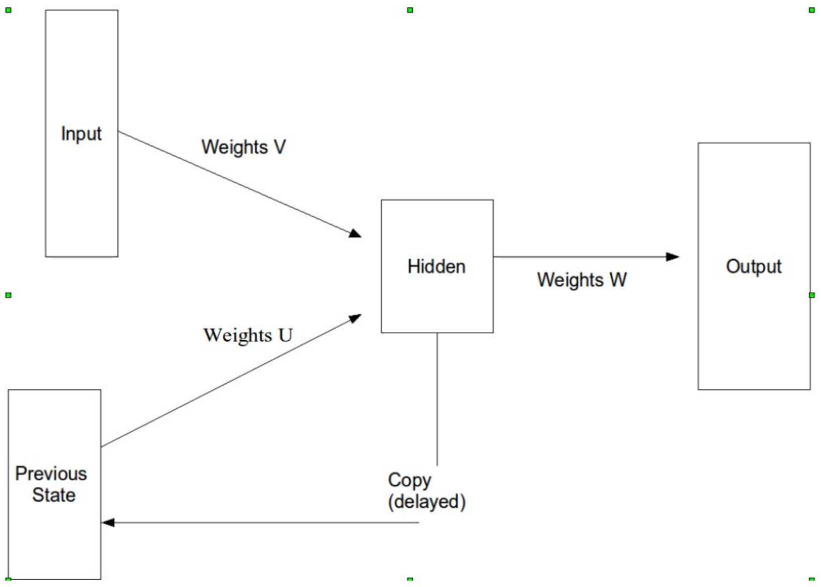


Figure 2-5 A simple recurrent network or Elman network

The numbers of hidden units are dependent upon the problem to be solved. Since there is no general process, the number of hidden units is usually determined by an iterative process of increasing or decreasing the hidden layer size during training. There are three types of common activation functions, namely the threshold, the piecewise-linear and the sigmoid function. The threshold function has a value of 0 if the summed input is less than a certain threshold value (θ), and the value 1 if the summed input is greater than or equal to the threshold value. The Piecewise-linear function can have values between 0 and 1.

A common transfer function is the sigmoid function, which can have a range of 0 to 1 or a range of -1 to 1. The hyperbolic tangent function is one example of a sigmoid function as denoted by Equation 2.2, where S is the weighted sum of the inputs of the unit.

$$\varphi(S) = \tanh\frac{S}{2} = \frac{1-\exp^{(-S)}}{1+\exp^{(-S)}} \qquad (2.2)$$

Another form of sigmoid function is given by Equation 2.3

$$\varphi(S) = \frac{1}{1+\exp^{(S)}} \qquad (2.3)$$

The common non-linear functions used for activation in the form of 'soft' (smooth) non-linearity, are (a) sigmoid and (b) tanh functions, and for abrupt, 'hard' non-linearity, the (c) signum and (d) step functions as shown in Figure 2.6 [NeuroAI2007, 2007]. This is a first-order basis function and the net value is a linear combination of the inputs in Figure 2.6

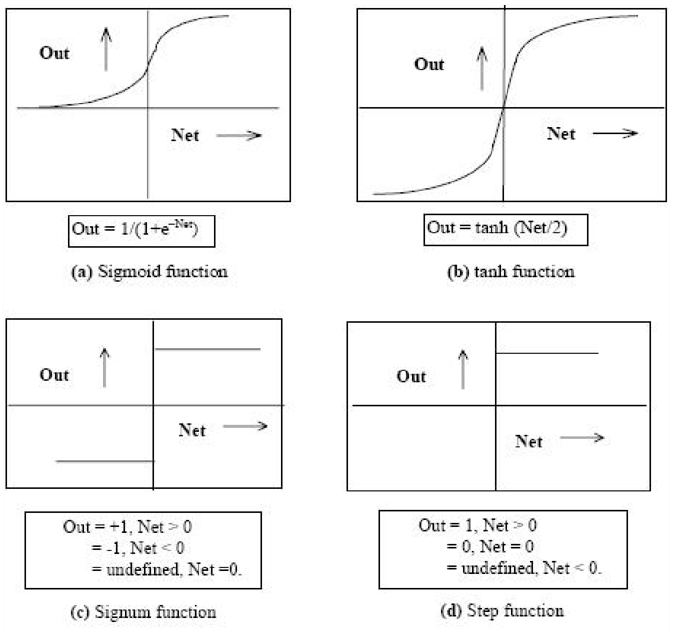


Figure 2-6 Common non-linear functions used for activation [NeuroAI2007, 2007]

A correct mapping of input to output requires determining the correct weights for the neural network. The processing that takes place in ANN is shown by Figure 2.7 where the input values from preceding neurons (x) are multiplied by the weight (w) that accompanies their connection. The results are summed and an additional value bias (b) is commonly added to this value. Thus, for *n* input units, the unit sum S for a neuronal unit is defined by equation 2.4

$$S = \sum_{i-1}^{n} . X_i W_i + b \qquad (2.4)$$

The resulting unit input is transformed by the activation value of neuron, denoted as in Figure 2.7. This activation value is propagated to subsequently connected neurons.

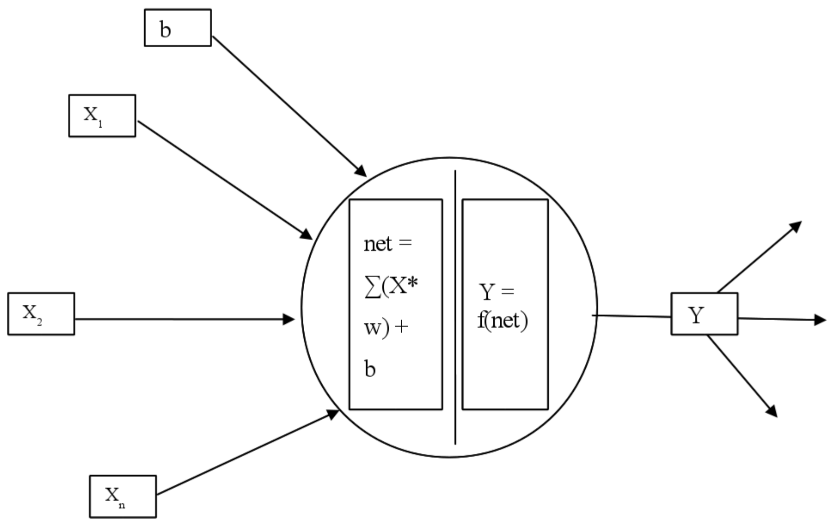


Figure 2-7 Schematic representations of the transformation inside a neuron unit

2.3.5.2 Training

The process of optimising the connection weights is known as training or learning. The network learns a function by adapting the strength of its connection weights in response to the training examples presented to it in accordance with a predefined learning law [Fan et al., 2002]. ANNs are trained by applying an optimising algorithm, which attempts to reduce the error in the network output by adjusting the matrix of network weights and the neuron biases.

A common approach to ANN training in function approximations is to use supervised training algorithms. Supervised training algorithms are used in combination with sample input and output data of the system that is to be simulated. The back propagation algorithm is regarded as the most popular ANN training algorithm. It is essentially a procedure to train feed-forward models by which the outputs are sent without a delay to the next layers. Kumar et al. [Kumar et al., 2004] consider the process of selecting a suitable architecture for a required problem as (1) fixing the architecture, (2) training the network and (3) testing the network.

For ANN flood forecasting to work well, Openshaw [Openshaw and Openshaw, 1997] suggested that extensive data of a more than 15 years period with 24 input variables are required. The variables which must be used should be long-term rainfall accumulation for 90 days, evapotranspiration of the previous day, mean daily temperature for six previous days, mean daily rainfall for six previous days, mean river level for six previous days, numbers of hours of daylight and a day-night flag. A single variable is used in this study, namely the previous daily rainfall value obtained from the South African Weather Services.

The size of the steps taken in weight space during training is a function of the number of internal network parameters and learning parameters. The learning parameters include the learning rate, momentum value, error function, epoch size and gain of the transfer function [Kumar et al., 2004, Dandy and Maier, 2000]. These parameters are discussed in some detail further in Chapter 3.

In order to optimise the performance of neural networks trained with the back propagation algorithm, it is essential to have a good understanding of the impact that step size has on training [Dandy and Maier, 2000].

The initialization of weights is the first step in preparing the internal parameters for training a neural network. For the back propagation algorithm, the weights are initialised to small zero-mean random values, and studies also suggested that a number of different sets of random starting values should be used to see whether consistent results are obtained.

Activation functions that are commonly used, are sigmoidal functions such as the logistic and hyperbolic tangent functions. In this research, sigmoidal functions were used. However, studies show that non-sigmoidal transfer functions perform best when the data is noiseless (for example contains no errors) and contain highly non-linear relationships [Parson, 1999]. Dandy [Dandy and Maier, 2000] suggested that using sigmoidal transfer functions in the hidden layers and linear transfer functions in the output layer could be advantageous when it was necessary to extrapolate beyond the range of the training data.

Another parameter which is commonly taken into consideration in neural network training algorithms such as the back propagation algorithm is the Error function, which is the function that is to be minimised during training. The mean square error

$$MSE = \frac{1}{N}\sum_{i-1}^{N}\varepsilon_i^2 ,$$

is the network error, where *i* is the error produced when the input pattern is presented to the network. The MSE is a commonly used error function and it has the advantages that (1) it can be calculated easily during training, (2) it penalises large errors and (3) it relates closely to the normal distribution.

2.3.6 Advantages and disadvantages of neural network models

The ANN learns how to relate the inputs to the outputs without being given any explicit equations when compared with K-Nearest Neighbour and ARMA method. The only real requirements for the ANN model are for sufficient data for flood modelling events, and the specification of appropriate neural network parameters values to be used. ANNs have relatively low computational demands and can easily be integrated with other techniques. They perform tasks that a linear programme cannot, and when an element of the neural network fails, it can continue without any problem due to their highly parallel nature [Openshaw and Openshaw, 1997].

One disadvantage of ANNs is that the optimal form or value of most network design parameters can differ for each application and cannot be theoretically defined in general. However, these values are commonly approximated using trial and error approaches [Kumar et al., 2004]. The other disadvantage of neural networks is that they require training to operate, and this may require much processing time for large neural networks [NeuroAI2007, 2007]. Hence the neural network as non-linear model is a promising approach compared to linear models for flood forecasting. As indicated by the results found during this research, their performance results are comparable with other prediction modelling techniques.

Chapter 3

Neural Networks and Experimental Setup

Summary

This chapter reviews the flood prediction methodologies used in previous studies, and the procedures followed in this study. Artificial Neural Networks applied to flood forecasting are reviewed in this chapter. Data sets were generated from meteorological data for the area of interest, obtained from the South African Weather Services. These data sets were pre-processed into a format that is appropriate for training, validation and testing of neural networks. Several neural network architectures were selected for training and validation. The trained networks were tested in order to assess their performance.

3.1 Introduction

Previous studies have shown that the ANN is an acceptable technique, with a good likelihood of successfully modelling rainfall forecasts [Rientjes and de Vos, 2005]. The unique approach in this research is that only one variable, considered to be the primary input variable for rainfall prediction, was considered in this study. The main reason for this approach is that it allows one to model rainfall prediction with a little amount of domain knowledge using the minimal number of variables. No detailed knowledge of the underlying physical characteristics such as hydrological, meteorological and environmental processes, and no complex mathematical computations are needed. The modelling technique is therefore entirely event-based, a sensible approach for modelling rainfall prediction, and also a key benefit of using neural networks.

The choice of rainfall as primary input variable is supported by previous research. Hung [Hung et al., 2009] for example, describes ANN models that could easily identify the patterns characterizing rainfall when using the rainy periods data as training data. Rainfall forecasts from trained artificial neural networks were more accurate than predictions from a simple persistent method that was also studied. Their ANN models, however, used a combination of meteorological parameters, namely relative humidity, air pressure, wet bulb temperature and cloudiness. The models were provided with screened input data that contained only rainy periods. Sensitivity analysis indicated that the most important input parameter in forecasting floods was rainfall, followed by the wet bulb temperature. The trained networks produced satisfactory forecasting performance for a period of one to three hours ahead. In most cases this predictive period is insufficient for the purposes of flood warning, preparation procedures and mitigation measures.

ANN were also used by French [French et al., 1992] to forecast rain intensity fields over space as well as time, for example over an area with a lead time of one hour. Input and output rainfall fields are presented to a three layer neural network as a series of learning sets. Results indicated that a neural network is capable of learning the space-time relationship model for rainfall for short-term forecasting. Comparisons with actual values indicate that in most cases this method performed well when a relatively large number of hidden nodes are used. The performance of the neural network was compared with two other methods for short-term forecasting, namely persistence and now casting.

Another investigation was conducted on the effect of input data with and without seasonal variation on the performance of ANN models, and the results indicated that ANNs have the ability to cater for irregular seasonal variation in the data with the aid of additional hidden layer nodes [Dandy and Maier, 2000]. Besides rainfall forecasting, neural networks have also been applied to model many hydrologic processes such as rainfall runoff [Abrahart, 2003, Hsu and Sorooshian, 1995, Shamseldin and Ling, 1997], stream flow [Abrahart and See, 2000, Campolo and Andreussi, 1999, Zealand et al., 1999], and groundwater management [Rogers and Dowla, 1994] as well as studies of water quality simulation [Dandy and Maier, 1996].

A similar procedure followed by Hung [Hung et al., 2009], was followed in this research with the most important difference that only the rainfall parameter was used for neural network training. Moreover, the prediction lead time was increased to one day, using the preceding six days' rainfall as input parameters. The choice of a one-day lead time is based on the reasoning that this would provide adequate time to prepare for the eventuality of flood.

3.2 Neural Network Learning

The most significant aspect of a neural network is its ability to learn from its environment, and to improve its performance through learning. A neural network learns about its environment or a dynamic system through an iterative process of adjustments applied to its weights and biases. The environment is characterised by a set of exemplars, which is typically a group of patterns of 'environmental' variables. The network becomes more "knowledgeable" about its environment after each iteration of the learning process. Like learning in human beings and animals, neural network learning is an inferred process which cannot be perceived directly, but can be assumed to have happened by observing changes in performance [Zurada, 1992].

Learning in the context of neural networks is defined as a process by which the free parameters of a neural network are adapted through a process of presenting signals from the environment in which the network is embedded[2]. The type of learning is determined by the manner in which the parameter changes take place [Haykin, 1994]. The notion of learning in a neural network, is the process of guiding the network to provide a particular output or response for a specific given input.

Learning is necessary when information about input-output relationship is unknown or incomplete a-priori. There are two different types of learning, namely unsupervised learning, which identifies or creates pattern-class information as the learning outcome. In this case, no desired or target classes are known beforehand, and thus no output information is known a-priori. The second neural network learning mode is supervised learning where a desired set of responses, outputs or classes for given inputs are known and provided during the learning process. In this case, the neural has to learn the

[2] A free parameter of a neural network is any independent variable that can be set or modified by the process of learning.

function, mapping or transformation that will produce the desired output for new inputs [Fu, 1994, Zurada, 1992]. Supervised and unsupervised can thus be distinguished as follows:

- Supervised learning algorithms utilize information of class membership for each training instance. This information allows supervised learning algorithms to detect pattern misclassification as feedback information for adapting their responses.

- Unsupervised learning algorithms use unlabelled instances. This type of learning often has less computational complexity and less accuracy than supervised learning algorithms. It can be designed to learn rapidly which makes it more practical in many high-speed, real-^{i}time environments.

Supervised training is hence used in applications where a desired output is known and where the network performance can be evaluated by comparing its output with a desired output. For flood prediction, supervised learning is used, with historical data for selected parameters such as rainfall and humidity and cloud cover typically used as inputs with the known event or non-event of flooding used as desired output for the given inputs. Unsupervised training can be used to perform some initial characterization of inputs. In this study some pre-processing is performed to transform input and output data into an appropriate format form neural network training. This pre-processing also involves some level of filtering and classification which is described in more detail in Section 3.5.

3.2.1 Neural network topology

Neural network architecture is determined by the number of layers, number of nodes (or neuron units) in each layer and the weighted connections between nodes. The topology of a neural network is defined by the nodes and their connections [Zurada, 1992]. The number of layers include the input layer, hidden layers (if any) and the output layer. Thus, a three layer network will have one hidden layer (in addition to the input and output layers)[3.]

[3] The layer counting convention used in this thesis is the commonly adopted convention, and differs from another approach where the input layer is ignored when counting the number of layers. In the

The determination of the network architecture is one of the most important steps in developing a model for a given problem. Although neural network construction has been extensively researched [Geman and Doursat, 1992, Kwok and Yeung, 1995], there is no known procedure or algorithm for this process for the general case. Two approaches have been proposed, namely constructive and destructive methods. In both constructive and destructive methods, the numbers of hidden nodes are considered.

Constructive methods start with a small number of network units, which is usually under-parameterised, and then proceed to increase the number of units during training until the performance of the network reaches a satisfactory level. Some of the most popular such methods are the Cascade Correlation Algorithm [Fahlman and Lebiere, 1991] and the Upstart Algorithm [Frean, 1991]. Destructive neural network methods, also called pruning methods, determine a suitable neural network structure by starting with a large number of units and then progressively removes some of these units during training until some or other performance criterion is met.

A constructive approach was adopted in this research, since this approach is more suitable to the problem of time series prediction and was shown to be more successful in general [Nabhan and Zomaya, 1994].

The most used neural network for prediction and forecasting applications is the feed-forward networks, whereby nodes in one layer are connected to nodes in the next layer. Feed-forward networks are the most commonly used network architecture, but both feed- forward and recurrent networks are used for time series prediction[4]. Unlike feed-forward networks, recurrent networks have the ability to ”remember” past events and they are generally regarded to be more effective for time series prediction problems. However, feed-forward architectures also perform well under certain circumstances.

Recurrent networks have the ability to cater for moving average components whereas

latter case, a neural network with one hidden layer would be described as a two-layer network.

[4] A forecasting competition is held annually (http://www.neural-forecasting-competition.com/) and the best results are presented at several symposia.

feed-forward networks cannot. However, feed-forward networks have been used almost exclusively for the prediction and forecasting of water resources variables, and they process data faster than the other models which are currently in use [Rientjes and de Vos, 2005]. While recurrent networks have potential advantages for time series applications they do not provide specific benefits over feed-forward networks for data limited by a number of time steps. For such time windows, feed-forward networks have been found to perform well in comparison with recurrent networks in many practical applications. Both feed-forward and recurrent network architectures were used with different numbers of hidden nodes for prediction experiments.

The neural network topology can be determined in two ways, namely by fixing the number of hidden nodes and by selecting the number of connection weights to each node. The relative properties of smaller and larger networks must also be considered when selecting the network. Smaller networks requires less storage and have higher processing speed during training and testing; however the error graph can be more complicated and such networks sometimes contains more local minima [Hutchins, 1995]. Larger networks tend to learn quickly in terms of the number of training cycles required and have an increased ability to avoid local minima in the error surface[5], however they require a large number of training samples in order to achieve better generalisation ability[6] [Zurada, 1992].

In this study, several numbers of hidden nodes have been explored during the training of both feed forward and recurrent networks, with each architecture providing a different performance when increasing and decreasing hidden nodes.

3.3 Neural Network Training Software

Neural network simulation software are programmes that allow one to build, train, and evaluate neural networks, and are referred to as neural network simulators. It was not the purpose of this research to develop new neural network simulation

[5] Each of the N weights and thresholds of the network is taken to be a dimension in space. The N+1th dimension is the network error. For any possible configuration of weights the error can be plotted in the N+1th dimension, forming an error surface. The main objective of the network training is to find the lowest point in this many-dimensional surface.

[6] Generalization in neural net is to have the outputs of the net approximate target values given inputs that is not in the training set.

software since many maturely developed training simulation software packages exist, both commercially and as open source packages. Commonly used applications for the purposes of research, include the Stuttgart Neural Network Simulator (SNNS), Emergent, JavaNNS and Neural Lab. Commonly used biological network simulators include Neuron, GENESIS, Nest and Brian. Some of these applications are discussed below, including the one used in this research.

3.3.1 Training simulators

The NeuroSolutions software [Bullinaria2005, 2005], Stuttgart Neural Network Simulator (SNNS) and Java Neural Network Simulator (JavaNNS) are popular neural network simulators used for research purposes. SNNS and JavaNNS are open source, for example are freely available online[7].

In this study, the JavaNNS package was used to perform the training and testing simulations for all the experiments. JavaNNS was developed at the Wilhelm Schickard-Institute for Computer Science (WSI) in Tubingen, Germany. The simulator is based on the SNNS 4.2 kernel, with a graphical user interface written in a Java set developed for it. The capabilities of JavaNNS are equal to the one of the SNNS, while the user interface has been newly designed. The advantage of JavaNNS is that it is independent of the operating system whereas SNNS was developed with primarily UNIX operating systems in mind. The installation process differs for Windows and UNIX operating systems. The Linux operating system (Ubuntu distribution, with kernel version 2.6.32-25-386) was used in this study. JavaNNS ran in a Java Runtime Environment (JRE) on Linux. JavaNNS is a jar file and can be invoked from the command prompt.

This presents a JavaNNS workspace interface which contains an empty Network window where the network is constructed as shown in Figure 3.1.

While JavaNNS holds the copyright to the JavaNNS Group, the software is distributed by the University of Tubingen and is freely available online[8].

[7] http://www.ra.cs.uni-tuebingen.de/downloads/
[8] http://www.ra.cs.uni-tuebingen.de/downloads/JavaNNS/

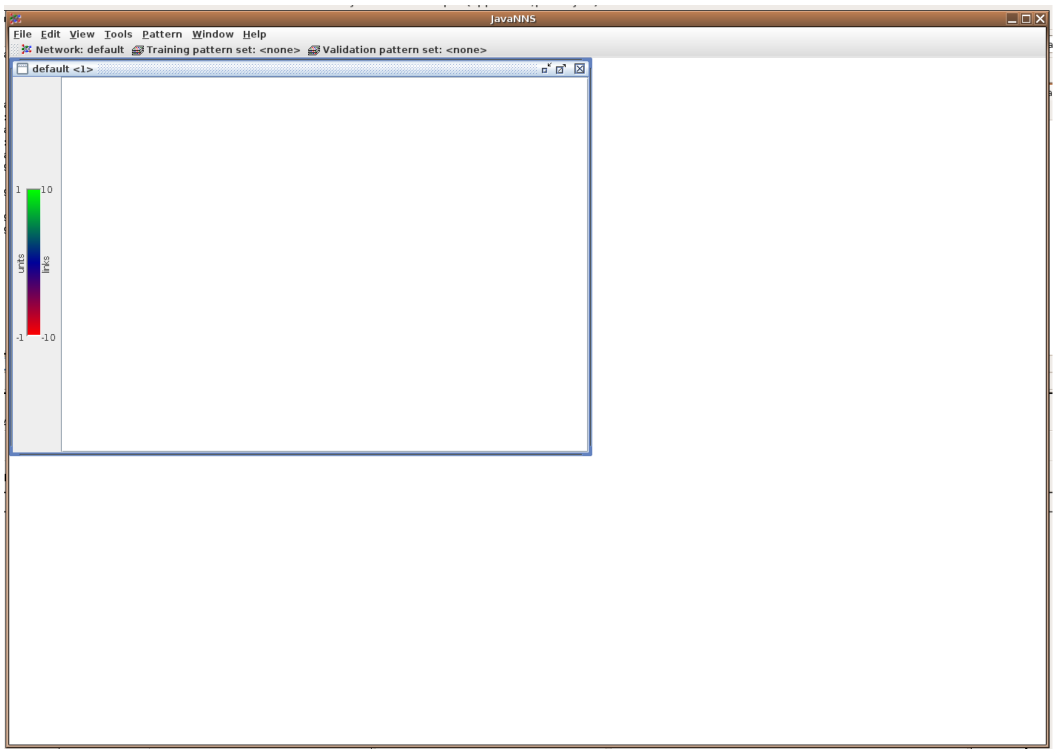
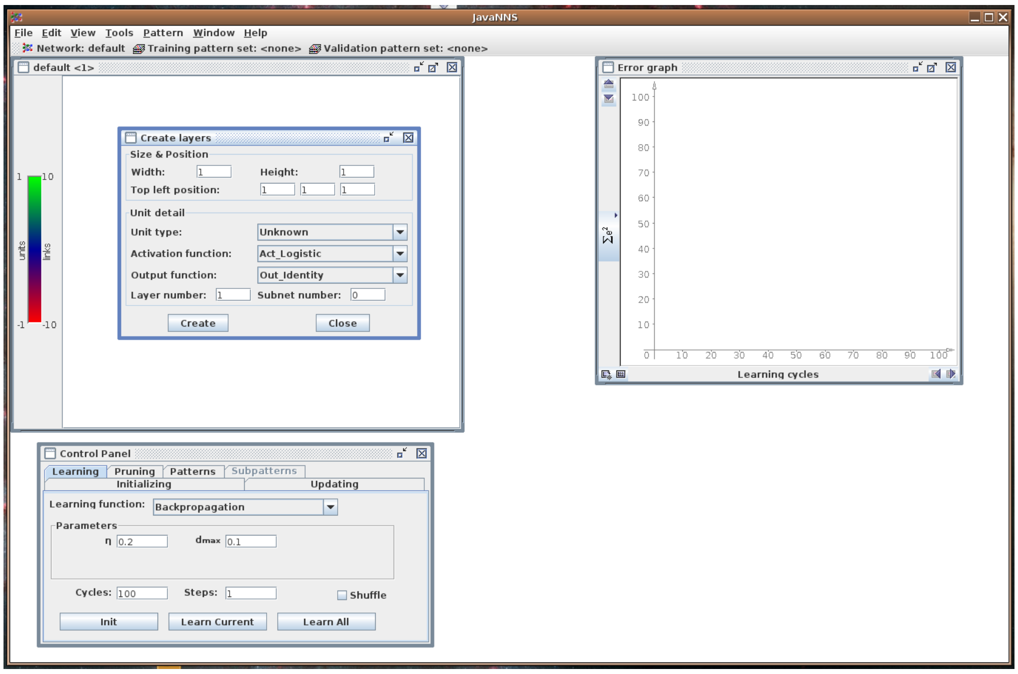

JavaNNS
File Edit View Tools Pattern Window Help
Network: default
Training pattern set: <none>
Validation pattern set: <none>
default <1>
1
10
units
links
-1
-10

Figure 3-1 The graphic interface of the JavaNNS workspace.

The units were generated by the Tools-Create-Layer menu option, which allows one to specify the selected input unit types and to create the neural network. Figure 3.2 shows the graphic interface as an example of the features presented in JavaNNS. Neural network units are represented as filled squares in the Network window and can be edited by right-clicking on them, as shown in Figure 3.3.

After units are edited, weight connections are created using the Tool-Create-Connections menu item. The colour of connection lines for the weights indicates their value according to the colour bar on the left of this window (see Figure 3.4). The network can then be saved under a desired name with the .net extension. A number of networks were created in this way, initially feed-forward architectures with the number of hidden nodes ranging from two to 12. Performance results for the trained networks are shown in Chapter 4.



Network: default Training pattern set: <none> Validation pattern set: <none>

Create layers

Size & Position
Width: 1 Height: 1
Top left position: 1 1 1

Unit detail
Unit type: Unknown
Activation function: Act_Logistic
Output function: Out_Identity
Layer number: 1 Subnet number: 0

Control Panel

Learning function: Backpropagation
Parameters
η 0.2 dmax 0.1
Cycles: 100 Steps: 1 Shuffle

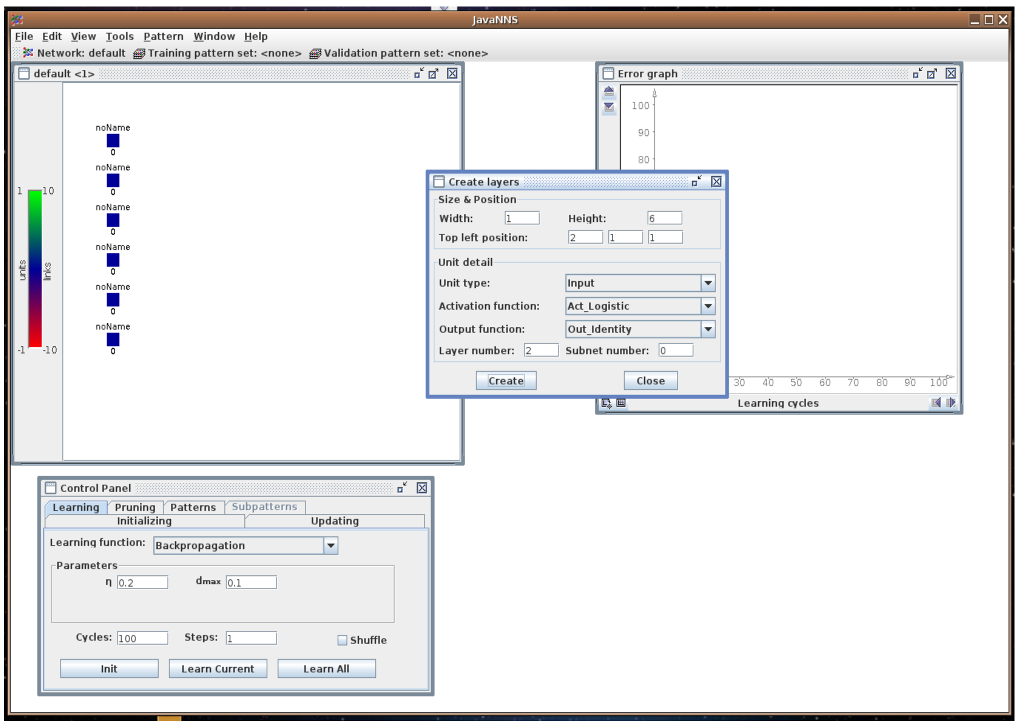
Figure 3-2 The JavaNNS Control Panel, Error Graph and Network Creation windows. The Create Layers dialogue box allows one to specify the network units and topology.

3.3.2 Neural network training

Neural network training amounts to iteratively adapting the connection weights of a neural network, until the connection weights defines an input-output function that approximates the relationship between the input and output patterns of a given training data set. In this research the network weights were adapted using the back-propagation algorithm for all neural networks considered. This error back-propagation algorithm is widely used for training feed-forward networks. For a three-layer network, the back-propagation algorithm is described by the following pseudo code fragment:

```
1.    Initialize the weights in the network
2.    S = the training set of input-output examples
3.    Repeat
4.    For each input-output pair P in S
5.    X = input pattern in P
6.    T = desired or target output pattern in P
7.    Compute Y = neural-net-output (network X)
8.    Calculate network error E = (T - Y) for the output units
9.    Compute Weight Change for all weights from hidden to output layer
10.   Compute Weight Change for all weights from input to hidden layer
11.   Update the weights in the network
12.   Until stopping criterion satisfied
```

In the pseudo code above, all the network weights are set to small non-zero random values in line 1. In line 2 a training set S, of input-output pattern pairs is read, usually from a file. From lines 3 to 12 the network weights are repeatedly adapted using the training set P, until a training stopping criterion is met.


Network: default   Training pattern set: <none>   Validation pattern set: <none>

Create layers
Size & Position
Width: 1   Height: 6
Top left position: 2 1 1
Unit detail
Unit type: Input
Activation function: Act_Logistic
Output function: Out_Identity
Layer number: 2   Subnet number: 0

Control Panel
Learning function: Backpropagation
Parameters
η 0.2   dmax 0.1
Cycles: 100   Steps: 1   Shuffle

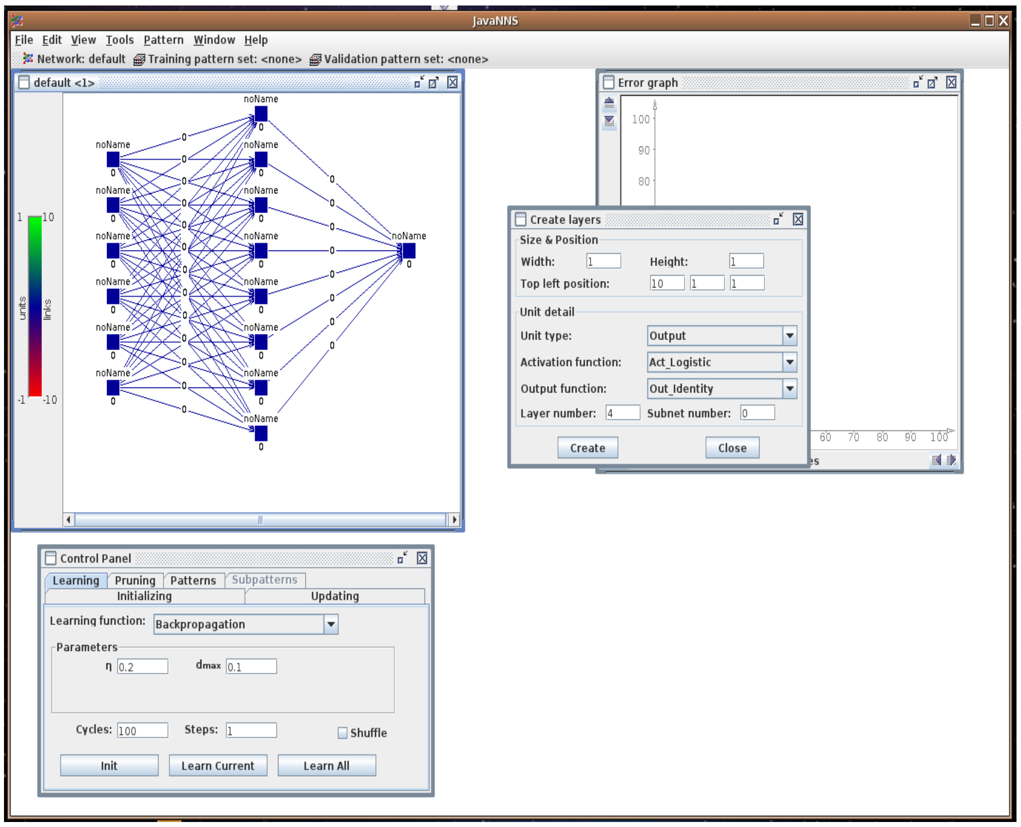
Figure 3-3 Edits units window of JavaNNS allows one to specify parameters for each created neural network unit.

Each such presentation of the training set is termed a training epoch. The stopping criterion for training can be a combination of several conditions, such as when all examples in P are classified correctly or when a certain number of epochs have elapsed, or when a predefined network output error threshold have been reached.

For each pattern pair P, the input pattern X and desired output pattern T is identified in lines 5 and 6, respectively. The network output Y is computed in line 7, and the network output error $\epsilon = (T - Y)$ is computed in line 8. In lines 9 and 10, the network weight changes are computed according to the network output error, and in line 11, the network weights are adapted according to these computed weight changes. The steps 8 to 11 form the error back-propagation process. More formally, if W (t) is the network weights at training step *t*, then the new weights at training step (t + 1) are given by Equation 3.1, where ΔW is the computed weight changes and η is the weight adaptation step size or learning rate.

$$W(t+1) = W(t) + \eta\Delta W \tag{3.1}$$

The step size, η, is a number between 0 and 1 and determines the fraction of the computed weight changes that is used to adapt the weights. The main reason for using a fractional value of the total weight change, is that the weight adaptation for a given training pair might be an over correction of the overall network error.



Network: default Training pattern set: <none> Validation pattern set: <none>

Size & Position
Width: 1 Height: 1
Top left position: 10 1 1

Unit detail
Unit type: Output
Activation function: Act_Logistic
Output function: Out_Identity
Layer number: 4 Subnet number: 0

Learning function: Backpropagation
Parameters
η 0.2 dmax 0.1
Cycles: 100 Steps: 1

Figure 3-4 Connections between neural network units shown in the Create Network window of JavaNNS.

It is important to consider the effect of the step size on training; a too small value implies a low approximation, but perhaps more focussed learning of the target function whereas a high step size might cause the network weights to be adjusted more rapidly but with greater variation.

The magnitude of the steps taken in the weight space[9] during training is a function

[9] A 'point' in weight space is that coordinate point in the Cartesian space defined by the current set of weights. The weight space is the set of all possible values of the weights.

of a number of internal network parameters including the learning rate, momentum value, error function, epoch size and the transfer function which are discussed in more detail in this section.

3.3.3 Learning rate

The learning rate is a constant used in error back-propagation learning that affects the speed of learning. The smaller the learning rate, the more steps it takes to get to the stopping criterion.

The learning rate is usually determined prior to training and while it can remain fixed during training, more sophisticated training algorithms can vary this value as training proceeds. The learning rate should ideally be decreased as training progresses, since the network weights tend to approximate the desired function more closely as training continues.

3.3.4 Momentum

The momentum parameter is used to prevent the system from converging to a local minimum[10]. The momentum term is an additional term that is added to the weight values, when the weights of the network are updated after each epoch. The value of the momentum term is a fraction of the previous weight update values. Thus, if Δw_i $(t - 1)$ is the weight update value during the previous epoch $(t - 1)$, and $\Delta w_i(t)$ is the weight update during the current epoch i, then the weight update rule is given by Equation 3.2

$$\Delta w_i(t + 1) = w_i + \Delta w_i(t) + M\,\Delta w_i(t - i) \tag{3.2}$$

The momentum term in Equation 3.2 is $M\,\Delta w_i(e - i)$, where M is a val between 0 and 1. The value M thus determines what fraction of the previous weight update value should be added to the current weight update. Momentum can speed up the

[10] When a learning algorithm in ANN causes the total error of the net descend into a valley of the error surface, that valley may or may not lead to the lowest point on the entire error surface. Therefore, the minimum into which the total error will eventually fall is referred to a local minimum.

training process by several orders of magnitude [Zurada, 1992, Dandy and Maier, 2000]. Once the training phase has been completed the performance of trained network has to be validated on an independent data set. Poor validation can be caused by a poor choice of network architecture, or poor data pre-processing.

The momentum term is a factor that can be used to speed up the training process by several orders of magnitude and the momentum factor is a value less than 1.0. If the momentum value is low, it can prevent the network from learning. When a steep error slope occurs during training phase, a small momentum factor is optimal and towards the end, a large momentum factor is desirable. Figure 3.5 shows the JavaNNS error graph for how the sum of the mean squared error, $\sum \varepsilon^2$, decreases with epoch (learning cycle). The red graph indicates the validation whereas the blue graph indicates the training of a network during the learning cycle.

3.3.5 Epoch size

The epoch size is equal to the number of training samples presented to the network between weight updates. Network weight adaptation can be done in on-line or batch mode. In on-line mode, the weights are updated after the presentation of each training pair, as is the case in line 11 in the pseudo code above.

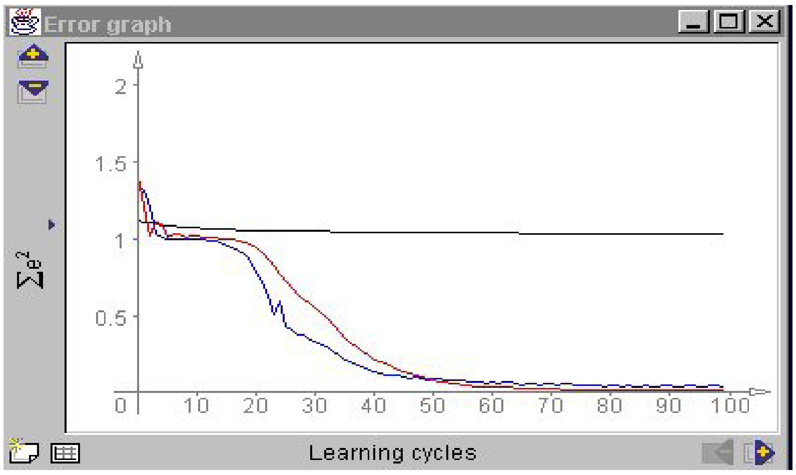


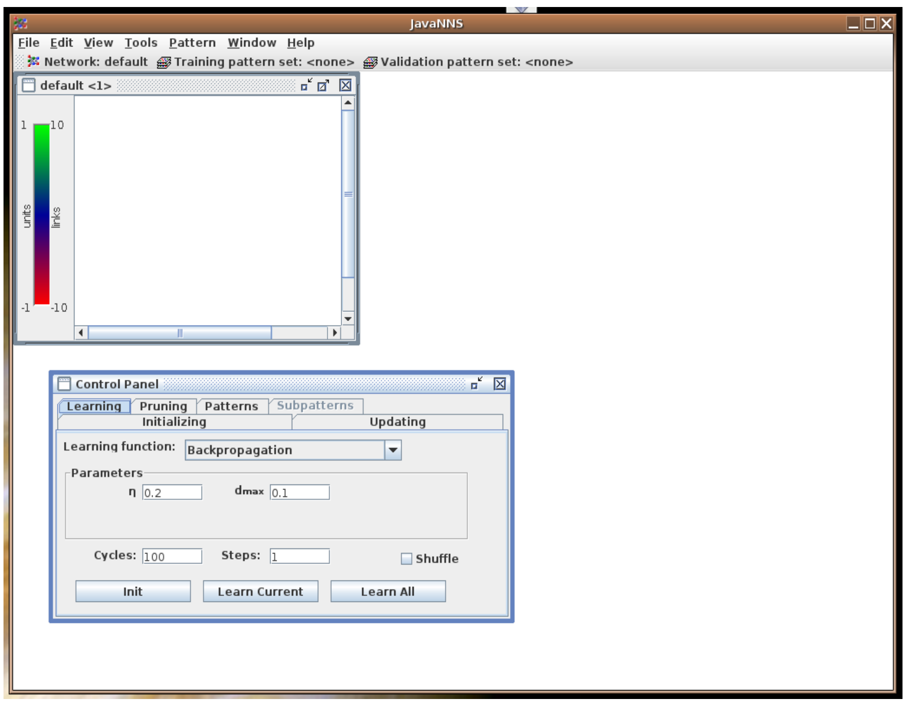
Figure 3-5 Shows the steep error slope

In batch mode, the weight changes for each training pair is accumulated, and the sum of these changes is applied after presentation of the entire training set, for example after one epoch. The preferred option in many applications is batch mode since weight changes for individual training pairs can cancel each other and a more accurate overall weight change is obtained. This guides the learning to move in the direction of the actual gradient at each weight update. The weight update mode can be set to in JavaNNS as shown in Figure 3.6.

3.3.6 Error function

The error function E is the function that is minimised during training. The most commonly used error function is the mean squatted error $\mathrm{E} = \frac{1}{N}\sum_{i=1}^{N} \varepsilon_i$, where ϵ_i is the network error T_i-D_i for all the training patterns $1 < i < N$ in the training set. The error function can penalise large errors and it relates to normal distribution.

The criteria used to decide when the training process should be stopped are important, as they determine whether the neural network has been optimally or sub-optimally trained. In the case of this study, training was stopped when the training error had reached a sufficiently small value or when changes in the training error remained within a small interval. The network training parameters described above, can be specified or adjusted in JavaNNS, as shown in Figure 3.6.



Network: default Training pattern set: <none> Validation pattern set: <none>

Learning function: Backpropagation

η 0.2 dmax 0.1

Cycles: 100 Steps: 1

Figure 3-6 JavaNNS Control panel indicating the learning, pruning, patterns, momentum, learning function and cycles parameters.

3.4 Neural Network Time Series Modelling

The research problem in this thesis can be viewed as the development of a predictive model from a time series of rainfall data. Time series modelling has been extensively researched, and the use of neural networks is an established technique [Corne et al., 1997]. Typical examples of neural network applications are market predictions, meteorological and network traffic forecasting [Davey et al., 1997, Chan et al., 1993, Dorffner, 1996, Collobert et al., 1995].

A time series T is a sequence of vectors, tuples or patterns, x (t), t = 0, 1. . . where t is the elapsed time instant. That is,

$$T = \{x(t_0), x(t_1), \ldots, x(t_i), \ldots\}$$

The sequence *x(t)* may be scalar values or structured objects such as vectors or images. Since rainfall is a scalar value, we consider here only sequences of scalars, although these concepts can easily be transferred to series of structures. The value

x can vary continuously with t, such as rainfall, but in practice it is sampled as a series of discrete data values usually at equal time intervals. In our case, rainfall values are sampled daily at the same time. The sampling rate determines the temporal resolution of the time series. However, it is not always the case that the highest resolution will produce the best predictive performance; it may be the case that every n-th point in the series produces an improved prediction model.

The forecasting problem can be stated as developing a model f that can be used to estimate a future value from a set of values up to the present time. Formally, this can be stated as finding the function f such that f: $R^N \rightarrow R$ that is used to obtain an estimate of x at time $t + d$ by using the historical values of x for N time steps preceding the time step t.

3.5 Data Preparation

The daily raw rainfall values for a number of weather stations in and around the KwaZulu-Natal province was used for experimental training simulations. Fifteen years' data (from 1995 to 2009) daily rainfall records obtained from SAWS were used to train the ANN models.

The weather station code numbered 30160, in the vicinity of Msundusi municipality in Pietermaritzburg coordinate (Latitude = −29.66763 and Longitude = 30.40599) was selected as first experimental data set. This station was selected because it is located within the region under consideration, namely the Msundusi River catchment area.

3.5.1 Data set generation

Split-sample training is a common ANN training method. The basic idea behind this approach is to withhold a small subset of the data for validation, and to train the network on the remaining data [Hung et al., 2009, Dandy and Maier, 2000]. However, it might be difficult to construct a representative validation set when a limited amount of data is available.

Dandy and Maier [Dandy and Maier, 2000] indicate that the holdout method is the only method which maximises utilization of available data. A small percentage of the training data set, is set aside in order to determine whether there is improvement in generalization thus avoiding over-training.

This subset of the data is used as a testing set in a trial phase to determine how long training should continue in order to achieve acceptable generalization ability. The testing subset can then be added to the initial training data set and the whole data set can ultimately be used to train the network for a fixed number of epochs, based on the results from the trial phase [Dandy and Maier, 2000].

The generalisation ability of a neural net refers to the ability of this net to correctly provide responses, for instances that the network has not been trained on previously unseen, extrapolated or interpolated data inputs [Geman and Doursat, 1992]. Generalisation ability is affected by the number of times a given data pair is used to train the network and the number of parameters in the neural network (the latter, indicated by the number of trainable weights in the neural network).

If the same instance or set of instances are not representative of the entire function space and they are repeatedly presented to a neural network, then the network might fit that particular sample set too closely, and could be unable to extrapolate beyond the range of the data used for training. Such a network is said to be over-trained. Conversely, if a neural network is not sufficiently trained on a representative sample set, the trained network might fit the training data too loosely. Such a neural network is said to under-train.

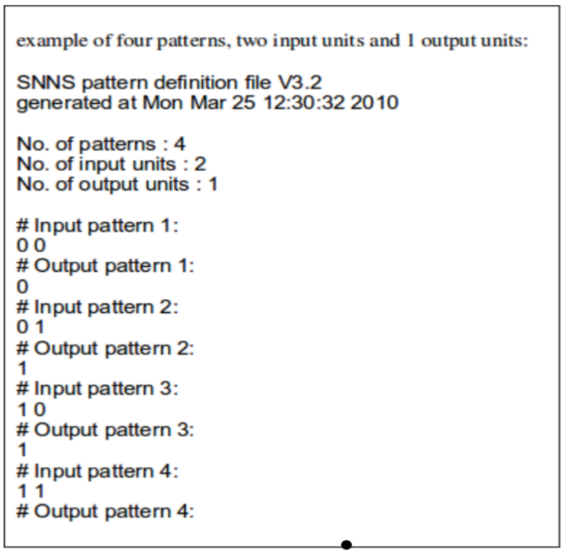
The cross validation technique is also frequently used in ANN training. However, this method substantially compromises the amount of data available for training. In this method, the complete data set is split into two sub-sets, namely a training set and an independent validation set.

In order to train the network using the JavaNNS software, it is necessary to create ASCII files containing training, validation and test data instances in a specific format. An example of the training data file format required by JavaNNS or SNNS was calculated on the excel spreadsheet Figure 3.7.

The overall set of rainfall values was partitioned into subsets. These subsets were reformatted into ASCII files of 100 input-output data pairs, which served as neural network training, validation and test data sets for JavaNNS.

The validation data set was used for cross validation to evaluate the neural network performance during training. The testing data set was used to evaluate the neural network performance after training had been completed.

Poor flood forecasts can be expected when the validation data contain values outside of the range of those used for training. The training and validation sets were representative of the same population since it was a must when doing the validation in training of the available data in ANNs.



```
example of four patterns, two input units and 1 output units:

SNNS pattern definition file V3.2
generated at Mon Mar 25 12:30:32 2010

No. of patterns : 4
No. of input units : 2
No. of output units : 1

# Input pattern 1:
0 0
# Output pattern 1:
0
# Input pattern 2:
0 1
# Output pattern 2:
1
# Input pattern 3:
1 0
# Output pattern 3:
1
# Input pattern 4:
1 1
# Output pattern 4:
```

Figure 3-7 JavaNNS file format

3.5.2 Data pre-processing

As previously noted, the data was divided into sub-sets for training, validation and testing. The output of a neural network is typically between 0 and 1 therefore the raw rainfall figures have to be scaled to a range between 0 and 1 before it can be used as input. Feed-forward neural network architecture was considered in all of the initial training experiments discussed in this section.

Initially input dimensions of five and six nodes were considered. The input data represent a moving window of the previous five or six days of rainfall values. Thus for a time window of rainfall figures for five days, for example D_i, D_{i+1}, D_{i+2}, D_{i+3}, D_{i+4} as inputs and the neural network, was trained on the following day (D_{i+5}). Given the previous days' rainfall figures, the ANN should output the next day's rainfall figure. Initially, a hidden layer of six units was used. The output dimension varied according to the classification method used as discussed further in this section.

Sets of 100 such pre-processed values (as shown in Table 3.1) were transferred to a text file in the JavaNNS file format. The results of training simulations for various network architectures are presented and discussed in Chapter 4.

Table 3.1 is an example of raw and scaled data. Column one contains values with merged zeros, column two indicate scaled training data between 0 and 1, and columns 3-8 indicate the sixth day used as input, while column 7 contains the output used.

TABLE3.1: EXAMPLE OF RAW AND SCALED DATA.

Rain	Scaled	In1	In2	In3	In4	In5	In6	Out
0	0	0	0.1	0.96	0	0.54	0.1	0
0.2	0.10	0.1	0.96	0	0.54	0.1	0	0.09
4	0.96	0.96	0	0.54	0.1	0	0.09	0.99
0	0	0	0.54	0.1	0	0.09	0.99	0.96
1.2	0.54	0.54	0.1	0	0.09	0.99	0.96	0
0.2	0.10	0.1	0	0.09	0.99	0.96	0	0.38
0	0	0	0.09	0.99	0.96	0	0.38	0.2
0.19	0.09	0.09	0.99	0.96	0	0.38	0.2	0.1
5.6	0.99	0.99	0.96	0	0.38	0.2	0.1	0
4	0.96	0.96	0	0.38	0.2	0.1	0	0.2
0	0	0	0.38	0.2	0.1	0	0.2	0
0.8	0.38	0.38	0.2	0.1	0	0.2	0	0.2
0.4	0.20	0.2	0.1	0	0.2	0	0.2	0.14
0.2	0.10	0.1	0	0.2	0	0.2	0.14	1
0	0	0	0.2	0	0.2	0.14	1	0.83

Rain	Scaled	In1	In2	In3	In4	In5	In6	Out
0.4	0.20	0.2	0	0.2	0.14	1	0.83	0
0	0	0	0.2	0.14	1	0.83	0	0.2
0.4	0.20	0.2	0.14	1	0.83	0	0.2	0
0.28	0.14	0.14	1	0.83	0	0.2	0	1
8.2	1.00	1	0.83	0	0.2	0	1	0.2
2.4	0.83	0.83	0	0.2	0	1	0.2	0.92
0	0	0	0.2	0	1	0.2	0.92	0
0.4	0.20	0.2	0	1	0.2	0.92	0	0.38
0	0	0	1	0.2	0.92	0	0.38	0
6.4	1.00	1	0.2	0.92	0	0.38	0	1
0.4	0.20	0.2	0.92	0	0.38	0	1	0.98
3.2	0.92	0.92	0	0.38	0	1	0.98	0.99
0	0	0	0.38	0	1	0.98	0.99	0.76
0.8	0.38	0.38	0	1	0.98	0.99	0.76	0.66
0	0	0	1	0.98	0.99	0.76	0.66	1
8.6	1.00	1	0.98	0.99	0.76	0.66	1	1
4.6	0.98	0.98	0.99	0.76	0.66	1	1	0.72
	0.99	0.99	0.76	0.66	1	1	0.72	0.2
2	0.76	0.76	0.66	1	1	0.72	0.2	0.29
1.6	0.66	0.66	1	1	0.72	0.2	0.29	1
10.2	1.00	1	1	0.72	0.2	0.29	1	0
40.4	1	1	0.72	0.2	0.29	1	0	0.2
1.8	0.72	0.72	0.2	0.29	1	0	0.2	0
0.4	0.20	0.2	0.29	1	0	0.2	0	0.76
0.6	0.29	0.29	1	0	0.2	0	0.76	0.6
10.2	1.00	1	0	0.2	0	0.76	0.6	0.1
0	0	0	0.2	0	0.76	0.6	0.1	0.93
0.4	0.20	0.2	0	0.76	0.6	0.1	0.93	1

Initially, several data scaling and network configurations methods were implemented. The goal of this approach was to determine a data pre-processing and network architecture as sensible starting point for training. We discuss these methods and the reasons for using them as a progression of our research efforts.

Real valued outputs-The raw data was scaled by calculating the maximum (R_{max}) value and minimum (R_{min}) value over an entire data set and for each rainfall value (r_i) in the set. Equation 3.3 shows the transformation used for scaling, where R_{max} and x_{min} are the maximum and minimum rainfall values, r_i is the original, and x_i is the scaled value.

$$x_i = \frac{r_i - R_{\min}}{R_{\max} - R_{\min}} \qquad (3.3)$$

Since the minimum rainfall is 0, *Xmin* = 0 and the scaling process amounted to *X/Xmax*. The data set contained few non-zero values (rainy days) and since *Xmax* was a relatively high value, the largest majority of the scaled data set - as high as 90% - contained zero or near-zero values. This presented a problem in that the neural network weights would tend to all default to zero during training on such sparse sets since in this state, the trained network would achieve correct a classification performance of 90%. It became clear from performance results of networks trained on these data sets, that a sensitivity analysis was needed to obtain a true picture of the neural network performance.

Two network input layer dimensions were considered for training, thus effectively resulting in two types of training data sets. The data training pair used for the first training method consisted of five input values and one output value. The second data set used to train the ANN consisted of the first six daily rainfall values as input and the ANN was then trained on the seventh daily rainfall value. Training was repeated for 100 cycles of epochs. The ANN was validated after every training epoch in order to determine the fraction of the times it produced the correct output.

The performance of networks trained to produce real-valued output were assessed by determining the percentage of outputs within a certain range of the desired outputs. This indicated that a categorical subdivision of the target outputs would be a more appropriate approach to evaluate network performance.

Categorical outputs- In order to obtain an idea of a trained network's predictive ability, it is reasonable to use a set of categorical outputs indicative of the probability of rainfall, thus effectively, a flood prediction index. The third training data pre-processing set hence consisted of the first six daily rainfall values as input, but categorical outputs for the seventh day instead of a real valued output. Two categorisations were considered; the first consisting of two categorical probabilities of flood, namely Low and High. A three category output classification, namely Low, Medium and High, was also used to classify the outputs. In either case, the number of output nodes was equal to the number of categories.

The classification of rainfall data into categories, was achieved as follows: the maximum rainfall value for the period 1995-2009 was calculated. For the case of three categories, two threshold values were selected, namely a lower threshold value t_l, which separates the Low and Medium flood probability categories, and an upper threshold value t_h, which separates the Medium and High categories. The maximum rainfall value, R_{max} was determined and several sets of values for t_l and t_h were experimented with. For example in the case where the range of rainfall values were divided into three equal intervals, t_l and t_h had values $R_{max}/3$ and $2R_{max}/3$, respectively. Thus the output categories for the rainfall value r were determined as follows.

- Low is the range of rainfall values 0 to X mm where X represents one third of the maximum rainfall value range thus, $1 \leq r \leq R_{max}/3]$
- Medium is the range from $R_{max}/3 < r \leq 2R_{max}/3$
- High is the range $2R_{max} < r \leq R_{max}$.

Thus for a two category partition of the outputs, only one threshold was necessary. The fourth data set consisted of the raw rainfall figures scaled to a range between 0 and 1 by using Equals 3.4.

$$x_i = 2 * \frac{2}{\left(1-exp^{(-r_i)}-0.5\right)} \tag{3.4}$$

Since we were interested only in the probability of flooding, we replaced contiguous sequences of zero rainfall values (no rain) by a single zero value. The argument for this action is that for the region under study, flooding is only possible if it rains.

The training algorithm that was used in the data sets above was back -propagation with variable learning rates and momentum values. The results obtained indicated that ANN model performance is very dependent on the number of nodes in the hidden layer. These are discussed in Chapter 4.

Chapter 4

Results:Experimental Results

Summary

The results of network training simulations that have been conducted are presented in this chapter. The goals of these experiments were to determine the data set composition and neural network architecture that would produce the best predictive performance for rainfall and hence, impending flood. These experiments address the two main questions of this research. We investigated predictive performance for the following day, for example for a 24 hour time step, given the previous five or six days' rainfall figures. The findings indicated that data set composition and neural network architecture had a critical influence on rainfall predictive performance.

Several initial ANN models were selected for investigation; these include feed-forward and recurrent network architectures. For each of these initial models, we adopted an incremental approach in adapting and refining the model. An initial plausible architecture was selected and repeatedly modified in order to determine a model that produced improved performance. The findings of these experiments were compared with each other, as well as with results presented in the literature in Chapter 2.

4.1 Introduction

The ANN models implemented in this study are of both feed-forward and recurrent architecture that are commonly used for solving regression problems [Hung et al., 2009]. The models typically consist of three layers, namely an input, hidden and output layer.

For recurrent networks, architectures that use special hidden units were also implemented. An initial ANN model was selected and an iterative approach was adopted in improving the predictive performance of the initial neural network architecture. This process mainly entailed modifying the number of hidden neurons, and evaluating the performance of the resulting trained neural network.

Two ’base’ categories of ANN models were explored, namely feed-forward architectures trained on a five or six day input window, and recurrent architectures trained on the previous six days of rainfall values. In general, the following below procedure was followed for a selected ANN flood prediction model. Specific parameters used for the initial neural network architectures are presented in Tables 4.1 and 4.2.

```
     Repeat for each selected initial ANN model (feed-forward or recurrent)

1. Select initial input and output dimensions of the model.
2. Configure initial network architecture.
4. Select and compose the data sets needed to train, validate and test the
   Model.
5. Repeat until peak or asymptotic ANN model performance is observed.
   5.1 Train and test the trained network using performance evaluation
       Statistics.
   5.2 Incrementally modify the number of hidden layer neurons.
```

We proceeded from the premise that an initial ANN model performance would improve or degrade if the architecture was modified slightly. If the performance remained largely the same, a larger modification was made upon which, if no significant change was observed, the original architecture was retained. The implication in the latter case was thus that the changed parameter did not have any significant effect on performance for the given model.

The first ANN models considered for training and testing were simple multilayer feed-forward and Elman models with sigmoid transfer function. The three layers, namely input, hidden, special hidden and output nodes were selected as a base architecture for the network since that was the de-facto standard for pattern recognition training problems as is the case in this study.

4.2 Model design

In the model design stage, several ANN models were tested on the data from one weather station code 30160 in order to find an appropriate ANN architecture that could be employed in the forecasting flood for our study.

4.2.1 Data normalization

As mentioned in Chapter 3, the split-sample training is a common method to train ANN models. The fifteen years data obtained were divided into training, validation and test data sets after it had first been normalized to a certain range as discussed below in further detail. Hung et al. [Hung et al., 2009] in their recent works show that a model with improved performance on the training set does not always provide better performance during testing. In this study, the accuracy of the training set and the validation set were observed and recorded during the training process.

The observed historical data for rainfall was first normalized within a range of 0.1...0.9 as the ANN model used the sigmoid transfer function. Sigmoid functions, for example the logistic function indicated in Figure 5.1.2 is used or recommended over the other functions such as threshold or step function when applied to hidden units because of its flexibility, and is easier to train from. Range values close to 0 or to 1, would have asymptotic domain values as shown in Figure 4.1 and these range values were avoided.

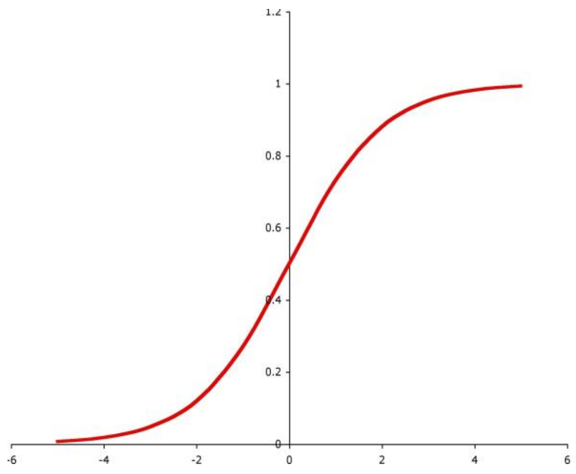


Figure 4-1 A sigmoid with a range between 0 to +1

The normalized data was then presented to the ANN model at the input layer. Hidden layers with different dimensions were tested as shown in Table 4.3.1 and 4.3.2 number of units in the output layer represents the number of values to be estimated. In this study only one output node was considered for feed-forward network models and Elman network models. This output value indicates the predicted rainfall value for the next day.

4.2.2 Model parameters

One hydrological parameter as already mentioned earlier, the historical rainfall data in millimetres, was used in this study since the advantage of our approach is that it does not require many variables and very little knowledge of the flooding domain. The model parameters such as the number of epochs were varied to achieve improved performance by conducting several training and testing experiments for different model architectures. In particular, the modifications considered for improving performance were as follows:

- Normalizing the historical rainfall data.
- The neurons in the hidden layer were progressively increased from 2 to 1neurons. An improvement in the performance of the model was observed as the hidden layer dimension was increased.
- The number of training epochs was set to 100 as a result of trials conducted with different and higher values.

The first network architecture employed during this part of the study is known as a multilayer perceptron (MPL), which is trained with accuracy correction back-propagation learning algorithm. Each weight in the network is adapted by correcting the current value of weight with a term that is proportional to the current output accuracy as determined by the gradient-descent learning method.

4.3 Elman and Feed-Forward Networks Architecture

One disadvantage associated with neural network learning is that it is necessary to specify the network architecture in terms of the number and configuration of its hidden units beforehand. ANN model performance and learning ability depend on the suitability of its architecture. If a network is too small, it may have insufficient degrees of freedom (number of parameters) to fully capture all the underlying relationships in the data. On the other hand, if a network is too large, it may not generalise events in the training data that are not necessarily representative of the system under consideration [Hanavar et al., 1999].

Following above procedure for the feed-forward networks used in this experiment, it was started with a small hidden layer of two nodes (network type A in the table), and then the number of hidden layer nodes progressively incremented by two. The same procedure was carried out till the number of hidden layer nodes reached 14. The same procedure was used in Elman network, whereby the hidden layer of two nodes (network type H in the table) were increased up to 14 hidden nodes for both two hidden layers. The architecture of these models was varied as given in Table 4.3.1 and Table 4.3.2, recurrent networks.

The different models of feed-forward network and Elman network are shown in the number of hidden nodes and special nodes. Table 4.3.1 and 4.3.2 show feed-forward and recurrent networks. The first column contains a label for the network architectures, which are named label A to G and label H to N for feed- forward and recurrent networks respectively. The second column contains the neural network architecture used in this experiment whereby the numbers, for example for feed-forward 6-2-1 and recurrent network 6-2-2-1, indicate the six input nodes, two hidden nodes and two special hidden nodes for recurrent network and one output node respectively. The third column contains the learning function, for example feed-forward networks were trained by the back-propagation, while the recurrent network was trained by the JE back-propagation.

4.3.1 Feed-forward network architecture

- The first model (Model A) used the feed-forward network with a simple structure involving six nodes in the input layer, the hidden layer having two hidden nodes, and one node in the output layer (6-2-1). The learning function used was back-propagation.

- The second model (Model B) the network type, learning function and input data were kept the same as model B, but the number of hidden nodes was increased from two to four (6-4-1).

- In the third model (Model C), the procedure was the same as model A and B except that the hidden nodes were increased from four to six (6-6-1).

- The fourth model (Model D) was also kept the same as model A, B and C, but the hidden nodes were also increased from six to eight hidden nodes (6-8-1).

- The fifth model (Model E), sixth model (Model F) and seventh model (Model G) were kept the same as model A and the hidden nodes were increased to ten (6-10-1), twelve (6-12-1) and fourteen (6-14-1) respectively as indicated in Table 4.3.1.

4.3.2 Recurrent networks

The eighth model (Model H) and other remaining models used the Elman network with a structure involving six nodes in the input layer, two hidden layers having two hidden nodes and output layer having one node (6-2-2-1). The learning function used here was JE back-propagation. The ninth model (Model I) was kept the same as model H except that there were four hidden nodes in the first and second hidden layer, and the output node remained the same (6-4-4-1).

TABLE 4.1: Feed-forward network architectures

Feed-Forward	Input-Hidden-Output	Learning Function
A	6 – 2 – 1	Back-propagation
B	6 – 4 – 1	Back-propagation
C	6 – 6 – 1	Back-propagation
D	6 – 8 – 1	Back-propagation
E	6 – 10 – 1	Back-propagation
F	6 – 12 – 1	Back-propagation
G	6 – 14 – 1	Back-propagation

The tenth model (Model J) was kept the same as Model H, but the number of hidden nodes in both hidden layers were increased from four to six (6-6-6-1). The same was applied to model eleven, twelve, thirteen and fourteen except that the hidden nodes in the hidden layers were increased to eight (6-8-8-1), ten (6-10-10-1), twelve (6-12-12-1) and fourteen (6-14-14-1) respectively, referring to Table 4.3.2. Elman network architectures used in this study; column one contains the label of the network architecture, column two contains different hidden nodes and column three indicates the learning function used.

TABLE 4.2: ELMAN NETWORK ARCHITECTURES

Elman	Input-Hidden-Special Hidden-Output	Learning Function
H	6 – 2 – 2 – 1	JE Back-propagation
I	6 – 4 – 4 – 1	JE Back-propagation
J	6 – 6 – 6 – 1	JE Back-propagation
K	6 – 8 – 8 – 1	JE Back-propagation
L	6 – 10 – 10 – 1	JE Back-propagation
M	6 – 12 – 12 – 1	JE Back-propagation
N	6 – 14 – 14 – 1	JE Back-propagation

The differences between the Elman network and feed-forward network architectures are seen on the extra hidden layer, which is normally called special hidden layer.

4.4 Training Procedure and Pattern Set

Architecture models were selected and only models which performed well. The results are listed in tabular form. The following procedure was followed for a developing the

ANN flood prediction model:

1. The daily rainfall data was selected to be used for training, validation and testing of the models, and also the input and output variables were selected.

2. The initial network architecture as presented in Table 4.3.1 and 4.3.2 were configured.

3. The number of hidden layer neurons was determined in order to improve the performance of the neural network.

4. The trained neural network was tested using selected performance evaluation statistics such as mean percentage accuracy, correlation coefficient and contingency table or confusion matrix.

5. From item four was repeated until a predetermined predictive performance criterion was satisfied.

The above-mentioned summarised the ANN development. More detailed training procedure was conducted as follows: training data patterns were presented sequentially to the input layer and this data was then propagated through the network. The resulting output predictions were $y_i(p)$, where $1 \leq i \leq N$ ranged over the number of output nodes and $1 \leq p \leq P$ denoted the pattern number. These values were compared with the corresponding desired or actual output, $d_i(p)$. Equation 4.1 was used to calculate the Mean Square Error, E, over all the patterns P in the training data set. Thus, a batch weight update procedure was adopted.

The back-propagation learning rule was used to adjust the weights until E had decreased till the training stopping criterion was met.

$$E = \frac{1}{n} \Sigma_{p=1}^{P} \Sigma_{i=1}^{n} [y_i(p) - d_i(p)]^2 \qquad (4.1)$$

The steps followed when applying back-propagation in training the neural network, included the following:

1. The weights and bias were initialized to small random values between 0 and 1.

2. The training data was normalized.

3. The training set inputs were presented to the neural network and for each training input, the network output of the neurons in the hidden layer and in the output layer were computed. Equation 4.2 shows how the network output was computed for the node *i*, with N input signals and a threshold value θ_i.

4. The activation output for each network output was computed by applying the sigmoid activation rule.

5. The global network error was computed.

6. The weights of the network were adjusted using the back-propagation rule.

7. Steps 3 to 5 were repeated until the global network error converged to a predetermined level or until a fixed number of epochs had elapsed.

$$net_i = \sum_{j=1}^{N} W_j X_j + \theta_i \; ; \qquad \theta_i \text{ is a threshold for node i} \qquad (4.2)$$

4.4.1 Pattern set composition

The number of epochs for both training and validation were adjusted to 1 000 with the number of training and validation patterns per training set to 100. A training pair consisted of rainfall figures data for six consecutive days as input data, and the rainfall figure for the following day (seventh day) as output data.

A training data set consisted of 100 such training pairs, obtained by moving a ”window” of six plus one rainfall values sequentially over the given rainfall data. The first pattern set was prepared by taking the first seven normalised daily rainfall data and splitting them into six inputs and one output, sequentially. The same procedure was repeated sequentially until 100 of six input and 100 of one output were created

Figure 3.7.

Following the same procedure for the input variables, daily rainfall data was divided into three different subsets, sequentially: one subset for training the neural network (100 patterns set), one for model validation (100 pattern set) and one for model testing (100 pattern set)[11]. The training pattern set in this case was used to update the weights of the network; while the validation set was used to observe the progress of training. The testing data set was used for the final evaluation of the model performance. This procedure was carried out for at least five such training, validation and testing pattern sets, for both the Elman and feed-forward networks.

The next step was to determine suitable output values for flood prediction. The output was then determined by the continuous values, the two categories and the categories below. At first only one output node was considered for feed-forward network models and Elman network models which were not normalised, and the results obtained showed that models did not perform well since the mean percentage accuracy was less than 20%. The second procedure was to categorise the seventh day rainfall data or the output into two categories. The results also showed bad performance. The third procedure was then followed in this manner:

- The output value was then categorised (flood and no flood; low, medium and high flooding) after it had been trained.

- Each[11] Testing (100 pattern set) in this study was named Experimental simulation.

- Exp1 to Exp5 categories were assigned to a neural network output interval, and these intervals were adjusted to more appropriately suit the likelihood and extent of flooding.

The following threshold values 0.25, 0.5, 0.75 were used in order to compare the neural network results with the observed data. The motivation for choosing one output and categorising it, was that it would be reasonable for the neural network to provide a

[11] Testing (100 pattern set) in this study were named Experimental simulation, Exp1 . . . ..Exp5

categorical output rather than a, numeric output for flood prediction. That is, the focus for flood prediction is not a predicted measure of rainfall, but rather an indication of whether, and the extent to which, a flood is likely to occur. The performance of all the models are summarised and described in later sections.

4.5 Experimental Performance

The results of each network performance in both Elman and feed-forward neural networks which were compared with the observed data are summarised in Tables below. The network performance was thus assessed by considering true and false positive and negative classifications.

4.5.1 Performance of neural network

Table 4.3 provides a summary of the results performance in mean percentage of trained network under Elman neural network for all models. Each network was trained from three different pattern sets, namely training, validation and testing.

These three sets are named simulation experiment (SE) Exp1 . . . Exp5. The target output range of 0 to 1 was divided into two categories (see Figure 4.2), indicating low and high probabilities of flood, respectively.

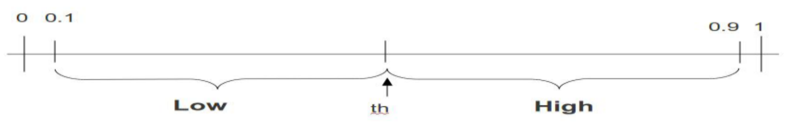


Figure 4-2 Target output range of 0 to 1 divided into two categories

The threshold value 0.75 in Table 4.3 was used to separate the two categories when comparing the computed data or trained data with observed data. Table 3 shows Elman computed results performance mean in % with the threshold value of 0.75.

TABLE 4.3: ELMAN COMPUTED RESULTS PERFORMANCE MEAN

SE	(6-2-2-1)	(6-4-4-1)	(6-6-6-1)	(6-8-8-1)	(6-10-10-1)	(6-12-12-1)	N(6-14-14-1)
Exp1	48	45	47	57	53	56	46
Exp2	58	61	65	57	65	63	65
Exp3	57	52	66	49	49	63	48
Exp4	57	53	51	55	52	43	45
Expe5	61	42	53	63	64	69	70

Table 4.4 shows a summary of the performance of a network trained in Elman neural network for all models. The target output range of 0 to 1 was divided into two categories, indicating low and high probabilities of flood, respectively. These categories were separated by a threshold value of 0.5.

TABLE 4.4: PERFORMANCE MEAN IN % FOR DIFFERENT CONFIGURATIONS OF ELMAN NETWORK

SE	(6-2-2-1)	(6-4-4-1)	(6-6-6-1)	(6-8-8-1)	(6-10-10-1)	(6-12-12-1)	N(6-14-14-1)
Exp1	50	54	45	53	56	56	58
Exp2	4848	57	59	52	55	48	56
Exp3	57	46	57	54	43	52	47
Exp4	50	40	48	44	45	51	48
Expe5	57	41	49	62	55	66	67

Tables 4.5 and 4.6 indicate the summary performance in mean of networks trained in Elman neural network for all models. The target output range of 0 to 1 was divided into three categories this time (see Figure 4.3), indicating low, medium and high probabilities of flood, respectively. These categories were separated by a threshold value of 0.5 . . . .0.75 and 0.25 . . . .0.5 as indicated in Tables 4.5 and 4.6 respectively when compared with observed data.

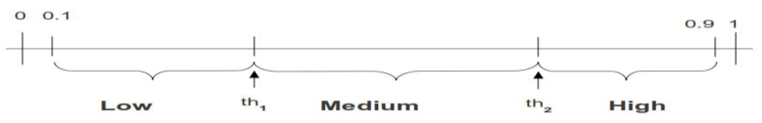


Figure 4-3 Target output range of 0 to 1 divided into three categories

TABLE 4.5: ELMAN COMPUTED RESULTS PERFORMANCE MEAN IN %, THRESHOLD 0.5...0.75

SE	(6-2-2-1)	(6-4-4-1)	(6-6-6-1)	(6-8-8-1)	(6-10-10-1)	(6-12-12-1)	(6-14-14-1)
Exp1	42	38	39	48	48	49	44
Exp2	44	50	56	33	49	39	56
Exp3	49	43	52	44	40	52	41
Exp4	45	40	41	43	40	39	36
Exp5	57	36	45	55	52	61	65

TABLE 4.6: ELMAN COMPUTED RESULTS PERFORMANCE MEAN, THRESHOLD 0.25....0.5

SE	(6-2-2-1)	(6-4-4-1)	(6-6-6-1)	(6-8-8-1)	(6-10-10-1)	(6-12-12-1)	(6-14-14-1)
Exp1	46	50	38	41	50	49	57
Exp2	41	43	50	33	24	32	47
Exp3	49	34	43	51	32	33	39
Exp4	41	32	38	33	39	48	43
Exp5	40	34	40	46	43	46	46

The same procedure done in the Elman network table was also applied to feed-forward networks as follows: Tables 4.7 and 4.8 provide a summary of the results performance in mean of each network using feed-forward neural network for all models.

The target output range of 0 to 1 was divided into two categories, indicating low and high probabilities of flood respectively. These categories were separated by threshold values of 0.75 and 0.5, as in Table 4.7 and 4.8.

TABLE 4.7: FEED-FORWARD COMPUTED RESULTS PERFORMANCE MEAN %, THRESHOLD 0.75

SE	(6-2-1)	(6-4-1)	(6-6-1)	(6-8-1)	(6-10-1)	(6-12-1)	(6-14-1)
Exp1	53	43	47	48	52	47	52
Exp2	48	58	57	52	58	45	61
Exp3	54	49	49	44	46	47	56
Expt4	42	55	55	45	54	51	50
Exp5	64	58	44	61	63	63	68

The Table 4.8 indicates the results for feed-forward neural network performance mean, the target output range of 0 and 1 was divided into two categories, indicating low and high probabilities of flood. The threshold value of 0.5 was used to separate these categories.

TABLE 4.8: FEED-FORWARD COMPUTED RESULTS PERFORMANCE MEAN IN %

SE	(6-2-1)	B(6-4-1)	C(6-6-1)	(6-8-1)	(6-10-1)	(6-12-1)	(6-14-1)
Exp1	44	41	39	47	47	45	58
Exp2	52	49	54	49	53	47	54
Exp3	47	42	41	49	48	46	53
Exp4	51	48	50	51	45	50	49
Exp5	58	59	37	56	60	56	57

Tables 4.9 and 4.10 provide the summary performance mean of each network trained in feed-forward neural network for all models. The target output range of 0 to 1 was divided into three categories, indicating low, medium and high probabilities of flood respectively. These categories were separated by threshold values of 0.5 . . . 0.75 and 0.25 . . . 0.5 see Table 4.9 and 4.10 respectively.

TABLE 4.9: FEED-FORWARD COMPUTED RESULTS PERFORMANCE MEAN

SE	(6-2-1)	B(6-4-1)	C(6-6-1)	(6-8-1)	(6-10-1)	(6-12-1)	(6-14-1)
Exp1	44	41	39	47	47	45	58
Exp2	52	49	54	49	53	47	54
Exp3	47	42	41	49	48	46	53
Exp4	51	48	50	51	45	50	49
Exp5	58	59	37	56	60	56	57

Table 4.10 was applied the same procedure as to Table 4.9, but this time the categories (low, medium and high) were separated by threshold values of 0.25 . . . .0.5. The results obtained in mean percentage are summarised in Table 4.10.

TABLE 4.10: FEED-FORWARD COMPUTED RESULTS PERFORMANCE MEAN PERCENTAGE,

	(6-2-1)	(6-4-1)	(6-6-1)	(6-8-1)	(6-10-1)	(6-12-1)	(6-14-1)
Exp1	40	33	34	42	43	38	48
Exp2	42	45	49	37	49	40	52
Exp3	45	35	39	40	38	44	50
Exp4	38	43	45	41	42	45	44
Exp5	57	54	34	56	58	41	44

4.5.2 Overall performance

The results in the form of mean percentage accuracy obtained for feed-forward models (A, B, C, D, E, F, G) with different categories used are shown in Table 4.11 threshold 0.5, Table 4.12 threshold 0.75, Table 4.13 thresholds 0.25 . . . .0.5, Table 4.14

thresholds 0.5 . . . .0.75.

TABLE 4.11: FEED-FORWARD NETWORKS SCALE USED 0.

Feed-forward network, threshold 0.5		
Model	Architecture used	Mean accuracy%
A	6 – 2 – 1	50.4 %
B	6 – 4 – 1	47.8 %
C	6 – 6 – 1	44.2 %
D	6 – 8 – 1	50.4 %
E	6 – 10 – 1	50.6 %
F	6 – 12 – 1	48.8 %
G	6 – 14 – 1	54.2 %

Table 4.12 provides the summary performance in mean percentage of each model for feed-forward neural networks with threshold value of 0.75.

TABLE 4.12: FEED-FORWARD NETWORKS SCALE USING 0.75

Feed-forward network, threshold 0.75		
Model	Architecture used	Mean accuracy%
A	6 – 2 – 1	52.2 %
B	6 – 4 – 1	52.6 %
C	6 – 6 – 1	50.4 %
D	6 – 8 – 1	50.0 %
E	6 – 10 – 1	54.6 %
F	6 – 12 – 1	50.6 %
G	6 – 14 – 1	57.4 %

Table 4.13 provides the summary performance in mean percentage of each model in feed-forward neural network. The output range [0, 1], was discretised into three categories output with threshold values of 0.25 and 0.75 between categories.

TABLE 4.13: FEED-FORWARD NETWORKS SCALE USING 0.25....0.5

Feed-forward network, threshold 0.25 . . . .0.5		
Model	Architecture used	Mean accuracy%
A	6 – 2 – 1	41.4 %
B	6 – 4 – 1	40.2 %
C	6 – 6 – 1	35.4 %
D	6 – 8 – 1	44.0 %
E	6 – 10 – 1	39.2 %
F	6 – 12 – 1	41.2 %
G	6 – 14 – 1	40.6 %

Table 4.14 provides the summary performance in mean percentage of each model in feed-forward neural network. Again here, the output range [0, 1], was discretised

into three categories output with threshold values 0.5 and 0.75 between categories.

TABLE 4.14: FEED-FORWARD NETWORKS SCALE USED 0.5...0.7.

Feed-forward network, threshold 0.5....0.75		
Model	Architecture used	Mean accuracy%
A	6 – 2 – 1	44.4 %
B	6 – 4 – 1	42.0 %
C	6 – 6 – 1	40.2 %
D	6 – 8 – 1	43.2 %
E	6 – 10 – 1	46.0 %
F	6 – 12 – 1	41.6 %
G	6 – 14 – 1	47.6 %

The summarised results for Elman network in the form of mean percentage accuracy were obtained for model (H, I, J, K, L, M, N). The target output range [0, 1] was divided into two categories as shown in Table 4.15 with threshold values 0.5 between the categories, and Table 4.16 with threshold value 0.75. As previously discussed, in Table 4.17 and Table 4.18 the target output range [0, 1] was divided into three categories with the threshold 0.2, 0.5, 0.5 and 0.75 respectively.

TABLE 4.15: ELMAN NETWORKS SCALE USING 0.5

Elman (recurrent) network, threshold 0.5		
Model	Architecture used	Mean accuracy%
H	6 – 2 – 2 – 1	52.4 %
I	6 – 4 – 4 – 1	47.6 %
J	6 – 6 – 6 – 1	51.6 %
K	6 – 8 – 8 – 1	53.0 %
L	6 – 10 – 10 – 1	50.8 %
M	6 – 12 – 12 – 1	54.6 %
N	6 – 14 – 14 – 1	55.2 %

Table 4.16 provides the summary performance in mean percentage of each model for Elman neural networks with threshold value of 0.75. The first column indicates the models used, the second column shows the architecture and the third column is the percentage mean accuracy obtained during the training of networks.

TABLE 4.16: ELMAN NETWORKS SCALE USING 0.75

Elman (recurrent) network, threshold 0.75		
Model	Architecture used	Mean accuracy%
H	6 – 2 – 2 – 1	56.2 %
I	6 – 4 – 4 – 1	50.6 %
J	6 – 6 – 6 – 1	56.4 %
K	6 – 8 – 8 – 1	56.2 %
L	6 – 10 – 10 – 1	56.6 %
M	6 – 12 – 12 – 1	58.8 %
N	6 – 14 – 14 – 1	54.8 %

Table 4.17 provides the summary performance in mean percentage of each model in Elman neural network. The first column indicates the models used, the second column shows the architecture and the third column is the percentage mean accuracy obtained during the training of networks. Again here, the output range [0, 1], was discretised into three categories output with threshold values 0.25 and 0.5 between categories.

TABLE 4.17: ELMAN NETWORKS SCALE USING 0.25….0.5

Elman (recurrent) network, thresholds 0.25….0.5		
Model	Architecture used	Mean accuracy%
H	6 – 2 – 2 – 1	43.4 %
I	6 – 4 – 4 – 1	38.6 %
J	6 – 6 – 6 – 1	41.8 %
K	6 – 8 – 8 – 1	40.8 %
L	6 – 10 – 10 – 1	37.6 %
M	6 – 12 – 12 – 1	41.6 %
N	6 – 14 – 14 – 1	46.4 %

Table 4.18 provides the summary performance in mean percentage of each model in Elman neural network. The first column indicates the models used, the second column shows the architecture and the third column is the percentage mean accuracy obtained during the training of networks. The output range [0, 1], was discretised into three categories output as in the previous table but this time was separated by threshold values 0.5 and 0.75 between categories.

Another parameter used to interpret the network simulation results was a classification model which could be defined as a mapping of instances into a certain class [Fawcett, 2004]. The classifier result is a real value or continuous output in which the classifier boundaries between classes are determined by the threshold value.

TABLE 4.18: ELMAN NETWORKS SCALE USING 0.5 . . . 0.75

Elman (recurrent) network, threshold 0.5 . . . .0.75		
Model	Architecture used	Mean accuracy%
H	6 – 2 – 2 – 1	47.4 %
I	6 – 4 – 4 – 1	41.4 %
J	6 – 6 – 6 – 1	46.6 %
K	6 – 8 – 8 – 1	44.6 %
L	6 – 10 – 10 – 1	45.8 %
M	6 – 12 – 12 – 1	48.0 %
N	6 – 14 – 14 – 1	48.4 %

For these experimental studies, a two class prediction problem was considered, in which the outcomes were labelled either as positive (*p*) or negative (*n*) class. The four outcomes from this type of binary classifier are as follows:

- If the outcome from a prediction is *p* and the actual value is also *p*, then it is called a true positive (TP); where the actual value is *n* then it is said to be a false positive (FP).

- A true negative is found when both the prediction outcome and the actual value are *n*, and false negative is when the prediction outcome is *n* while the actual value is *p*.

For this study, two selected Elman and Feed-Forward neural network simulations were considered in order to determine whether the neural network predicted low (no rainfall) and high (rainfall) flood correctly. The results of the two best performing models are presented in Figure 4.4 and Figure 4.5.

In the tables in the figures below, TP represents a high rainfall prediction corroborated by a high expected result. FP represents false positives. Similarly, true negatives are represented by TN and false negatives by FN. Some of the terms used are:

- The *sensitivity* is an indication of how well the model can discriminate positive expected values. It is simply the *True Positive proportion* of all positive events. *Sensitivity* gives information about the proportion of cases picked out by the model.

  True Positive rate is given by $TPR = \dfrac{TP}{TP + FN}$

- The *False Positive Rate* is given by $FPR = \dfrac{FP}{FP + TN}$, i.e., incorrectly identified positive events

- The *Accuracy* is given by $ACC = \dfrac{TP + TN}{P + N}$, whereby the P and N are the prediction results from 100 positive and negative instances. In this case a false positive occurred when the network predict a high flooding output (*p*), but actually there is low flooding expected value (*n*).

On the other hand a false negative occurs when the network predicts low flooding (*n*), but actually there was a high flooding expected value (*p*). Table 4.4 shows the Receiver Operator Characteristic (ROC) table for the feed-forward neural network labelled I, with 6 input, 10 hidden and 1 output units (6-10-1). The table indicates a sensitivity of 33.7%, and an accuracy of 54.6%

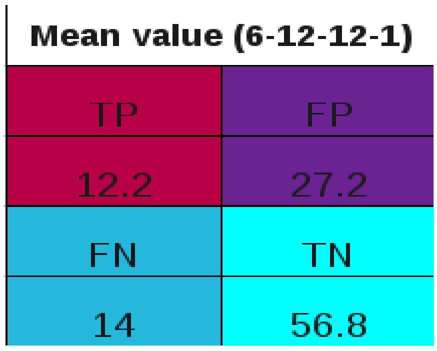
TABLE 4.4: ROC table of the mean values for the feed-forward neural network labelled I (6-10-1)

Mean value (6-10-1)	
TP	FP
6.2	33.2
FN	TN
12.2	48.4

Table 4.5 shows the ROC table for the experimental results of the Elman neural network labelled M, with 6 input, 12 hidden, 12 context and 1 output units (6-12-12-1). The table indicates a sensitivity of 46.6% and an accuracy of 69%. This neural network model (with 10 hidden and 10 context units) is more suitable for

flood prediction than the feed-forward model.

TABLE 4.5: ROC table of the mean values for the Elman neural network labelled M (6-12-12-1)



Mean value (6-12-12-1)	
TP	FP
12.2	27.2
FN	TN
14	56.8

The findings presented in Chapter 5 show that the performance of the networks is influenced by different threshold categories and also by varying the number of hidden node in the hidden layers. Certain of these architectures yielded the best performance of the ones explored during this study. The results are illustrated in tabular forms for analysis and discussion in Chapter5.

Chapter 5

Analysis and Discussion

Summary

This chapter presents the analyses of the results obtained for flood prediction, as reported in Chapter 4. We present and discuss responses to the research problem, research objectives and research questions.

5.1 Hypotheses of the Study

The following sub-objectives were given in Chapter 1:

1. Pre-process historical weather data into a form that is suitable for training neural networks.

2. Show that artificial neural networks can be used as a valid effective approach to predict floods from meteorological data.

3. Determine the architecture of the neural network that will yield the best predictive performance for precipitation.

These sub-objectives were achieved as described. This section presents the motivation for the methodology that was followed and the deductions inferred from results of the experiments conducted during this study.

5.1.1 Methodology

A total of 14 neural network models were implemented in order to study the use of daily rainfall data as predictive variable for floods in the selected region. Neural networks were trained, verified and tested on data for the period 1995 to 2009. The aim of these training simulations was to determine the neural network architecture that would yield the best flood predictive performance. Each type of neural network was studied for different hidden layer dimensions, and a minimum of five neural networks were trained to obtain preliminary results for each architecture and for each hidden layer size. This approach was successful and an appropriate architecture was identified.

5.1.2 Pre-processing and Post-processing

Given the Msundusi region's historical rainfall patterns, it was expected that the data would contain a great proportion of zero rainfall values compared to non-zero values. Stated differently, the non-zero rainfall values were sparsely distributed in the data. This condition caused neural networks to lapse into learning a zero output mapping since the trained network would then correctly predict the majority of outputs. To obviate this situation, consecutive instances of zero rainfall were substituted by a single zero value. This approach is reasonable since flooding cannot occur if there is no preceding rainfall.

A number of scaling methods were considered in preparing the data for training and testing. Since the neural network employed a sigmoidal activation function, the output of the network would be constrained to the unscaled range of $[0, \ldots, 1]$. In order to compare this output with desired values, it was thus necessary to scale the target values to the same range. The sigmoid function range is also asymptotic along the range $[0 \ldots 1]$, and hence the rainfall data was scaled to within a range of $[0.1 \ldots 0.9]$ to prevent the neural network weights from having to increase in magnitude to excessively large positive or negative values.

The scaled output was further categorised using two classification schemes. The first scheme employed a single threshold value, th, to separate the output range into two categories, namely a low probability of flooding and a high probability of flooding. Thus for $0.1 < \text{th} < 0.9$, an output value less than th indicates a low likelihood, and a value greater than or equal to *th* indicates a high likelihood of flooding. The second categorisation approach introduced two thresholds like *th*, which separated the output into low, medium and high probabilities of flood. Several values were tried for *th*. The findings of these experiments are summarised under 5.2.

5.2 Findings

The performance results of the 14 ANN models are presented in tabular form. These results show the forecast accuracy of the 14 models for evaluation purposes. The measures used to evaluate performance included the mean percentage accuracy and correlation coefficient.

The mean percentage accuracy is computed by [*Number of correctly predicted output/Total number of outputs*100*]. The average of each network simulation output has been calculated and the results presented in this section. The correlation coefficient r_{xy} between the variables *x* and *y*, along with the merging means and variances of X and Y, determines this linear relationship:

$$r_{xy} = \Sigma_{i=1}^{n} \frac{(x_i - x)(y_i - y)}{n - 1 s_x s_y} \qquad (5.1)$$

where x and y are the mean values for x and y, respectively, and s_x and s_y are the standard deviations of *x* and *y* respectively for *n* value pairs. The correlation coefficient used was obtained by using the scatter plot[12] for comparing the observed data and simulation results.

5.2.1 Feed-forward neural network performance

The performance results for feed-forward neural network models labelled A-G are shown in Table 5.1. The categories of the target output were separated by different threshold values, which are presented in Table 5.1. The first column contains the label (A-G) of the network architecture which comprised of six input units, one output unit and different hidden nodes. Network architectures were created as follows: A= 6:2:1, B=6:4:1, C= 6:6:1, D= 6:8:1, E= 6:10:1, and F= 6:12:1. Columns two to five contain the mean percentage accuracy obtained using different thresholds. Column six contains correlation coefficient (r) for the observed and actual results.

[12] A scatter plot is a type of mathematical diagram using Cartesian coordinates to display values for two variables of a set of data.

The results of the best predictive models using confusion matrix are shown in Table 5.2. The results of model M (Elman) in experiment four clearly shows the best predictive power among the experiments presented in the table since the accuracy is 49% which is better than the rest of the tests. In Table 5.1 notice the performance statistics for the feed-forward models and the mean percentage accuracy for different threshold categories

TABLE 5.1: PERFORMANCE STATISTICS FOR FEED-FORWARD MODELS

Feed-forward network performance (%)					
Model	0.5	0.75	0.25, 0.5	0.5, 0.75	R
A	50.4	52.2	41.4	44.4	-0.5198
B	47.8	52.6	40.2	42.0	-0.4397
C	44.2	50.4	35.4	40.2	-0.5794
D	50.4	50.0	44.0	43.2	-0.9872
E	50.6	54.6	39.2	46.0	-0.3136
F	48.8	50.6	41.2	41.6	-0.5279
G	54.2	57.2	40.6	47.6	-0.295

TABLE 5.2: COMPARISON OF THE BEST PREDICTIVE MODELS USING TPR, FPR AND ACC

Elman		
TPR	FPR	ACC
0.55	0.43	0.28
0.33	0.35	0.32
1	0.37	0.32
0.43	0.17	0.49
0.4	0.31	0.32

Feed-forward		
TPR	FPR	ACC
0	0.48	0.26
0.36	0.35	0.29
0.32	0.43	0.19
0.5	0.6	0.23
0.14	0.3	0.32

5.2.2 Elman recurrent network performance

Table 5.3 shows the performance results for models H-N, which are of the Elman recurrent network type. The same procedure of categorizing was also followed, as mentioned above. The first column in this table contains the label of the network architecture (H- N). Network architectures were created as follows: A= 6:2:2:1, B= 6:4:4:1, C= 6:6:6:1, D= 6:8:8:1, E= 6:10:10:1 and F= 6:12:12:1, with columns two to five containing the mean percentage accuracy obtained using thresholds 0.25, 0.5 and 0.75. Column six contains the correlation coefficient (r) used to compare the observed and network results.

TABLE 5.3: MEAN PERCENTAGE ACCURACY FOR THRESHOLD CATEGORIES

Elman network performance					
Model	0.5	0.75	0.25; 0.5	0.5; 0.75	R
H	52.4	56.2	43.4	47.4	0.534
I	47.6	50.6	38.6	41.4	0.5245
J	51.6	56.4	41.8	46.6	0.3837
K	53.0	56.2	40.8	44.6	0.4487
L	50.8	56.6	37.6	45.8	0.5076
M	54.6	58.8	41.6	48.0	0.5406
N	55.2	54.8	46.4	48.4	0.3453

5.3 Comparison of Models

The ANN model produced overall results that were satisfactory for forecasting as shown in the tables above, but with poor results for a few models. These results for some of the models might have been caused by the fact that the daily rainfall variable was sparsely distributed in the data. Many researchers such as Varoonchotikul, Rientjes and Hsu [Rientjes and de Vos, 2005, Hsu and Sorooshian, 1995] used a combination of rainfall intensity, relative humidity, cloudiness and average hourly rainfall intensity in their studies. The models were evaluated using the following parameters as shown in tables above, correlation coefficient (r) and mean percentage accuracy.

The correlation coefficient is used in statistics to provide information in terms of the strength of a linear relationship between the simulated or forecasts and the observed values. In this study the correlation coefficient of observed rainfall data and simulated results were calculated for ANN model as shown in tables above. When the correlation coefficient value is between 1.0 and 0.7, it indicates a strong positive correlation; 0.3 and 0.1 indicates weak correlation, and less than 0.1 very little correlation at all. The same procedure also applies to negative values, when the value is less than -0.1 indicates no correlation, -0.3 and -0.1 moderate negative correlation, -0.7 and -1.0 strong negative correlation. The correlation coefficient obtained is negative, some are -0.7 and less confirming that the Elman models are acceptable for flood prediction.

5.3.1 Feed-Forward model comparison

Based on models A-G, the model D on average, yielded the best results with a correlation coefficient of negative value -0.987. It was also supported by the mean percentage accuracy of 50.4%. The second best model was C, followed by F. Model A yielded a moderate negative correlation coefficient of -0.579, -0.527, -0.519. The weak negative correlation coefficient -0.439, -0.313, -0.295, which indicated that poor performance was obtained by model B, followed by E and G respectively. The overall best results in terms of mean percentage accuracy was given by model E for which was used the threshold value 0.75 assigned for categories low and high flooding. Poor performance of the models are seen when using threshold values between 0.2, . , 0.5 and 0.5, . . , 0.75 respectively, assigned to determine the low, medium and high likelihood of flooding.

5.3.2 Elman model comparison

The best results were seen for model H with positive correlation coefficient value of 0.534 which was confirmed by the mean percentage accuracy value of 56.2% obtained using the threshold value 0.75. The models M, I and L yielded poorest performance with negative correlation coefficient values of -0.540, -0.524, -0.507 respectively. A few models K, J and N showed weak correlation with coefficient of -0.448, -0.383, -0.345 respectively. A poor performance were obtained for the models using threshold values between 0.2 . . . 0.5 and 0.5 . . . 0.75 respectively, assigned to determine the low, medium and high flooding likelihood.

5.4 Overall Model Performance

The investigation of this study has tried to establish an approach that would be used for flood modelling and early warning using daily rainfall data. The method of evaluation for performance reporting has been the same for all ANN models considered. When comparing the feed-forward and recurrent network models, the best percentage error for feed-forward networks (model A) was not as good as the best for recurrent models (H).

This is also supported by the correlation coefficient values. The effect of both model A and H is due to the number of hidden nodes, model H has extra hidden layers and hidden node than model A. This indicates that the number of hidden nodes in model A was insufficient to memorise and learn the pattern of training. The overall performance of model B and I shows more less the same results, although the number of hidden nodes for both was increased by two in each hidden layer. The improvement for both models B and I were minor. There was significant improvement in Elman models, the mean percentage accuracy increased as the number of nodes increased and showed improved results for forecasting.

On average for the feed-forward networks, the threshold category with threshold values of 0.5 and 0.75 produced the best performance. This is so because it determines only the low and high flooding likelihoods unlike the threshold of
0.25 . . . 0.5 and 0.5 . . . 0.75, which determines probabilities of low, medium and high flooding. The mean correlation coefficient indicates that the model is acceptable for prediction.

The objective of this study was to research and develop artificial neural networks that can be used as model to predict the onset of floods in a region of Msundusi River catchment. Several types of artificial neural network model were studied, including their architectures and variations of associated learning rules to determine the neural network parameters that provide the best prediction for impending floods.

Chapter 6

Conclusion and Extension

6.1 Conclusion

This research was aimed at the development of a computational intelligence model for improved prediction of floods in an area which is known to be at risk. We considered the application of artificial neural networks, a non-linear auto-regressive machine learning technique trained on patterns of preceding rainfall values in order to predict the likelihood of a flood. The chosen methodology was to evaluate the performance of various architectures in order to determine the most appropriate predictive model. Our approach differs from reported results in that it is parsimonious, for example the number of predictive variables was restricted to what is regarded as the most influential one, namely rainfall.

In comparing the preliminary performance results of the fourteen models, the recurrent neural network with more hidden nodes are shown to be the more effective neural network architecture for flood prediction. The best mean prediction obtained was 58.8% for the Elman network of two hidden layers containing two hidden nodes. The best accuracy result of the simulation experiments obtained, was 49% for the Elman network of two hidden layers containing twelve hidden nodes.

Overall, the results show that daily rainfall data variable can be used to predict impending floods. The results of the preliminary tests also indicated that the sigmoidal activation function was appropriate for the network architectures considered in this study. Publications for neural network flood prediction, employed dedicated sensor networks, involved a number of meteorological parameters or considered shorter predictive periods such as a few hours [Mandal 2005, Fan et al., 2002].

While the percentage accuracy of the results reported in this work is not as high as those reported in the literature, our approach is a reasonable technique to apply for flood prediction, if a limited number of variables are available.

6.2 Extensions

This study was confined to one type of feed-forward and one type of recurrent architecture. Several other neural network architectures, such as Jordan recurrent networks, could also be considered. It is also possible to vary the learning parameters such as momentum, and introduce other variations of learning such as simulated annealing. More research can still be done in order to improve the ANN flood prediction model results by investigating these different topologies and training algorithms. Furthermore, other machine learning algorithms for time series prediction, such as Hidden Markov Models and Support Vector Machines also warrant further study as flood prediction techniques.

Since short time intervals of days were considered, seasonality was not a concern. However, it is possible to consider other input and prediction time frames. More input parameters than five or six days may have a significant influence on the predictive performance.

While the limitation on the number of input variables was deliberate, it may be useful to explore various combinations of the most influential variables. Other studies indicate that hydrological parameters, such as run-off, also influence flooding and it would be useful to study the extent to which these parameters play a role in flood prediction.

Bibliography

[Abrahart, 2003] Abrahart, R. J. (2003). Neural network rainfall-runoff forecasting based on continuous resampling. Journal of Hydro informatics, 05 (1): 51–60.

[Abrahart and See, 2000] Abrahart, R. J. and See, L. (2000). Comparing neural network and statistics to parameter uncertainty in stream flow synthesis. Water Resources,
22 (3): 495–507.

[Ahmad and Ismail, 2004] Ahmad, A. B. M. and Ismail, S. (2004). Recurrent neural network with back propagation through time algorithm for Arabic recognition. In Proceedings of 18th European Simulation Multiconference. Graham Harton.

[Alexander, 2002] Alexander, W. J. R. (2002). The standard design flood. Journal of the South African Institution of Civil Engineering, 44(1):26–30. Accessed: 10 June
2009. Available: www.saice.org.za/Portals/0/pdf/journal/vol44-1-2002/Alexander2.pdf

[Alho, 2009] Alho, P. (2009). Flood modelling and mapping combining hydraulic modelling and GIS: characteristics of present and future flooding problems in Finland. Accessed: 28 July 2009. Available: www.sci.utu.fi/maantiede/en/staff/research_plan_abstract_applied.pdf

[Baoli et al., 2003] Baoli, L., Shiwen, Y., and Qin, L. (2003). An Improved k-Nearest Neighbor Algorithm for Text Categorization. In Proceedings of the 20th Internetional Conference on Computer processing of Oriental Languages.

[Barros and Kim, 2001] Barros, A. P. and Kim, G. (2001). Qualitative flood forecasting using multisensor data and neural networks. Journal of Hydrology, 246(1-4):45–62.

[Borga et al., 2008] Borga, M., Gaume, E., Creutin, J. D., and Marchi, L. (2008). Surveying flash floods: gauging the engaged extremes. Hydrological processes. Accessed: 23 May 2009. Available: www.interscience.wiley.com

[Bullinaria, 2005] Bullinaria (2005). Introduction to Neural Networks Course Material and Useful Links. Electronic publication. url = http://www.cs.bham.ac.uk/ jxb/inn.html

[Campolo and Andreussi, 1999] Campolo, M. and Andreussi, P. (1999). Forecasting river flow rate during low-flow periods using neural network. Water Resource,35:3547–3552.

[Campolo et al., 2003] Campolo, M., Soldati, A., and Andreussi, P. (2003). Artificial neural network approach to flood forecasting in River Arno/ Une approche base de Roseau de neurones artificiels pour la provision des crues du fleuve Arno. Hydrological Science Journal, 48(3):381–398. Accessed: 22 April 2009. Available: http://dx.doi.org/10.1623/hysj.48.3.381.45286

[Chan et al., 1993] Chan, K. H., Patterson, D. W., and Tan, C. M. (1993). Time Series Forecasting with neural nets: a comparative study. In Proceedings the international conference on neural network applications to signal processing, pages 269–274.

[Chen et al., 2002] Chen, Y., Dawson, C. W., Harpham, C., and Wilby, R. L. (2002). Evaluation of artificial neural network techniques for flow forecasting in the River Yangtze, China. Hydrology and Earth system sciences, 6(4):619–626.

[Collobert et al., 1995] Collobert, D. A., Bengio, S., and Fessant, F. (1995). Time Series Forecasting with neural nets: a comparative study. In Proceedings International Workshop Application Neural Networks to Telecoms, pages 308–315.

[Corne et al., 1997] Corne, S., Dougherty, M., Openshaw, S., and See, L. (1997). Some Initial Experiments with Neural Network Models of Flood Forecasting on the River Ouse. Received from author.

[Damle, 2003] Damle, C. (2003). Flood Forecasting Using Time Series Data Mining. PhD thesis, Engineering Department of Industrial and Management Systems Engineering College of Engineering University of South Florida.

[Dandy and Maier, 1996] Dandy, C. G. and Maier, R. H. (1996). The Use of Artificial Neural Networks for the Prediction of Water Quality Parameters. Water Resources Research, 32(4):1013–1022.

[Dandy and Maier, 2000] Dandy, C. G. and Maier, R. H. (2000). Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environmental Modelling and Software, 15:101–124. Accessed: 22 February 2010. Available: http://www.gpa.etsmtl.ca/cours/sys843/pdf/Maier2000.pdf

[Davey et al., 1997] Davey, N., Edwards, T., Frank, R. J., and Tansley, D. S. W. (1997). Traffic Trends Analysis using Neural Networks. In Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications, pages 157–164.

[Dorffner, 1996] Dorffner, G. (1996). Neural Networks for Time Series Processing. Neural network world, 4(96):447–468.

[DPLG, 2007] DPLG (2006/2007). Inaugural annual report. Report, National Disaster Management Centre, Department of Provincial and Local Government. Accessed: 20 September 2010. Available: http://www.info.gov.za/view/DownloadFileAction?id=85534

[DPLG, 2008] DPLG (2008). National disaster management centre: Flood awareness, department of provincial and local government, republic of south Africa. Electronic publication. Accessed: 21 August 2010. Available = http://www.preventionweb.net/files/.

[Emanuel, 2009] Emanuel, P. (2009). STATUS QUO REPORT Environmental Management Framework. Technical report, SRK Consulting. Accessed: 16 June 2009. Available: www.greennetwork.org.za/docs/376998_FinalSQR_June09.pdf

[Fahlman and Lebiere, 1991] Fahlman, E. S. and Lebiere, C. (1991). The Cascade- correlation Learning Architecture. Access: 13 October 2010. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.6421&rep=rep1&type=pdf

[Fan et al., 2002] Fan, Y., Wei, H. T., and Xu, W. (2002). Artificial neural network based predictive method for flood disaster. Journal of Hydrology, 42:383–390.

[Fawcett, 2004] Fawcett, T. (2004). ROC Graphs: Notes and Practical Considerations for Researchers.

[Frean, 1991] Frean, M. (1991). The Upstart Algorithm: A Method for Constructing and Training Feed forward Neural Networks. Neural Computation, 2(2):198–209. Access: 23 October 2010. Available: http://www.mitpressjournals.org/doi/abs/10.1162/neco.1990.2.2.198

[French et al., 1992] French, M. N., Krajewski, F. W., and R., C. (1992). Rainfall fore-casting in space and time using a neural network. Journal of Hydrology, 137(1-4):1–31. Access:

22 October 2010. Available: http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V6C-488G9JD-42&_user=958262&_coverDate=08%2F15%2F1992&_rdoc=1&_fmt=high&_orig=search&_origin=search&_sort=d&_docanchor=&view=c&_acct=C000049363&_version=1&_urlVersion=0&_userid=958262&md5=98c66a943a8130f27e0f0463055d4806&searchtype=a

[Fu, 1994] Fu, L. (1994). Neural Networks in Computer Intelligence. McGraw-Hill, Inc., University of Florida, Gainesville.

[Geman and Doursat, 1992] Geman, S. and Bienenstock, E. and Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. Neural Computation, 4:1–58. Massachusetts Institute of Technology.

[Hanavar et al., 1999] Hanavar, V., Parekh, R., and Yang, J. (1999). DistAI: An inter-pattern distance-based constructive learning algorithm. Intelligent data analysis, 3:55–73.

[Haykin, 1994] Haykin, S. (1994). Neural Networks A Comprehensive Foundation. McMillan College publishing company, Inc.

[Hazarika et al., 1979] Hazarika, M. K., Kafle, T. P., Sharma, R., Karki, S., Shrestha, R. M., and Samarkoon, L. (1979). Statistical approach to discharge prediction for flood forecasts using TRMM DATA. Accessed: 16 August 2010. Available: www.geoinfo.ait.ac.th/publications/Statisticalapproach_TRMM.pdf

[Hsu and Sorooshian, 1995] Hsu, K. and Gupta, H. V. and Sorooshian, S. (1995). Artificial Neural Network Modelling of the Rainfall-Runoff Process. Water resources Research, 31(10):2517–2530. Access: 15 October 2010. Available: http://www.agu.org/pubs/crossref/1995/95WR01955.shtml

[Hung et al., 2009] Hung, N. Q., Babel, M. S., Weesakul, S., and Tripathi, N. K. (2009). An artificial neural model for rainfall forecasting in Bangkok, Thailand. Hydrol. Earth Syst., 13:1413–1425. Accessed: 24 January 2010. Available: www.hydrol-earth-syst-sci.net/13/1413/2009/

[Hutchins, 1995] Hutchins, R. G. (1995). Neural network topologies and training algorithms in nonlinear system identification. Intelligent Systems for 21st Century, 3:2512–2515.

[IDP, 2002] IDP (2002). Integrated Development Plan. Technical report, The Msundusi Municipality. Accessed : 11 February 2009. Available: http://devplan.kzntl.gov.za/Municipal/IDPs/Msunduzi/Combinedfinalidp.htm

[Karlsson and Yakowits, 1987] Karlsson, M. and Yakowits, S. (1987). Nearest-neighbor methods for nonparametric rainfall-runoff forecasting. Water resources research, 23(7):1300–1308. Accessed: 28 September 2010. Available: http://www.agu.org/pubs/crossref/1987/WR023i007p01300.shtml

[Kjeldsen et al., 2001] Kjeldsen, T. R., Smithers, J. C., and Schulze, R. E. (2001). Regional flood frequency analysis in the Kwazulu-Natal province, South Africa, using the index-flood method. Journal of Hydrology, 225:194–211.

[Kumar et al., 2004] Kumar, D. N., Raju, K. S., and Sathish, T. (2004). River Flow Forecasting Using Recurrent Neural Networks. Water resources management, 18:143–161. Kluwer Academic Publishers, Printed in the Netherland.

[Kwok and Yeung, 1995] Kwok, T. Y. and Yeung, D. Y. (1995). Constructive Feed forward Neural Networks for Regression Problems: A Survey. Technical report hkust- cs95-43, Hong Kong University of Science and Technology Clear Water Bay, Kowloon.

[Lee, 1996] Lee, H. J. (1996). Regional Forecasting of hydrologic parameters. Master's thesis, The Faculty of the Fritz. And Dolores H. Russ College of Engineering and Technology Ohio University.

[Mandal2005, 2005] Mandal2005 (2005). A neural network based prediction model for flood in a disaster management system with sensor networks. XLRI Jamshedpur, School of Management.

[McCarthy et al., 2007] McCarthy, S., Parker, D., and Tapsell, S. (2007). Enhancing the human benefits of flood warnings. Nat Hazards, 43:397–414.

[McCulloch and Pitts, 1990] McCulloch, S. W. and Pitts, W. (1990). A logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biology, 52(1-2):99–115. Accessed: 21 September 2010. Available: http://www.springerlink.com/content/xw16676748262521

[Messner and Meyer, 2005] Messner, F. and Meyer, V. (2005). Flood damage, vulnerability and risk perception-challenges for flood damage research. In Flood Risk Management Hazards, Vulnerability and Mitigation Measures, Nato Science Series, pages 1–26. Springer Published.

[Minsky and Papert, 1972] Minsky, M. and Papert, S. (1972). Perceptrons, An introduction to computational geometry. bulletin of the American mathematical Society, 78(1):12–15. Accessed: 21 September 2010. Available: http://projecteuclid.org/DPubS/Repository/1.0/Disseminate?view=body&id=pdf_1&handle=euclid.bams/1183533389

[Nabhan and Zomaya, 1994] Nabhan, T. M. and Zomaya, Y. A. (1994). Toward generating neural network structures for function approximation. Neural Networks, 7(1):89–99. Accessed: 03 September 2010. Available: http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6T08-485RHMN-39&_user=6747651&_coverDate=12%2F31%2F1994&_rdoc=1&_fmt=high&_orig=search&_origin=search&_sort=d&_docanchor=&view=c&_searchStrId=1525093265&_rerunOrigin=google&_acct=C000049363&_version=1&_urlVersion=0&_userid=6747651&md5=f7aa664fbcded4f61415f37a228660db&searchtype=a

[Nelson, 2009] Nelson, A. S. (2009). Flood Hazards, Prediction and Human Intervention, Natural Disasters. Technical report, Tulane University. Accessed 18 March 2009. Available: www.delsiegle.com

[Nelson, 2010] Nelson, S. (2010). Natural Disasters EENS 2040 and EENS 6050. Tulane University, Dept. Earth and Environmental Sciences. Accessed: 15 September 2010. Available: http://www.tulane.edu/~sanelson/geol204/index.html

[NeuroAI2007, 2007] NeuroAI2007 (2007). Neuro AI - Intelligent systems and Neural Networks, Algorithms and Applications.Electronic publication. Accessed: 21 August 2010. Available: http://www.learnartificialneuralnetworks.com

[Openshaw and Openshaw, 1997] Openshaw, S. and Openshaw, C. (1997). Artificial Intelligence in Geography. London: John Wiley and Sons.

[Papoulis and Unnikrishna, 2001] Papoulis, A. and Unnikrishna, P. S. (2001). Probability, random variables and stochastic processes. McGraw-Hill Science/Engineering/- Math.

[Parson, 1999] Parson, C. S. (1999). Use of Artificial Neural Networks to facilitate hydrological modelling on the internet. PhD thesis, Development of an internet water- shed education tool (interwet) for the spring creek watershed of central pennsylvania. Accessed 17 May 2010. Available: http://www.interwet.psu.edu/chapter4.htm

[Reynolds, 2003] Reynolds, D. (2003). Value-Added Quantitative Perception Forecasts.

[Rientjes and de Vos, 2005] Rientjes, T. M. and de Vos, N. J. (2005). Constraints of artificial neural networks for rainfall-runoff modelling: trade-offs in hydrological state representation and model evaluation. Hydrology Earth Syst.Sci, 2:365–415.

[Rietjes and de Vos, 2008] Rietjes, T. M. and de Vos, N. J. (2008). Multi objective training of artificial neural networks for rainfall-runoff modelling. Water resources research,44.

[Rogers and Dowla, 1994] Rogers, L. L. and Dowla, F. U. (1994).Optimization of groundwater remediation using artificial neural networks and parallel solute trans-port modelling. Water Resources Research, 30(2):457–481.

[Rumelhart and McClelland, 1987] Rumelhart, D. E. and McClelland, L. J. (1987). Parallel Distribution Processing, volume 1 of Bradford Books. Bradford.

[Shamseldin and Ling, 1997] Shamseldin, A. Y. and O’Connor, K. M. and Ling, G. C. (1997). Methods for combining the outputs of different rainfall runoff models. Journal of Hydrology, 197:203–229.

[Smith, 2001] Smith, L. (2001). An Introduction to Neural Networks. University of Stirling. Accessed: 19 February 2009. Available: http://www.cs.stir.ac.uk/~1ss/NNIntro/InvSlides.html

[Sparks et al., 1998] Sparks, R. E., Nelson, J. C., and Yin, Y. (1998). Is this Naturalization of the Flood Regime in Regulated Rivers. Bioscience, 48(9):706–720. Flooding: Natural and Managed.

[Tingsanchali, 2009] Tingsanchali, T. (2009). Forecasting model of chao phraya river flood levels at Bangkok. Electronic publication. Accessed: 12 September 2009. Available: http://std.cpc.ku.ac.th/delta/conf/Acrobat/PapersEng/Volume2/TawatchaiPaperOK.pdf

[Toth et al., 2000] Toth, E., Brath, A., and Montanari, A. (2000). Comparison of short-term rainfall prediction models for real-time flood forecasting. Journal of Hydrology, 239:132–147. Accessed 12 March 2010. Available: www.elsevier.com/locate/jhydrol

[Toth et al., 2002] Toth, E., Brath, A., and Montanari, A. (2002). Neural networks and non-parametric methods for improving real-time flood forecasting through conceptual hydrological models. Hydrology and Erath system sciences, 6(4):627–640. Accessed 09 March 2010. Available: www.elsevier.com/locate/jhydrol

[Varoonchotikul, 2003] Varoonchotikul, P. (2003). flood forecasting using artificial neural networks. A.A. Balkema Publisher, a member of Swets and Zeitlinger. Accessed : 21 February 2009. Available: http://repository.tudelft.nl/assets/uuid:c03d4025-43d1-48b0-a330-3c054af0c3ba/FLOOD_FORECASTING_USING_ARTIFICIAL_NEURAL_NETWORKS.PDF

[Verdin, 2002] Verdin, J., editor (2002). A flood early warning system for Southern Africa. USGS, EROS Data, Pecora 15/Land Satellite Information IV/ISPRS Commission I/IFIEOS 2002 Conference Proceedings. Accessed : 13 February 2009. Available: http://earlywarning.usgs.gov/adds/pubs/Pecora/FloodEarlyWarningGAetal.pdf

[Wallingford, 2008] Wallingford, H.R. (2008). Integrating flood risk analysis and management methodologies. Electronic publication. Accessed August 2010. Available: www.floodsite.net Report No: T03-07-01

[Wikipedia, 2007] Wikipedia (2007). KnnClassification. Electronic publication. Last accessed: September 2010. Available: http://commons.wikimedia.org/

[Xue et al., 2000] Xue, M., Droegemeier, K. K., and Wong, V. (2000). The Advanced Regional Prediction System (ARPS) - A multi-scale non hydrostatic atmospheric simulation and prediction model. Part I: Model Dynamics and Verification. In Meteorology and Atmospheric Physics.

[Zealand et al., 1999] Zealand, C. M., Burn, D. H., and Simonovic, S. P. (1999). Short term stream flow forecasting using artificial neural networks. Journal of Hydrology, 214:32–48.

[Zurada, 1992] Zurada (1992). Artificial Neural Systems. West Publishing Company.