Development of neural networks in civil engineering applications

Zhang, Yun

2010

Zhang, Y. (2010). Development of neural networks in civil engineering applications.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/20870>

<https://doi.org/10.32657/10356/20870>

DEVELOPMENT OF NEURAL NETWORKS IN CIVIL ENGINEERING APPLICATIONS

ZHANG YUN

**SCHOOL OF CIVIL & ENVIRONMENTAL ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY**

2010

Development of Neural Networks in Civil Engineering Applications

Zhang Yun
B. Eng., M. Sc.

School of Civil and Environmental Engineering

**A thesis submitted to the Nanyang Technological University
in fulfillment of the requirements for the degree of
Doctor of Philosophy**

2010

ACKNOWLEDGMENTS

It is difficult to use a single page to express my appreciation to all those who have contributed to the success of my research over the last four years. Herein, I would like to take this opportunity to thank those whose contributions to the research were indispensable.

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Zhao Zhiye for his profound vision, his excellent guidance, patience and encouragement in conducting this research. His unwavering enthusiasm and interest in scientific research is much appreciated and unforgettable. It has been a great privilege and honor to work with him.

I am much honored to be offered the PhD Scholarship by the School of Civil and Environmental Engineering (CEE), Nanyang Technological University (NTU). My experience as a graduate student at NTU has been pleasurable and stimulating, thanks largely to the remarkable faculty and the outstanding academic environment that the institution offers. It would not be possible to complete this research work without the love and support of all my friends in and outside of NTU.

Finally, I must acknowledge the continual support and encouragement provided by my family in China and Singapore for their patience. Most of all, I would like to express my deepest appreciation to my beloved husband, Dr. Wang Chao, for his love, encouragement, understand, help and many sacrifices, all of which made this effort possible.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	ii
TABLE OF CONTENTS	iii
SUMMARY	viii
LIST OF TABLES	x
LIST OF FIGURES	xii
CHAPTER 1 INTRODUCTION	1
1.1 OVERVIEW	1
1.2 OBJECTIVES AND CONTRIBUTIONS	4
1.3 ORGANIZATION OF THE THESIS	5
CHAPTER 2 NEURAL NETWORK.....	8
2.1 BIOLOGICAL NEURAL NETWORK (NN)	8
2.2 MATHEMATICAL MODEL OF ARTIFICIAL NN UNIT	9
2.3 NETWORK ARCHITECTURE.....	12
2.3.1 Single-Layer and Multi-Layer Networks	12
2.3.2 Feed-Forward and Feedback Networks.....	13
2.4 NETWORK LEARNING.....	14
2.4.1 Learning Strategies.....	14
2.4.2 Error Correction Learning Rule	16
2.4.3 Learning, Generalisation and Overfitting.....	17
2.4.4 Bias and Variance.....	20
2.4.5 Stopping Strategies.....	21

2.4.6	Performance Measure	23
2.5	BACK PROPAGATION NNS	25
2.6	NETWORK PARAMETERS	27
2.6.1	Training, Test, Cross Validation Data Set	27
2.6.2	Number of the Hidden Layers and Number of the Hidden Nodes	29
2.6.3	Other Parameters.....	32
2.6.3.1	Activation Functions	32
2.6.3.2	Learning Rate	35
2.6.3.3	Momentum	35
2.6.4	Tackling a Problem with NN	35
2.7	PROPERTIES AND APPLICATIONS OF NNS	36
2.7.1	Properties of NNs.....	36
2.7.2	Applications of NNs and Their Future	38
CHAPTER 3 ENSEMBLE NEURAL NETWORKS		40
3.1	INTRODUCTION	40
3.1.1	Ensemble and Modular NNs	41
3.1.2	Bias, Variance and Covariance	42
3.1.2.1	Bias and Variance of the ENN	42
3.1.2.2	Covariance	44
3.2	ENSEMBLE METHODOLOGY	46
3.2.1	Creating Ensemble Members.....	47
3.2.2	Selecting Appropriate Ensemble Members	49
3.2.3	Combining Networks in an ENN.....	52
3.2.4	Important Factors for a Perfect ENN	54
3.2.5	Number of Examples in Training Set	55
3.3	RECENT DEVELOPMENTS IN ENNS AND THEIR LIMITATIONS....	56
3.3.1	Bagging	56
3.3.2	Boosting	57
3.3.3	Mixtures of Experts	58
3.3.4	Negative Correlation Learning	60

3.3.5	Recent Developments.....	61
3.4	PREDICTION MODEL OF TUNNEL BORING MACHINE PERFORMANCE BY ENNS	68
3.4.1	Introduction	68
3.4.2	Database Properties and Regression Analysis	70
3.4.3	ENN Modelling.....	73
3.4.4	Rock Mass Boreability Analysis by ENN.....	76
3.4.5	Conclusions	83
 CHAPTER 4 DESIGN OF ENSEMBLE NEURAL NETWORK WITH AKAIKE INFORMATION CRITERION 84		
4.1	INTRODUCTION.....	84
4.2	AKAIKE INFORMATION CRITERION IN MODEL SELECTION ..	86
4.3	DESIGN ENSEMBLE NNS USING AIC	88
4.3.1	Creating the Component Networks	88
4.3.2	Combining the Networks with AIC	90
4.3.3	Algorithm	93
4.4	COMPUTATIONAL EXPERIMENTS	95
4.4.1	Peak Function Approximation	95
4.4.2	Friedman Function Approximation.....	99
4.4.3	The Stress-Strain-Time Relationship Modeling of Mudstone	102
4.4.4	The Prediction of Peak Particle Velocity Damage Criterion for Rock Mass	107
4.5	CONCLUSIONS	114
 CHAPTER 5 DESIGN OF ENSEMBLE NEURAL NETWORK WITH THE ENTROPY 115		
5.1	INTRODUCTION.....	115
5.2	ENTROPY.....	119
5.3	NEWTON’S METHOD	121
5.4	DESIGN ENN USING ENTROPY AND NEWTON’S METHOD....	123

5.5	ALGORITHM.....	126
5.6	COMPUTATIONAL EXPERIMENTS.....	127
5.6.1	Peak Function Approximation	128
5.6.2	Friedman Function Approximation	134
5.6.3	The Prediction of Peak Particle Velocity Damage Criterion for Rock Mass.....	136
5.7	CONCLUSIONS.....	140
 CHAPTER 6 DESIGN OF ENSEMBLE NEURAL NETWORK USING THE AIC AND ENTROPY.....141		
6.1	INTRODUCTION	141
6.2	DESIGN ENN USING AIC-ENTROPY AND NEWTON'S METHOD	144
6.3	ALGORITHM.....	147
6.4	COMPUTATIONAL EXPERIMENTS.....	148
6.4.1	Peak Function Approximation	149
6.4.2	Friedman Function Approximation	155
6.5	CONCLUSIONS.....	160
 CHAPTER 7 CONCLUSIONS AND FUTURE WORK.....161		
7.1	SUMMARY	161
7.2	CONCLUSIONS.....	163
7.3	FUTURE WORK.....	164
7.3.1	Improve the Model Performance	165
7.3.1.1	Optimize the ENNs with AIC	165
7.3.1.2	Combine the Selected Networks with the Entropy Method	168
7.3.2	Model Application in Geological Problems	176
 PUBLICATIONS.....		177
 REFERENCES		178

APPENDIX I	TBM EXPERIMENT DATA	197
APPENDIX II	MUDSTONE EXPERIMENT DATA	199
APPENDIX III	PPV EXPERIMENT DATA	205
APPENDIX IV	MATLAB PROGRAM	207

SUMMARY

The architecture of neural networks (NNs) has a significant impact on a network's generalization ability. Given a learning task, an NN with too few parameters may not be able to perform the task, while an NN with too many parameters may overfit noise in the training data and fail to establish a good generalization. The design of NN architecture can be formulated as a challenging multi-criterion optimization problem.

Ensemble neural networks (ENNs) are commonly used networks in many engineering applications due to their better generalization properties. An ENN usually includes several back-propagation networks in its structure, where the back-propagation network is a single feed-forward network trained with the back-propagation learning rule.

In this thesis, the Akaike information criterion (AIC) and the entropy are used as the automating design tools for balancing the generalization against the parameters and finding the best combining weights of the ENNs. Two ENNs, namely, the AIC based ENN and the entropy based ENN are developed first. AIC is a performance measurement method that can balance the performance of the NNs and the complexity of the NNs. Entropy is another performance measurement method that can help the NNs to fit the data points by controlling both error and standard deviation. Since the AIC and entropy have their own merits for solving different problems, a new AIC-entropy based ENN is developed.

Two analytical functions – the peak function and Friedman function are used first to assess the accuracy of the proposed ensemble approaches. The computational

experiments conducted in this study have verified that the proposed ENNs have better performance than the simple averaging ENN, and the single NN alone. The verified approaches are then applied to the following engineering applications: 1) the penetration rate of tunnel boring machine with rock parameters; 2) the stress-strain-time relationship modeling of mudstone; and 3) prediction of peak particle velocity damage criterion for the rock mass.

LIST OF TABLES

Table No.	Description	Page
Table 3.1	Range of parameters	72
Table 3.2	TBM rock mass borability index (kN/cutter/mm/rev) modeling (20 runs).....	77
Table 4.1	Results of twenty runs on peak function with 3 component networks	99
Table 4.2	Results of twenty runs on Friedman #1 function with 3 component networks	102
Table 4.3	Results of twenty runs on mudstone deviator stress (kPa) modeling with 4 component networks	105
Table 4.4	Parameters for network and their range.....	109
Table 4.5	Weights of the component networks and β value in AIC-ENN in each run on PPV (m/s) modeling	112
Table 4.6	Results of twenty runs on PPV (m/s) modeling with 3 component networks	113
Table 4.7	Predicted threshold PPV (m/s) for granite rock mass	114
Table 5.1	Results of twenty runs in EN-ENN with different B0 and C0 in peak function.....	130
Table 5.2	Results of twenty runs on peak function with 4 component networks	131
Table 5.3	Weights of the component networks in EN-ENN in peak function	133
Table 5.4	Results of twenty runs on Friedman #1 function with 4 component networks	135

Table 5.5	Results of twenty runs in EN-ENN with different B0 and C0 in PPV (m/s) modeling.....	137
Table 5.6	Weights of the component networks in EN-ENN in PPV modeling	139
Table 5.7	Results of twenty runs on PPV (m/s) with 4 component networks	140
Table 6.1	Weights of the component networks in AIC-EN-ENN in peak function	153
Table 6.2	Results of twenty runs in AIC-EN-ENN with different B0 and C0 in peak function	154
Table 6.3	Results of twenty runs on peak function with 4 component networks	155
Table 6.4	Results of twenty runs on Friedman #1 function with 4 component networks	157
Table 6.5	Weights of the component networks in AIC-EN-ENN in Friedman function	159
Table 7.1	The number of the component networks in ENNs and weights of the component networks in EENN	174
Table 7.2	The number of the hidden nodes of the component networks in EENN.....	175
Table 7.3	Results of twenty runs on peak function in EENN	176

LIST OF FIGURES

Figure No.	Description	Page
Figure 2.1	A biological neuron.....	9
Figure 2.2	A mathematic model of a neuron model with the bias	10
Figure 2.3	The model of a neuron model with the threshold	11
Figure 2.4	Single-layer and multi-layer feed-forward networks	12
Figure 2.5	Feed-forward NN	13
Figure 2.6	Feedback NN.....	14
Figure 2.7	Supervised learning.....	15
Figure 2.8	Error correction learning rule.....	16
Figure 2.9	Different generalizations in a network.....	18
Figure 2.10	Under-trained and over-trained	19
Figure 2.11	The trade-off between bias and variance as a function of training time.....	21
Figure 2.12	A typical network training procedure	23
Figure 2.13	Step function	33
Figure 2.14	Linear transfer function.....	33
Figure 2.15	Sigmoid transfer function.....	34
Figure 3.1	Ensemble NNs.....	41
Figure 3.2	Modular NNs.....	42
Figure 3.3	Bagging	57
Figure 3.4	Boosting	58
Figure 3.5	A mixture of experts model	59
Figure 3.6	Flowchart of the proposed ENN	75

Figure 3.7	Comparison between the test measured and predicted specific rock mass boreability index	78
Figure 3.8	Comparison between the training measured and predicted specific rock mass boreability index	78
Figure 3.9	The influence of rock strength on specific rock mass boreability index	79
Figure 3.10	The influence of rock brittleness index on specific rock mass boreability index	80
Figure 3.11	The influence of volumetric joint count on specific rock mass boreability index	81
Figure 3.12	The influence of joint orientation on specific rock mass boreability index	82
Figure 4.1	Flowchart of AIC based ENN	94
Figure 4.2	Peak function	96
Figure 4.3	Comparison between the actual and predicted peak function test results of AIC-ENN	98
Figure 4.4	Comparison between the actual and predicted peak function training results of AIC-ENN	98
Figure 4.5	Comparison between the actual and predicted Friedman #1 function test results of AIC-ENN.....	101
Figure 4.6	Comparison between the actual and predicted Friedman #1 function training results of AIC-ENN.....	101
Figure 4.7	Strain softening behavior (Liao, et al., 1998).....	103
Figure 4.8	Modified Cam Clay model (Liao, et al., 1998)	103
Figure 4.9	Comparison between the measured and predicted mudstone modeling test results of AIC-ENN	106
Figure 4.10	Comparison between the measured and predicted mudstone modeling training results of AIC-ENN.....	106
Figure 4.11	Comparison between the measured and predicted PPV modeling test results of AIC-ENN	111
Figure 4.12	Comparison between the measured and predicted PPV modeling training results of AIC-ENN.....	111

Figure 5.1	Comparison between the actual and predicted peak function test results of EN-ENN	129
Figure 5.2	Comparison between the actual and predicted peak function training results of EN-ENN	130
Figure 5.3	Comparison between the actual and predicted Friedman #1 function test results of EN-ENN.....	135
Figure 5.4	Comparison between the actual and predicted Friedman #1 function training results of EN-ENN.....	136
Figure 5.5	Comparison between the measured and predicted PPV modeling test results of EN-ENN	138
Figure 5.6	Comparison between the measured and predicted PPV modeling training results of EN-ENN.....	138
Figure 6.1	Comparison between the actual and predicted peak function test results of AIC-EN-ENN	150
Figure 6.2	Comparison between the actual and predicted peak function training results of AIC-EN-ENN	151
Figure 6.3	Comparison between the actual and predicted Friedman #1 function test results of AIC-EN-ENN	156
Figure 6.4	Comparison between the actual and predicted Friedman #1 function training results of AIC-EN-ENN.....	157
Figure 7.1	Flowchart of AICENN	167
Figure 7.2	Flowchart of EENN	169

CHAPTER 1 INTRODUCTION

1.1 OVERVIEW

Artificial neural networks (NNs) have been widely studied for their ability to correctly learn the true distribution of the data from a sample. This ability is called generalization. NNs are a new generation of information progressing systems that are deliberately constructed to make use of some of the organization principles that characterize the human brain. They are parallel computational models comprised of densely interconnected adaptive processing units. The collective behavior of an NN, like a human brain, demonstrates the ability to learn, recall, and generalize from training patterns or data. The motivation behind combining a number of NNs is to improve upon a network's generalization performance.

Recently, NNs have been intensively studied with impressive successes across a wide variety of applications. These include pattern classification, speech synthesis and recognition, function approximation, combinatorial optimization, and non-linear system modeling and control (Fausett, 1994). Adeli and Yeh (1989) were the first to apply the NNs in civil and structure engineering. Since then, a large number of applications of NNs in civil engineering areas have been done. These applications include pattern recognition and machine learning in structural analysis and design, structure system identification, structural control, structural material characterization and modeling, construction scheduling and management, construction cost estimation, etc (Adeli, 2001).

Most of the work in the artificial NN literature concentrates on finding a single network. However, a single network found using the training set alone may not be the best network on the test set (i.e. it may not generalize well). In practical applications, a key problem is how to select an appropriate NN structure for a given task aiming to achieve the best performance. For complex problems, a too big structure of NN may induce NN over fitted, but a small structure cannot fit NN properly. It is a generally accepted fact that combining predictors can improve prediction performance and it has been widely investigated in the area of NNs recently. Multi-network systems can result in solutions to tasks which either cannot be solved by a single NN, or can be more effectively solved by a system of NN components. Ensemble NN (ENN) is one kind of the multi-network systems (Sharkey, 1999).

For a good ENN, the main idea is to create the component networks with both accuracy and diversity, then combine these component networks to obtain the good performance. Akaike information criterion (AIC) is a performance measure that can be used to help determine the best network size. It measures a network with low error but penalizes networks with a large number of free parameters. So, using AIC to determine the contributions of all the component networks and combining them is a feasible method.

In this thesis, the AIC concept has been used to investigate the performance of the ENNs and a new AIC based ENN that combined the different component networks by using AIC based weights is proposed. A peak function and the Friedman function are used first as the computational experiment to verify the accuracy of the proposed model. Then, the proposed ENN is applied to model the stress-strain-time relationship of mudstone and to predict peak particle velocity (PPV) damage

criterion for the rock mass. All examples show that the proposed AIC based method has better performance and is more accurate than other NN methods.

In 1948, Shannon (1948) introduced the concept of information entropy what is considered one of the most important achievements in communication systems. His work was the foundation of a new research area currently known as information theory. The entropy of a random variable is defined in terms of its probability distribution and can be shown to be a good measure of randomness or uncertainty. This appealing new branch of mathematics attracted several researchers that produced contributions both in theoretical and practical aspects. Although the information measures were originally adopted in communication systems, it is such a fundamental concept that it has been widely applied in areas such as physics, chemistry, computer science, neuroscience, economics, biology, psychology and linguistics.

In this thesis, an ENN that combined the different component networks by using entropy based weights was proposed. Entropy is a measure of the disorder or complexity of a system that can be used to help determine the best networks or reasonably combine the component networks. It measures a network performance by low error and low difference of the standard deviation between the output and target. Combined by the Lagrange's method and Newton's method, the entropy to determine the contributions of each component network is a feasible method. A peak function and the Friedman function are used first as the computational experiment to verify the accuracy of the proposed model. Then, the ENN is applied to predict PPV damage criterion for the rock mass. All examples show that the proposed entropy based method has better performance and is more accurate than other NN methods.

Compared the AIC based ENN to the entropy based ENN, each proposed ENN has its own advantages. AIC based ENN is able to balance the error and the number of the free parameters in the NN. The entropy based ENN focuses more on the performance of the NN. The peak function and Friedman function are used to compare them again. For peak function, the entropy based ENN is better than AIC based ENN, but for Friedman function, the AIC based ENN outperforms the entropy based ENN. So a new ENN method which adopts both AIC and entropy is proposed. And both examples show that the proposed AIC-entropy based ENN has better performance and is more accurate than other ENNs.

1.2 OBJECTIVES AND CONTRIBUTIONS

Generally, the performance of an NN mainly depends on its architecture. The optimal numbers of hidden neurons and optimal numbers of component networks are varied with the properties of the problem to be solved. Too few parameters may not be possible to learn the rule of the learning set, and too many parameters may fit not only the data points but also the noise in the data set which may lead to a bad generalization. So, the main objective of this study is to develop new ENN to avoid these shortcomings of the traditional ENNs. An optimal ENN should include fewer parameters (weights and biases) and better generalization performance that could be used as an alternative tool for practical civil engineering problems solving. The other objective of this study is to solve the realistic civil engineering problems, such as the penetration rate of tunnel boring machine (TBM) with rock parameters; model the stress-strain-time relationship of mudstone; and PPV damage criterion for the rock mass by using the proposed methods.

In this thesis, two theoretical functions – peak function and Friedman function have been used first as examples to test the proposed methods and followed by several

practical civil engineering examples. The present work has the following major contributions:

- (1) Develop a new weighting method based on the AIC for the ENNs, which has a better generalization performance than other popular ENN and the single NN.
- (2) Develop a new weighting method based on entropy for the ENNs, which has a better generalization performance than other popular ENN and the single NN.
- (3) Compare AIC based ENN to entropy based ENN, and develop a new weighting method by combining the AIC and the entropy for the ENNs, which has a better generalization performance than other ENNs.
- (4) Use the developed ENNs to solve complex civil engineering problems and numerical results are used to verify the effectiveness of these proposed ENNs.

1.3 ORGANIZATION OF THE THESIS

This thesis consists of seven chapters.

The first chapter presents a brief introduction to the research area, and lists down the objectives and the main contributions of this thesis.

In Chapter 2, a general description of NN is presented, followed by several most popular network models. The properties and applications of the single artificial NN are also provided in this chapter.

Chapter 3 gives an overview of the ENNs. A brief review of previous work on the

Chapter 1 Introduction

ENNs is included in this chapter. And the research in the area of the ENN design is also described and compared. The simple averaging ENN is applied to the prediction model of TBM. It offers an alternative tool to establish the relationship between those rock parameters, since it is difficult to build a pure mathematical model. In this chapter, simple ENN is developed to establish the complex nonlinear relationship between the specific rock mass boreability index and the four influencing rock mass properties: rock compressive strength, rock brittleness, joint spacing and joint orientation.

Chapter 4 develops the ENN model with the AIC. The detailed algorithm used to design the ENN with the AIC based weights is presented in this chapter. It also extends the proposed approach to computational experiments on data from the theoretical functions and practical examples. The first two examples are the peak function and the Friedman function with sufficient data; the others are mudstone modeling and PPV modeling with limited data. Several kinds of NNs are adopted to solve these problems respectively, including single NN, simple averaging ENN, and AIC based ENN. From these results, it can be found that the proposed AIC based ENN outperforms other methods.

Chapter 5 develops another ENN model with the entropy. The detailed algorithm used to design the ENN with based weights is presented. It also extends the proposed approach to computational experiments on data from the theoretical functions and a practical example, similar to Chapter 4. From these results, it can be found that the proposed entropy based ENN outperforms other methods.

Chapter 6 compares the AIC based ENN to the entropy based ENN, and develops another new ENN model by combining the AIC with the entropy. The detailed algorithm used to design the proposed ENN is presented. For peak function, the

Chapter 1 Introduction

entropy based ENN is better than AIC based ENN, but for Friedman function, the AIC based ENN is the better one. To utilize the advantage of both AIC and entropy, a new ENN method combined AIC and entropy is proposed. And numerical simulations show that the proposed AIC-entropy based ENN has better performance and is more accurate than other ENNs.

Finally, Chapter 7 gives a brief conclusion of the present work and discusses the areas of the future works. The ENN modeling is built in three steps, create component networks, select component networks (can be combined to the first step), and combine them. In this thesis, more focus is on the new methods to combine the component networks, but AIC and entropy also can be used to select the component network. A new method by using entropy to select component network is presented. And the preliminary study shows that this method may be a feasible way to build a good ENN.

CHAPTER 2 NEURAL NETWORK

2.1 BIOLOGICAL NEURAL NETWORK (NN)

Artificial neural networks (NNs) draw much of their inspiration from the biological nervous system. It is therefore very useful to have some knowledge of the way this system is organized. The brain consists of a very large number of neurons, about 10^{11} in average. These can be seen as the basic building bricks for the central nervous system. The neurons are interconnected at points called synapses. The complexity of the brain is due to the massive number of highly interconnected simple units working in parallel, with an individual neuron receiving input from up to 10000 others.

A simplified neuron is represented in Figure 2.1. The dendrites carry incoming voltage impulses from neighbouring neurons to the cell body where they are integrated (summed up) and compared with a voltage threshold. If the summation exceeds this threshold, then the neuron will 'fire', causing the cell body to propagate an impulse along its axon to the dendrite of another neuron, otherwise, no impulse is generated. A neuron's cell body will often have many simultaneous incoming signals that can have either an inhibitory or excitatory effect on the summation process. Interneuron connections are mediated by electrochemical links between axon and dendrite, called synapses, which can alter the impulses passing through them (Campbell et al., 1999).

Figure 2.1 A biological neuron

2.2 MATHEMATICAL MODEL OF ARTIFICIAL NN UNIT

Originally, McCulloch and Pitts (1943) proposed a model based on simplified “binary” neurons, where a single neuron implements a simple thresholding function: a neuron's state is either “active” or “not active”, and this is determined by calculating the weighted sum of the states of neurons which it is connected to. For this purpose, connections between neurons are directed (from neuron i to neuron j), and have a weight (w_{ij}). If the weighted sum of the states of the neurons i connected to a neuron j exceeds some threshold, the state of neuron j is set to active, otherwise it is not.

So, when creating a functional model of the biological neuron, there are three basic components of importance. First, the synapses of the neuron are modelled as weights. The strength of the connection between an input and a neuron is noted by the value of the synaptic weight. An adder sums up all the inputs modified by their respective weights. This activity is referred as linear combination. Finally, an activation function controls the amplitude of the output of the neuron. The adder and the activation function model the actual activity within the neuron cell.

Mathematically, this process is described in Figure 2.2.

Figure 2.2 A mathematic model of a neuron model with the bias

In mathematical terms, the neuron can be described by the follow equations:

$$v_k = \sum_{j=1}^m w_{kj} x_j + b_k \quad (2.1)$$

$$y_k = \varphi(v_k)$$

where $x_1, x_2, \dots, x_m$ are the input signals; $w_{k1}, w_{k2}, \dots, w_{km}$ are the weights of neuron k connected with $x_1, x_2, \dots, x_m$ input signals; b_k is the bias; v_k is the linear combiner outputs; $\varphi(\cdot)$ is the activation function; and y_k is the neuron's output.

Each neuron has an internal state, called its activation, which is a function of its received inputs. Typically, a neuron sends its activation as a signal to several other neurons. And the same activation function is used for all neurons in any particular layer of an NN, although this is not required. The activation function (or transfer function) can be linear or nonlinear function such as binary function, piecewise-linear function, sigmoid function and Gaussian function, etc. For limiting the amplitude of the neuron output, the normalized amplitude of the neuron output is in the range $[0, 1]$ or $[-1, 1]$, depending on the activation function used in the neuron. The details of the activation functions can be found in Section 2.6.4.

The bias b_k is an external parameter of artificial neuron k . It may be adjusted as part of the network training. The bias has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative, respectively. Although this could be achieved simply by adding a constant input with an appropriate weight, often the bias is considered separately. A bias b_k acts exactly as a weight on a connection from a unit whose activation is always 1.

Note that the bias as just introduced is not the only possible way to incorporate this parameter into the unit characteristic. Instead of using a bias weight, some researchers adopted a fixed threshold θ (Figure 2.3) for the activation function. The role of the threshold is to lower the input to the activation function.

Figure 2.3 The model of a neuron model with the threshold

In mathematical terms, the neuron with threshold can be described by the follow equations:

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (2.2)$$

$$y_k = \varphi(u_k - \theta_k)$$

where θ_k is the threshold; and u_k are the linear combiner outputs.

2.3 NETWORK ARCHITECTURE

2.3.1 Single-Layer and Multi-Layer Networks

A layered NN is a network of neurons organized in the form of layers. In the simplest form of a layered network, there is only an input layer of source nodes that projects onto the output layer of neurons (Figure 2.4(a)). This kind of structure is called single-layer feed-forward network. The term single layer refers to the output layer of the computation neurons.

A multi-layer network is a network with one or more layers of nodes between the input units and the output units. Any layer between the input and output layers is called hidden layer. The architecture of the multi-layer network is shown in Figure 2.4(b).

(a) Single-layer network

(b) Multi-layer network

Figure 2.4 Single-layer and multi-layer feed-forward networks

The multi-layer NN distinguishes itself from the single layer NN by the presence of one or more hidden layers, whose computational units are the hidden layers neurons

(nodes). In the architecture, the hidden layer neuron outputs are used as inputs to the third layer (output layer), as shown in Figure 2.4(b).

2.3.2 Feed-Forward and Feedback Networks

Network architecture is the arrangement of neurons into layers and the connection patterns within and between layers. Feed-forward networks and feedback networks are two types of the network architectures. A feed-forward network has a layered structure. Feed-forward NNs allow signals to travel one way only: from input to output. The output of any layer does not affect the same layer, i.e., there are no connections within a layer. The commonest type of feed-forward NNs consists of several layers—input layer, hidden layers and output layer. The activity of the input units represents the raw information that is fed into the network. The outputs of the network are generated from the output layer. As shown in Figure 2.5, the X_j inputs are fed into the *layer 1* of hidden units. The activation of a hidden unit is a function of the weighted inputs plus a bias. The output of the hidden units is distributed over the next layer of hidden units, until the *layer N-1* of the last hidden units, of which the outputs are fed into *layer N* of output units.

Figure 2.5 Feed-forward NN

Feedback architectures are also referred as interactive or recurrent. The architecture of a typical feedback NN is shown in Figure 2.6.

Figure 2.6 Feedback NN

A feedback network is a feed-forward NN plus a feedback loop which feeds the outputs of output layer neurons back to the input neurons. The feedback NN can be considered as a dynamic system.

2.4 NETWORK LEARNING

2.4.1 Learning Strategies

The NN learns from the input data and according with some measures, it adjusts its weights and bias in order to achieve a better performance. The method of determining the weights on the connections are called training, or learning. In general, there are two basic types of learning paradigms: supervised and unsupervised training. A supervised learning is the process that the weights are adjusted according to a learning algorithm. In a supervised learning, it is assumed that the correct “target” output values are known for each input pattern. The weights

are usually obtained by minimizing some error functions, which measure the difference between the “target” and the values computed by the NNs (Figure 2.7).

Figure 2.7 Supervised learning

In an unsupervised training, there is no teacher to provide any feedback information. The network must discover for itself the patterns, features, regularities, corrections, or categories in the input data and code for them in the output.

Another learning is reinforcement learning. This type of learning may be considered as an intermediate form of the above two types of learning. Here the learning machine does some action on the environment and gets a feedback response from the environment. The learning system grades its action good (rewarding) or bad (punishable) based on the environmental response and accordingly adjusts its parameters. Generally, parameter adjustment is continued until an equilibrium state occurs, following which there will be no more changes in its parameters. The self-organizing neural learning may be categorized under this type of learning.

Except these three types of the learning paradigms, the notion of multi-instance learning was proposed by Dietterch et al. (1977) in their investigation of drug activity prediction. In multi-instance learning, the training set is composed of many bags each containing many instances. If a bag contains at least one positive instance then it is labeled as a positive bag. Otherwise it is labeled as a negative bag. The labels of the training bags are known, but those of the training instances are

unknown. The task is to learn something from the training set for correctly labeling unseen bags. Due to its unique characteristics and extensive applicability, multi-instance learning has been regarded as a new learning framework parallel to supervised learning, unsupervised learning, and reinforcement learning (Maron, 1998).

2.4.2 Error Correction Learning Rule

There are many different kinds of learning rules used by NNs (such as error-correction learning, Hebbian learning, competitive learning and Boltzmann learning), but the most common one is called error-correction learning rule. The error correction learning procedure is simple enough in conception. The procedure is shown in Figure 2.8.

Figure 2.8 Error correction learning rule

During training, an input is put into the network and flows through the network generating a set of values $y_k(n)$ on the output neuron k . Then, the actual output is compared with the desired target $t_k(n)$, and error signal is computed as $e_k(n) = t_k(n) - y_k(n)$. The error signal actuates a control mechanism, based on which weights are adjusted. The adjustment of weights is designed to make the output signal comes closer to the desired target in a step-by-step manner, and until the weights are stabilized where the minimum $e_k(n)$ is obtained. At that point, the learning process finished. The minimization of the error function can be achieved by applying the well known delta rule of Widrow-Hoff rule (Widrow and Hoff, 1960).

Consider a single-layer NN with linear activation function where $w_{kj}(n)$ is the value of weight w_{kj} of the connection between neuron k and the element $x_j(n)$ of the input vector $x(n)$ at step n . The delta rule states that the adjustment $\Delta w_{kj}(n)$ applied to weight $w_{kj}(n)$ at time step n is defined by

$$\Delta w_{kj}(n) = \eta \cdot e_k(n) \cdot x_j(n) \quad (2.3)$$

where η is a positive constant determining the rate of learning when processing from one learning step to another. Parameter η known as the learning rate is one of the most important parameters in the stability of a closed-loop feedback system. The learning rate must be carefully chosen so that the convergence of the iterative learning process is achieved. After computing $\Delta w_{kj}(n)$, the update of the weight w_{kj} is obtained by

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \quad (2.4)$$

2.4.3 Learning, Generalisation and Overfitting

An appropriate architecture selected for the NN often needs process two phases, training and test. First, the number of neurons in the input layer should be the same as the number of input variables. The training set is then used to train the network using a suitable algorithm. This phase of the network design is also called training or learning. Second, the performance of the trained network is tested with the test set, which has examples that are not used in the training stage. This phase is called test or generalization. The term generalization is borrowed from psychology. Here it is assumed that the test data are drawn from the same population used to generate the training data.

The learning process may be viewed as a “curve-fitting” problem. The network itself may be considered simply as a nonlinear input-output mapping. The network performs useful interpolation primarily because multi-layer feed-forward networks also known as multilayer perceptron (MLP) with continuous activation functions lead to output functions that are also continuous. Figure 2.9 illustrates how generalization may occur in a hypothetical network. The nonlinear input-output

mapping represented by the curve depicted in this figure is computed by the network as a result of learning the points labelled as training data. The point marked on the curve as generalization is thus seen as the result of interpolation performed by the network.

Figure 2.9 Different generalizations in a network

An NN that is designed to generalize well will produce a correct input-output mapping even when the input is slightly different from the examples used to train the network, as illustrated in Figure 2.9. When, however, an NN learns too many input-output examples, the network may end up memorizing the training data. It may do so by finding a feature (e.g., noisy) that is present in the training data but not true of the underlying function that is to be modelled (the dash dot line in this figure). Such a phenomenon is referred to overfitting or overtraining. When the network is overtrained, it loses the ability to generalize between similar input-output patterns. Ordinarily, loading data into a MLP in this way requires the use of more hidden neurons than is actually necessary, with the result that undesired contributions in the input space due to noise are stored in synaptic weights of the network (Haykin, 1999). The dash line is shown as undertrained results. Since the training is not enough, the dash curve can not be fitted with the training data. To

avoid this phenomenon, adding the number of hidden nodes or increasing the training time may be a feasible way. The optimal trained curve is presented by the solid curve. It has been trained enough and not affected by the trivial features.

A typical sample of network training applied to the peak function (details can be found in Section 4.4.1) with a training set and a test set is presented in Figure 2.10. In this figure, the lower curve and the upper curve show the network performance on the training set and the test set, respectively. With the increasing of the epoch, the squared error of the test set reduces until it reaches to the optimal model at about 35 epochs.

Figure 2.10 Under-trained and over-trained

Before 35 epochs, the training performance and the test performance do not reach to best situation, neither. This is the under trained phenomenon. They still need more training time to arrive the optimal model. After 35 epochs, the test performance becomes worse, however, the training error continues to reduce. This tendency to

concentrate on the training data can cause the network to ignore other features of the data which would otherwise allow good generalisation (Geman et al., 1992). This is mainly due to either too many epochs performed in network training or too many hidden nodes used in the network architecture. In order to avoid overfitting, the networks should be trained neither for too many epochs nor with too many hidden nodes. Although several methods have been introduced to avoid overfitting, there is still no general advice be given when to stop the learning and the number of the hidden nodes are suitable for the network.

2.4.4 Bias and Variance

In general, there is bias-variance dilemma during training the network. The original decomposition (Geman et al., 1992) applies to quadratic loss error functions, and states that the generalisation error can be broken into two components: bias and variance. A bias is used to measure how close the average error of different training sets to its target value. A variance is to measure how stable the solution is. These two parameters are working as balance. Reducing the bias will increase the variance, and vice versa. An estimator with high variance will tend to produce wildly varying performance. Training an estimator for a long time tends to decrease bias, but slowly increase variance; at some point there will be an optimal trade-off that minimises the generalisation error. This is the bias-variance dilemma.

A typical training result is shown in Figure 2.11 (Gavin, 2004). Formally, the bias-variance decomposition can be presented as follows:

$$MSE(f) = \text{var}(f) + \text{bias}(f)^2 \quad (2.5)$$

Taking too much information from the training data or training more epochs can reduce the bias and make the network perform well but increase the variance. On the other hand, if the network performs badly on the training set (i.e., taking less data or training less epochs), the network might be under-trained. And an under-trained network also performs badly on the test set, since it is not able to fit the data (high bias). So the requirement of a good network (low variance, low bias) is to take sufficient data, but to avoid overfitting.

Figure 2.11 The trade-off between bias and variance as a function of training time

2.4.5 Stopping Strategies

There are a number of different ways to determine when to stop training (Zhang, 2000). The main ones are:

- The epoch/cycle control strategy, where the training will keep going until the training epochs/cycles reach a user defined number. The definition of the epoch will be presented in Section 2.5.
- The error control strategy, which uses the mean squared error (MSE) as termination criterion. When the MSE of the training set is smaller than a user defined value, training will be stopped. Sometimes, the sum squared error (SSE), the root mean squared error (RMSE) and the normalised error are also used for this purpose. The definitions of these errors will be presented in Section 2.4.6.

- The proportion control strategy. When the proportion of the number of patterns correctly classified among the number of total training set reaches a pre-defined percentage, the training will be terminated. A classification is regarded as correct if the output node with the largest activation value (actual output activation value) among all the output nodes is identical to that whose target output value is 1.0 (or 0.9). Otherwise, this pattern is incorrectly classified.

- The early stopping strategy. This strategy is used to avoid overfitting. In order to obtain good generalization, a third data set, the validation set, can be introduced (Zell et al., 1995) and the number of hidden nodes (and /or hidden layers) should be carefully determined. The determination of the number of hidden nodes will be described in Section 2.6.2. The validation set is separated from the training set, but it is used for network training. A typical sample of network training applied to the peak function (details can be found in Section 4.4.1) with a training set, a validation set and a test set is presented in Figure 2.12. In this figure, the curves show the network learning on the training set, the validation set and the test set, respectively. The training is stopped at the minimum of the validation set error, that is, when the MSE reaches 0.3 at about 41 epochs. At this point the network generalizes best and overtraining can be avoided.

Figure 2.12 A typical network training procedure

- The user control strategy, where the user/network trainer forces the training to stop in case the researcher thinks there is no need to continue the training.

2.4.6 Performance Measure

The main goal of network training is to optimize the generalization of the network by minimizing the network error. The error measured during network training or test can be the SSE (Judge et al., 1980), the MSE (Rumelhart et al., 1986), the RMSE (Rumelhart et al., 1986; Tortum et al., 2007), the normalized mean squared error (NMSE) (Pineda, 1988; Yu et al., 2005; Tortum et al., 2007), the normalised root mean squared error (NRMSE) (Dondeti et al., 2005; Rocha et al., 2007), the average inference error rate (AIER) (Kim et al., 2001; or is called as mean absolute percentage error (MAPE) in Chen et al., 2007) or the value of AIC (Tortum et al., 2007). These errors are defined according to Equation (2.6) to Equation (2.12), respectively.

$$SSE = \sum_{i=1}^K (T_i - P_i)^2 \quad (2.6)$$

$$MSE = \sum_{i=1}^K (T_i - P_i)^2 / K \quad (2.7)$$

$$RMSE = \sqrt{\sum_{i=1}^K (T_i - P_i)^2 / K} \quad (2.8)$$

$$NRMSE = \frac{RMSE}{\sum_{i=1}^K T_i / K} \times 100\% \quad (2.9)$$

$$NMSE = \frac{\sum_{i=1}^K (T_i - P_i)^2}{\sum_{i=1}^K (T_i - \bar{T})^2} = \frac{MSE}{\sigma^2(T)} \quad (2.10)$$

$$AIER = \frac{1}{K} \sum_{i=1}^K \frac{|T_i - P_i|}{T_i} \times 100\% \quad (2.11)$$

$$AIC = \log(MSE) + 2 \times (m + 1) / K \quad (2.12)$$

where K denotes the number of examples; P_i , T_i are the predicted and target values for the i th example, respectively; and m is the number of weights and biases used in the NNs (i.e., the number of parameters of the network model). For a good regressor, it should present a low error.

Typically, the error measured during training is the average SSE, i.e., MSE (Rumelhart, 1986). The MSE is calculated by dividing the SSE by the total number of training examples. In some circumstances it may be important to measure the error on a per-unit basis. In this thesis, MSE is adopted to measure the performance of the NNs.

To avoid the overfitting problem and improve the generalization of the NN, the regularization method can be used which modifies the error function E to be a linear sum of MSE and the mean squared network weights and biases as shown in the following:

$$E = \gamma MSE + (1 - \gamma) MSW \quad (2.13)$$

where γ is the performance ratio; MSE is the mean squared error; and MSW is the mean squared weights and biases (Zheng, 2009).

2.5 BACK PROPAGATION NNS

In the late 1950's, the perceptron (Rosenblatt, 1958) and MADALINE (Multiple ADaptive LINear Element) (Widrow and Hoff, 1960) were introduced that have had a great impact on current NN models. But the main drawback of these two NN models is their restriction to one layer of adaptive connections. The back propagation NNs (BPNNs) were proposed by Rumelhart et al. (1986) that could overcome those restrictions, which is similar to the work done by Werbos (1974), Le Cun (1985) and Parker (1985) earlier.

Although back-propagation can be applied to networks with any number of layers, it has been shown by Hornik et al. (1989), Funahashi (1989), Cybenko (1989) and Hartman et al. (1990) that one layer of hidden units suffices to approximate any function with finitely many discontinuities to arbitrary precision, provided the activation functions of the hidden units are non-linear.

The BPNNs are multi-layer feed-forward networks combined with a back-propagation-learning algorithm. Back-propagation is a specific technique for implementing gradient descent in weight space for a multi-layer feed-forward network. In BPNNs, the main idea is that the error for the units of the hidden layer is determined by back-propagating the error of units of the output layer, which is called back-propagation learning rule. The back-propagation algorithm or generalized delta rule is a rule based on error-correction learning rule. It is a gradient descent method to minimize the total squared error of the output computed by the network.

Usually, a network is trained over a number of training pairs, which can be considered as a set of ordered vector pairs $\{(I_1, t_1), (I_2, t_2), \dots, (I_p, t_p)\}$, where each I_i represents an input vector and each t_i represents the desired target vector

associated with the input vector I_i . The learning algorithm for the training of a BPNN is shown as follows:

1. Initialization: Decide the number of layers and neurons of BPNNs; initialize the weights and the thresholds to some random values. In general, the initial weights are small and uniformly distributed.
2. Forward propagation: Apply the i th input vector I_i to the input layer and specify the desired target output vector t_i . At each node, calculate the weighted sum of the training inputs and apply the appropriate activation function, calculate the actual output by proceeding forward through the network. The sigmoid function is a widely used activation function

$$\phi(v_j(n)) = y_j(n) = \frac{1}{1 + e^{-(v_j)}} \quad (2.14)$$

where v_j (as shown in Equation (2.1)) is the weighted sum of inputs coming to the j th node, and y_j is the output of the j th node.

For the output node, the error is computed straight forward as the difference between the desired target and the computed output:

$$e_j(n) = t_j(n) - y_j(n) \quad (2.15)$$

But for the hidden node, there is no target value given. The error has to be back-propagated from nodes further to the right (downstream) in the network:

$$e_j = \sum_{k \in \text{downstream}(j)} W_{kj} \cdot [e_k \cdot \phi'_k(v_k)] \quad (2.16)$$

3. Error back propagation: Propagate the errors backward to update the weights and adjust the weights by using the delta rule

$$\Delta W_{ji}(n) = \eta \delta_j(n) y_j(n) \quad (2.17)$$

Then calculate the weights as follow:

$$W_{ji}(n+1) = W_{ji}(n) + \Delta W_{ji}(n) + \beta [W_{ji}(n) - W_{ji}(n-1)] \quad (2.18)$$

where W_{ji} is the weight from i th node to the j th node; δ_j is the local gradient at the j th node, $\delta_j(n) = e_j(n)\phi'_j(v_j(n))$; η is the learning rate; and β is the moment. It is important to carefully select η to ensure stability or convergence of the iterative learning process. The choice of parameter η also has a profound influence on the accuracy and other aspects of the learning process. β can accelerate the learning process if the direction of the gradient does not change too much.

4. Loop: Steps 2 and 3 called one epoch. Repeat steps 2 and 3 for as many epochs, usually between 10 and 100 times, as it takes to reduce the SSE to a minimal value. If the training error is within a tolerance limit and weights stabilize, terminate the training process. The learning rate is typically decreased as the number of iterations increases.
5. Test: Substitute the test data into the network for test. Calculate the error between the actual output value and the target output value of test data.
6. Total error checking: If the error for test data is acceptable, output the final weights; otherwise, adjust the architecture of the network and initiate the new training epoch by going to step 1.

After sufficient iterations of steps 2 to 4, the BPNNs can successfully learn to replicate all the training output vectors given any of the input patterns. Then the learning stop and connection-weight values are frozen. The trained network is ready for their specific application.

2.6 NETWORK PARAMETERS

2.6.1 Training, Test, Cross Validation Data Set

The input signals are usually obtained by observations (measurements) of a task. These observations are often referred as examples. A set of examples is usually

divided into two subsets. One subset is called training set, and the other subset is called test set. The definitions of training and test have been presented in Section 2.4.3.

A benefit offered by NNs is their ability to generalise. Typically the goal of using an NN is to train the network with known data in order to have the network predict the outcome in cases for which the outcome is not currently known. A serious problem for training NNs for generalisation is the phenomenon of over-fitting. Over-fitting is the tendency for an NN to learn the training data so well that it achieves an extremely low value for the error function during training, but is incapable of predicting unseen data with any degree of accuracy. That is, a trained network often performs well on the training set, but it does not necessarily perform well on the test set. If a network performs well on the training set but performs very badly on the test set, the network might be over-trained or overfitted.

Since over-fitting is a problem when training NNs, it is important to determine the correct point to stop training. Probably the most commonly used technique is known as early stopping, or stopped training (Lawrence et al., 1996).

The default method for improving generalization is called early stopping. In this technique the available data is divided into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The third subset is the test set, which is used after the network is trained to measure the ability of the network to generalise. The test set error is not used during training, but it is used to compare different models. A typical sample of network training applied to the peak function with a training set, a validation set and a test set is presented in Figure 2.12 (details can be found in Section 2.4.5).

During the training process, the validation set is used to measure the quality of the network. The validation error normally decreases during the initial phase of training,

as does the training set error. However, when the network begins to overfit the data, the error on the validation set typically begins to rise. Training of the network is stopped when the error for the validation set reaches a minimum. The weights and biases at the minimum of the validation error are fixed. Finally, the test set is used to measure the accuracy of the network.

Several methods exist for estimating the generalisation error more accurately (these techniques however are not generally used in early stopping methods). Two of these methods are *k-fold* cross-validation and *leave-one-out* cross-validation. In *k-fold* cross-validation, the data is divided into k equal sized subsets. Training is conducted on all but one of the subsets of data; the remaining subset being used as the validation set. The network is trained k times, each time using a different subset of data as the validation set. Leave-one-out cross-validation is just *k-fold* cross-validation with k equals to the number of training examples in the training set.

2.6.2 Number of the Hidden Layers and Number of the Hidden Nodes

The size of a hidden layer is one of the most important considerations when solving actual problems using multilayer feedforward networks. The problem of the size choice is under intensive study with no conclusive answers available so far for many tasks. The exact analysis of the issue is rather difficult because of the complexity of the network mapping and due to the nondeterministic nature of many successfully completed training procedures. In this section, some guidelines are presented which may assist an NN modeller with a number of useful hints.

The number of hidden layers and their corresponding number of nodes are important parameters which determine the size of the NN. In most cases, one hidden layer is sufficient to compute arbitrary decision boundaries for the outputs. Others have used two hidden layers network architecture for more complicated applications (Baum and Haussler, 1989). The three layers feed forward NNs with one input layer, one output layer and one hidden layer is applied in this thesis. Thus,

the design of the network architecture becomes the determination of the number of input nodes, the number of output nodes and the number of hidden nodes. The number of input nodes and outputs are problem dependent.

Adaptive algorithms have been devised that either begin from a large network and successively remove some nodes and links until network performance degrades to an unacceptable level, or begin from a very small network and introduce new nodes and weights until performance is satisfactory; the network is retained at each intermediate state. Some researchers, e.g., Kung and Hwang (1988) and Sietsma and Dow (1988), have proposed penalties for choosing the number of hidden nodes.

Rules based on the degrees of freedom in the model have been proposed for selecting the topology of an MLP, e.g., the number of parameters in the network should be (significantly) less than the number of examples. A rule of thumb, known as the Baum-Haussler rule (Baum and Haussler, 1989), is used to determine the number of hidden neurons to be used:

$$N_{hidden} \leq \frac{N_{train} E_{tolerance}}{N_{pts} + N_{output}} \quad (2.19)$$

where N_{hidden} is the number of hidden neurons; N_{train} is the number of training examples; $E_{tolerance}$ is the error tolerance; N_{pts} is the number of data points per training example; and N_{output} is the number of output neurons.

In order to establish the number of neurons required, Kolmogorov's and Lipmann's approaches (Khaw et al., 1995) as shown respectively below can be used to set the lower and upper boundary level of the hidden nodes:

Lower bound of neurons in first hidden layer: $2N+1$

Upper bound of neurons in first hidden layer: $OP*(N+1)$

where N is the number of input neurons, which is dependent on input representation scheme used; and OP is the number of output neurons, which is set at three (for three operational policies).

Seibi and Al-Alawi (1997) suggested the following formula for calculating the appropriate number of hidden neurons to be used in a single hidden layer if the number of training pairs is known:

$$n = \theta \times (N_h \times (m + 1) + p \times (N_h + 1)) \quad (2.20)$$

where n is the number of training pairs available; θ is a constant greater than 1.0 (i.e., $\theta=1.25$ would give a 25% overdetermined approximation); N_h is the number of hidden neurons to be used in a network that has only one hidden layer; m is the number of input nodes; and p is the number of output nodes.

At present, in many applications of networks, the main method to select the number of hidden neurons is shown as follows. Firstly, a series of NNs with different number of hidden neurons are tested, and then their inductive errors are evaluated. Finally the best one is selected. This method is time-consuming and has great limitations. Some people also suggest that the maximal number of hidden neurons can be made according to the number of samples. This experiential method can only solve the problem of over-learning.

Li (1999) put forward a monotone index-based method to directly estimate the number of hidden neurons in a three-layer feed-forward network. This method requires a good set of sample set, and is not suitable for a small set of samples or multi-input and multi-output issues.

Yuan et al. (2003) proposed a new method for optimizing the number of hidden neurons based on information entropy. Firstly, an initial NN with enough hidden neurons should be trained by a set of training samples. Secondly, the activation values of hidden neurons should be calculated by inputting the training samples that can be identified correctly by the trained NN. Thirdly, all kinds of partitions should be tried and its information gain should be calculated, and then a decision tree for correctly dividing the whole sample space can be constructed. Finally, the important and related hidden neurons that are included in the tree can be found by searching the whole tree, and other redundant hidden neurons can be deleted. Thus, the number of hidden neurons can be decided.

These rules aim to prevent overfitting, but they are unreliable as the optimal number of parameters is likely to depend on other factors, e.g., the quality of the solution found, the distribution of the data points, the amount of noise, and the nature of the function being approximated. Specific rules, such as those mentioned above, are not commonly believed to be accurate (Lawrence et al., 1996). However, the stipulation that the number of parameters must be less than the number of examples is typically believed to be true for common datasets.

Hassoun et al. (1996) had stated that in the language of approximation theory (overfitting etc.) or statistical estimation (bias vs. variance) it was clear that too many parameters in some nonparametric models can be grievous, however with many NNs, more parameters can actually improve things. And they also stated that such phenomena which arise uniquely in NN applications should be more of a focus for statisticians rather than an anomaly to be ignored. The phenomenon had also been observed by others, e.g., Barron (1992) stated that overparameterized synapses give lower MSE and variance than exact order synapses.

2.6.3 Other Parameters

2.6.3.1 Activation Functions

Each neuron has an internal state, called its activation, which is a function of its received inputs. Typically, a neuron sends its activation as a signal to several other neurons. The activation function, denoted by $\phi(v)$, defines the output of a neuron in terms of the induced linear combiner output v . The details of the definition of the combiner output v can be found in Section 2.2. There are several basic types of activation functions are shown as following:

1. Step function

The step function (Equation (2.21)), which is known as Heaviside or threshold function, is shown in Figure 2.13.

$$\varphi(v) = \begin{cases} 1, v \geq 0 \\ 0, v < 0 \end{cases} \quad (2.21)$$

Figure 2.13 Step function

2. Linear transfer function

The linear transfer function (Equation (2.22)) is shown in Figure 2.14.

$$\varphi(v) = v \quad (2.22)$$

Figure 2.14 Linear transfer function

3. Sigmoid transfer function

The sigmoid function, whose graph is s-shaped, is by far the most common form of activation function used in the construction of artificial NNs. The sigmoid transfer function shown below takes the input, which may have any value between plus and minus infinity, and squashes the output into the

range 0 to 1 (Log-sigmoid function, Figure 2.15(a)), or into the range -1 to 1 (Tan-sigmoid function, Figure 2.15(b)). These transfer functions are commonly used in backpropagation networks, in part because it is differentiable.

Figure 2.15 Sigmoid transfer functions

An example of the sigmoid function is the logistic function defined by

$$\phi(v) = \frac{1}{1 + \exp(-av)} \quad (2.23)$$

where a is the slope parameter of the sigmoid function. By varying the parameter a , the sigmoid functions of different slopes can be obtained. In the limit, as the slope parameter approaches infinity, the sigmoid function becomes simply a threshold function. Whereas a threshold function assumes the value of 0 or 1, a sigmoid function assumes a continuous range of values from 0 to 1. Note also that the sigmoid function is differentiable, whereas the threshold function is not.

The hyperbolic tangent function has the activation function range from -1 to +1, which defined by

$$\phi(v) = \tanh(v) \quad (2.24)$$

In this thesis, tangent-sigmoid transfer function is adopted in the hidden nodes in the NN, the output nodes are used the linear transfer function. Normally, they are used in this way to the BPNNs.

2.6.3.2 Learning Rate

Standard backpropagation uses a single learning rate parameter, η . The value of the learning rate is problem dependent and is usually set by the practitioner during training experiments. This parameter constrains the step sizes that are taken in each component direction in weight space through the multiplication with a common constant. Principled methods exist for computing the single optimal learning rate for backpropagation and optimal learning rates for each weight, using Hessian information during training (Cun et al., 1998).

2.6.3.3 Momentum

The momentum is a constant which determines the effect of past weight changes on the current direction of movement in weight space. The gradient descent algorithm is generally very slow because it requires small learning rates for stable learning. The momentum variation is usually faster than simple gradient descent, because it allows higher learning rates while maintaining stability, but it is still too slow for many practical applications.

As with momentum, if the new error exceeds the old error by more than a predefined ratio, the new weights and biases are discarded. In addition, the learning rate is decreased. Otherwise, the new weights, etc., are kept. If the new error is less than the old error, the learning rate is increased.

2.6.4 Tackling a Problem with NN

Assuming that a generalized tool based on the BPNNs is available, the task of using BPNNs for any given problem becomes one of determining suitable network

architecture and a set of suitable parameters. More specifically, the following questions need to be considered before the experiments can be performed:

1. What kind of NN is used? How many training samples are required for successful learning and how large the test data set is for measuring the results?
2. How large an NN is required for a specific task? What is the number of output nodes? How many input nodes are needed? How many hidden layers are needed and how many nodes in each hidden layer?
3. And what kinds of activation function are used in the hidden layer and output layer? What values can be given for the parameters and variables for controlling the training process, for example, learning rate, range of initial weights, momentum and number of epochs?
4. What termination strategies need to be applied during network training and how many runs need to do perform for the problem?
5. What kind of learning strategies is adopted in this NN? At what stage are the network weights updated when training patterns are presented?
6. Is the cross validation used? How many samples in the cross validation data set?

2.7 PROPERTIES AND APPLICATIONS OF NNS

2.7.1 Properties of NNs

NNs are very good at a wide variety of problems, most of which involve finding trends in large quantities of data. They are better suited than traditional computer architecture to problems that humans are naturally good at, such as image

recognition, making generalizations, control etc. These wide applications are owed to the useful properties and capabilities of the NNs as following:

1. Nonlinearity: An NN is made up of an interconnection of neural which is basically a nonlinear device, so it is nonlinear itself.
2. Input-output mapping: The network learns from the examples through supervised learning by changing the synaptic weights of an NN in order to construct an input-output mapping for problem at hand. NNs are very flexible tools.
3. Adaptivity: NNs can automatically adjust their synaptic weights. In particular, an NN trained to operate in a specific environment can be easily retrained to deal with minor change in the operating environmental conditions. Moreover, when it is operating in a nonstationary environment, an NN can be designed to change its synaptic weights in real time.
4. Fault tolerance: An NN has the potential to be fault tolerant in the sense that its performance is degraded gracefully under adverse operating conditions (Bolt, 1992). The inherent fault-tolerance capability of NNs comes from the large number of connections providing much redundancy.
5. VLSI implementability: The massively parallel nature of an NN makes it potentially fast processing of certain tasks. So it is ideally suited for an NN using very-large-scale-integrated (VLSI) technology. The particular virtue of VLSI is that it provides a means of capturing truly complex behavior in a highly hierarchical fashion (Mead and Conway, 1980), which makes it possible to use an NN as a tool for real-time application.
6. Real time: NNs are parallel structures; if they are implemented in this way using computers or special hardware real time can be achieved.

7. Scalability: An NN can be easily ported to fit any problem from a particular problem area.
8. On-line: The response of the trained NNs is extremely fast.
9. Black box: There is not revealing deterministic physics behind.

NNs offer a different way to analyze data but they are not a solution for all computing problems, since the NNs are working as a “black-box” and need to learn by examples. Some problems such as processing data and track inventories can easily be handled and are suitable for traditional computing method, and do not require the special characteristics of NNs.

However, NNs are a kind of unstable learning methods, i.e., small changes in the training set and/or parameter selection can produce large changes in the predicted output. This diversity of NNs is a natural by-product of the randomness of the inherent data and training process, and also of the intrinsic nonidentifiability of the model. For example, the results of many experiments have shown that the generalization of a single NN is not unique. That is, the NN’s results are not stable. Even for some simple problems, different structures of NNs (e.g., different number of hidden layers, different hidden nodes and different initial conditions) result in different patterns of network generalization. In addition, even the most powerful NN model still cannot cope well when dealing with complex data sets containing some random errors or insufficient training data. Thus, the performance for these data sets may not be as good as expected (Naftaly et al., 1997; Carney and Cunningham, 2000).

2.7.2 Applications of NNs and Their Future

NNs are good at providing very fast, very close approximations of the correct answer. Although they are not as well suited as conventional computers for performing mathematical calculations or moving and comparing alphabetic

Chapter 2 Neural Networks

characters, NNs excel at recognizing shapes or patterns, learning from experience, or sorting relevant data from irrelevant data. Their applications can be categorized into classification, recognition and identification, assessment, monitoring and control, and forecasting and prediction. Among the tasks for which they are well suited are handwriting recognition, foreign language translation, process control, financial forecasting, medical data interpretation, artificial intelligence research, and parallel processing implementations of conventional processing tasks.

All current NN technologies will most likely be vastly improved upon in the future. For the civil engineering, NNs might allow: structural health monitoring; determining distribution of the rock parameters (such as seepage, joint, fracture, stress, etc.) in underground structures; modeling dynamic fracturing of rock under blast load; predicting seismic activity; structural failure analysis; and much more.

CHAPTER 3 ENSEMBLE NEURAL NETWORKS

3.1 INTRODUCTION

Ensemble neural network (ENN) is receiving increasing attention in recent NN research, due to its interesting feature. For complex problems, instead of constructing one large network, one may try constructing a number of smaller networks. The ensembles of NNs have some of the advantages of large networks without their problems of long training time and risk of overfitting. The NN ensembles are effective techniques to improve the generalization of an NN system. Combining a set of NN whose error distributions are diverse can lead to generating more accurate results than any single network (Krogh and Vedelsby, 1995). These networks may be trained separately and then combined. Ensemble and modular approaches are two kinds of methods to combine the NNs. In an ensemble, several redundant approximations to the same function are combined by some methods, while in a modular system the task is decomposed into a number of simpler components. In this chapter, a comprehensive review on the ENN is presented, and the standard ENN is employed to a practical application: prediction the penetration rate of tunnel boring machine with rock parameters.

When proposing a new ensemble approach, many researchers aimed to increase the accuracy of the ENN. This may induce to overfitting. So, this thesis is focused to

propose an effective, simple, accurate and stable ensemble approach to reduce overfitting, to partially avoid overfitting in the ENN. Usually, for the practically experiments or engineering problems, limited data sets are available for the analysis. So, how to obtain the reasonable results from these limited data is another issue that needs to deal with.

3.1.1 Ensemble and Modular NNs

The term ensemble is used to refer a set of combined redundant networks. In an ensemble combination, there are a finite number of NNs. Each NN in this ensemble network is referred to a component network. All component networks provide a solution to the same task. In other words, any one of the individual component network of the ENN could be used on its own to provide a solution or a predictor to the task, but better results might be obtained by an ENN through the different methods to combine the solutions or the predictors that are achieved by the component networks. The architecture of the ENNs is shown in Figure 3.1.

Figure 3.1 Ensemble NNs

The modular approach performs task decomposition and combines all the modules to form the complete task solution as shown in Figure 3.2. Different networks learn

from different training patterns and learn to compute different functions. Thus, the individual inputs are usually used in only one of the module networks. The architecture learns to partition a task into two or more functionally independent sub-tasks and allocates distinct networks to learn each subtask. In addition, the modular network tends to allocate to each task the network whose topology is most appropriate to that task.

Figure 3.2 Modular NNs

Both ensemble and modular combinations can exist at either a task, or a sub-task level. In fact, some multi-layer system could consist of a mixture of ensemble and modular combinations at different levels (Sharkey, 1999).

3.1.2 Bias, Variance and Covariance

3.1.2.1 Bias and Variance of the ENN

The ensemble error could be smaller than the single network error if the ensemble component NNs are accurate and diverse. This is proved by Krogh and Vedelsby (1995), who explained that the effect of ensemble combining had been expressed in terms of the variance and bias components of the ensemble error. This is due to each component network's error on an input would be compensated by a correct output

Chapter 3 Ensemble Neural Networks

made by the other networks. The accuracy of each component network is important to improve the accuracy of the ENN. But if all the component networks are exactly same as each other, there would be no advantage to including networks in an ENN. So both accuracy and diversity are required in an ENN. The detailed expression of this relationship in mathematics is stated as follows.

A dataset with pairs of data (x_i, T_i) , $i=1, \dots, N$, where x_i are input data and T_i are target values, is considered. The output of component network i on input x is $f_i(x)$; the output of ensemble network is $f_{ensemble} = \sum_i w_i f_i(x)$, where w_i is the weight of the i th component network and represents the contribution of this component network to the ensemble result. So, the weights must be non-negative, i.e., $w_i \geq 0$ and the weights sum to 1, i.e., $\sum_i w_i = 1$.

The diversity (Krogh and Vedelsdy (1995) refer to diversity as ambiguity) on input x of a component network i in the ensemble is $a_i(x) = [f_i(x) - \bar{f}(x)]^2$. So the average ambiguity of the ensemble network on input x is $\bar{a}(x) = \sum w_i a_i(x)$. The quadratic error of the i th component network is: $\varepsilon_i(x) = [T - f_i(x)]^2$. The average error of the component networks is: $\bar{\varepsilon}(x) = \sum w_i \varepsilon_i(x)$. The error of the ENN is: $e(x) = [T - \bar{f}(x)]^2$. Then the relationship can be obtained as follows:

$$e(x) = \bar{\varepsilon}(x) - \bar{a}(x) \quad (3.1)$$

This relationship of ensemble error is called error decomposition. From it, some conclusions can be drawn:

1. The ensemble generalization error $e(x)$ (as shown in Equation (3.1)) is always smaller than the average error of all individual component networks $\bar{\varepsilon}(x)$,

reducing the average error of the component networks can result to reduce the ensemble error.

2. Increasing the diversity $\bar{a}(x)$ can help to reduce the error of the ENN.

It is clear that both accuracy and diversity are necessary for a good ENN.

3.1.2.2 Covariance

The decomposition is extended to a bias-variance-covariance decomposition (Ueda and Nakano, 1996) when using an ensemble of regression estimators. Gavin et al. (2005) illustrated that the diversity cannot be simply maximized without affecting the other parts of the error for regression ensembles. For a single regression estimator, generalisation error is determined by a two-way bias-variance trade-off; for an ensemble of regression estimators, the ‘diversity’ issue is simply a three-way bias-variance-covariance trade-off.

Treating the ensemble as a single learning unit, the decomposition can be formulated as (Gavin et al., 2005)

$$E\{(\bar{f} - t)^2\} = \text{bias}(\bar{f})^2 + \text{var}(\bar{f}) = (E\{\bar{f}\} - t)^2 + E\{(\bar{f} - E\{\bar{f}\})^2\} \quad (3.2)$$

Redefine the random variable X as a set $X = (X_1, \dots, X_M)$, so the i th estimator is trained with a training set x_i drawn from its own random variable X_i . It should be noted that X_i potentially may be identical for all i , or not. If the training data is identical for two NNs i and j , it does not imply that the expected values $E\{f_i\}$ and $E\{f_j\}$ are equal, since other differences may be present between NNs i and j . Although the decomposition presented below does hold for non-uniformly weighted ensembles, Gavin et al. (2005) restricted their analysis to the uniform case, as it corresponds to the simple average combination technique used commonly in

Chapter 3 Ensemble Neural Networks

practice. They defined the averaged bias of the ensemble members as,

$bias = \frac{1}{M} \sum_i (E\{f_i\} - t)$; the averaged variance of the ensemble members as,

$var = \frac{1}{M} \sum_i E\{(f_i - E\{f_i\})^2\}$; the average covariance of the ensemble members as,

$cov ar = \frac{1}{M(M-1)} \sum_i \sum_{j \neq i} E\{(f_i - E\{f_i\})(f_j - E\{f_j\})\}$.

And then a bias-variance-covariance decomposition can be formulated as

$$E\{(\bar{f} - t)^2\} = bias^2 + \frac{1}{M} var + (1 - \frac{1}{M}) cov ar \quad (3.3)$$

It illustrates that in addition to the bias and variance of the individual estimators, the generalisation error of an ensemble also depends on the covariance between the individuals. This raises the interesting issues of why ensemble members need to be trained separately and why the effects of the covariance need to be considered in the error function.

Krogh and Vedelsby (1995) showed that at a single arbitrary data point, the quadratic error of the ensemble estimator is guaranteed to be less than or equal to the weighted average quadratic error of the component estimators,

$$(f_{en} - t)^2 = \sum_i w_i (f_i - t)^2 - \sum_i w_i (f_i - f_{en})^2 \quad (3.4)$$

where t is the target value of an arbitrary data point, $\sum_i w_i = 1$, $w_i \geq 0$, and f_{en} is the

convex combination of the M component estimators, and $f_{en} = \sum_{i=1}^M w_i f_i$. Preceding

the bias-variance-covariance decomposition, this was a very encouraging result for ensemble research, providing a very simple expression for the effect of error correlations in an ensemble. The decomposition is made up of two terms.

$\sum_i w_i (f_i - t)^2$ is the weighted average error of the individuals, and $\sum_i w_i (f_i - f_{en})^2$

is referred as the *Ambiguity*, which measuring the amount of variability among the ensemble member answers for this particular $(\mathbf{x}, t)$ pair. The trade-off between these two determines how well the ensemble performs at this data point.

After some manipulations, the expected value of the average individual error is

$$E\left\{\frac{1}{M} \sum_i (f_i - t)^2\right\} = \frac{1}{M} \sum_i E\{(f_i - E\{\bar{f}\})^2\} + (E\{\bar{f}\} - t)^2 = \Omega + \text{bias}(\bar{f})^2; \quad (3.5)$$

the ambiguity term is

$$E\left\{\frac{1}{M} \sum_i (f_i - \bar{f})^2\right\} = \frac{1}{M} \sum_i E\{(f_i - E\{\bar{f}\})^2\} - E\{(\bar{f} - E\{\bar{f}\})^2\} = \Omega - \text{var}(\bar{f}) \quad (3.6)$$

The interaction term, Ω , is present in both sides, and cancels out to allow the normal bias-variance decomposition of ensemble error. Thus, we have

$$\begin{aligned} \Omega &= \frac{1}{M} \sum_i E\{(f_i - E\{\bar{f}\})^2\} \\ &= \frac{1}{M} \sum_i E\{(f_i - E\{f_i\} + E\{f_i\} - E\{\bar{f}\})^2\} \\ &= \frac{1}{M} \sum_i E\{(f_i - E\{f_i\})^2\} + \frac{1}{M} \sum_i (E\{f_i\} - E\{\bar{f}\})^2 \end{aligned} \quad (3.7)$$

Equation (3.7) shows that the interaction term, Ω , is the average variance of the estimators, plus the average squared deviation of the expectations of the individuals from the expectation of the ensemble. The fact that the term Ω exists illustrates again that this interaction quantifies the diversity trade-off for regression ensembles (Gavin et al., 2005).

3.2 ENSEMBLE METHODOLOGY

Ensemble NNs are a learning paradigm where many NNs are used together to solve a particular problem. They are a powerful tool especially in dealing with complex problems. In fact, many different methods for ensemble NNs have been proposed.

In general, an ENN is constructed in three steps: creating ensemble members (i.e. component networks), selecting appropriate ensemble members and combining networks in an ENN. Some researchers combine the creating ensemble members and selecting appropriate ensemble members into the first step (Sharkey, 1999). In other words, the ENN can also be constructed in two steps: creating ensemble members and combining networks in an ENN.

3.2.1 Creating Ensemble Members

For creating ensemble members, good regression or classification component networks must be both accurate and diverse. There is no advantage to combining a set of networks which are identical; identical that is, in that they generalize in the same way. Therefore, how to generate diverse models is a crucial factor. To find networks which generalize differently, a number of training parameters can be manipulated. These parameters include the following:

- 1) Initial conditions, e.g., starting different weights for each component network,
- 2) The training data, e.g., randomly pick the training data from the data set to train each component network,
- 3) The typology of the networks, e.g., changing the different numbers of layers or different numbers of hidden nodes in each hidden layers,
- 4) The training algorithm, e.g., the back-propagation algorithm, radial basis function algorithm or Bayesian regression algorithm.

One of the most frequently used methods is varying the training data. There are a number of ways to alter the training data. For example, sampling data is a method to train each network on a different subsample of the training data. The sampling data has pairs of values, each pair including an input value and corresponding output value, the input value having regular intervals. Resampling method is a kind of sampling data. Resampling methods which have been used for this purpose include cross-validation (Krogh and Vedelsby, 1995), bootstrapping (Breiman, 1996), and smooth bootstrapping (Efron and Tibshirani, 1993); although in statistics the methods are better known as techniques for estimating the error of a predictor from limited sets of data. The perturbed data set may contain repeats. This procedure can be repeated several times to create a number of different, although overlapping, data sets. So for a small size of data, resampling data is useful. Using disjoint training sets is another similar method to the sampling data, but there is no overlap between the data used in the disjoint training sets. The problem is that the size of the training set may be reduced, and this may result in deteriorated performance (Tumer and Ghosh, 1996). Different data source is a method using data from different input sources to train the networks. And the data also can be trained after data have been used for different preprocessing methods, such as pruning methods, injecting noise or non-linear transformations (Sharkey, 1999).

Two of the most widely used techniques for creating the training data are Bagging and Boosting. The detail techniques of Bagging and Boosting will be presented in Section 3.3.1 and Section 3.3.2, respectively.

Another most frequently used method is changing the number of hidden nodes in different component networks. Zhao et al. (2008) proposed a simple procedure to define the number of hidden nodes in each component network. First, the best hidden node number in a single NN is worked out by the trial and error method.

Since the small test MSE and the small AIC value indicate sufficient training and proper number of parameters, to choose the best hidden nodes number is to find the single NN with the smallest test MSE and the smallest AIC value. And this number is selected as the maximum hidden nodes number of the component network. Then several other networks with the number of hidden nodes less than the maximum one are added. The gap of the hidden nodes number between any two component networks should be as big as possible to increase the diversity. And the minimum hidden nodes of component networks would be as small as possible but with sufficient accuracy. An upper boundary for the number of parameters that could be incorporated in the model is determined by the fact that it is not possible to determine more parameters than the number of samples in the data set (Ren and Zhao, 2002). The boundary for the number of hidden nodes is thus determined as (for 1 output node case):

$$N_h < (N_{tr} - 1)/(N_i + 2) \quad (3.8)$$

where N_i is the number of the input nodes of the component network; N_h is the number of the hidden nodes; and N_{tr} is the number of the training data. In this thesis, these two methods, which are different training data set and different number of hidden nodes in the component networks, are adopted simultaneously if the training data is enough, or adopted alternatively if using special method (e.g., AIC based method require the same training data).

3.2.2 Selecting Appropriate Ensemble Members

After training, each component network has generated its own result. According to the selective ensemble learning theory (Zhou et al., 2002), assembling many of the available NNs may be better than assembling all. If there are a great number of available individual NNs, it is necessary to select a subset of representatives in order to improve ensemble efficiency and have a good performance. In other words,

the total error of combined networks does not necessarily increase with a decrease of the number of individual network models. There are redundant or repeated (even noisy) information among the individual prediction. Furthermore, the redundant information often directly impacts on the effectiveness of the combined method.

One approach is to create a pool of networks and then to use selection criteria to pick the ensemble members from this pool. That is the approach taken by Perrone and Cooper (1993). Perrone and Cooper suggested a heuristic selection method whereby the population of trained nets are ordered in terms of increasing MSE, and an ensemble is created by including those with lowest MSE. The process can be further refined by constructing a small ensemble and then only adding a new component network if it results in a lower MSE for that ensemble.

In competitive combination, it is assumed that for each input only the most appropriate ensemble members will be selected based on either the inputs or outputs of the models (Sharkey, 1999). The two main methods for a competitive combination are: gating and rule-based switching. A gating model is used to output a set of scalar coefficients that weights the contributions of the various inputs. The details can be found in Section 3.3.3. In rule-based switching, the switching between the models can be triggered on the basis of the input or the output of one of the models. In such a rule-based switching (Ting, 1996), the control is switched between the ensemble members depending on the output of one of the members.

In order to show that those “many” NNs can be efficiently selected from all of the available NNs, the principal component analysis (PCA) technique (Yu et al., 2005) had been adopted to select appropriate ensemble members. The PCA technique is widely used in signal processing, statistics and neural computing. The basic idea in PCA is to find the components $(s_1, s_2, \dots, s_p)$ that can explain the maximum amount

Chapter 3 Ensemble Neural Networks

of variance possible by p linearly transformed components from data vector with q dimensions. The mathematical technique used in PCA is called eigenanalysis. In addition, the basic goal in PCA is to reduce the dimension of the data. Thus, one usually chooses $p \leq q$. Indeed, it can be proven that the representation given by PCA is an optimal linear dimension reduction technique in the mean-square sense (Jolliffe, 1986). Such a reduction in dimension has important benefits. If there is a dataset, PCA can determine the axes of maximum variance. For example, if the dataset was shaped like an egg, then the long axis of the egg could be the first principal component because the variance is greatest in this direction. All subsequent principal components are found to be orthogonal to all previous components.

Yu and Xi (2009) developed a selective NN ensemble approach called DPSOEN (Discrete Particle Swarm Optimization). In their model, DPSOEN will select several NNs from the “candidate” NNs to construct the NN ensemble. Particle Swarm Optimization (PSO) is a relatively new evolutionary algorithm that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems. PSO was originally developed by a social psychologist (James Kennedy) and an electrical engineer (Russell Eberhart) in 1995, and emerged from earlier experiments with algorithms that modelled the flocking behaviour seen in many species of birds. In the DPSO (Kennedy & Eberhart, 1997) algorithm, the particles are encoded as binary strings with length N wherein each element of each particle vector v indicates the presence (1) or absence (0) of “candidate” networks in the ensemble. The optimal solution can be obtained through evolving these particles.

In this chapter, a simple method is used to select the component network, randomly create several networks for each component network, and choose the best network

with the minimum MSE as the component network from the candidate networks. The details can be found in Section 3.4.3.

3.2.3 Combining Networks in an ENN

When a set of appropriate component networks has been created and selected, the method of combining these networks should be considered. From the beginning of the 90's, several methods have been proposed. These methods include linear combining methods, non-linear combining methods, and stacked generalization methods (Wolpert, 1992). Typically, linear ensemble strategy includes two approaches: the simple averaging approach (Hashem, 1993) and the weighted averaging approach (Perrone and Cooper, 1993). There are two types of weighted averaging: the MSE based regression approach and variance-based weighted approach (Krogh and Vedelsby, 1995).

The simplest combination strategy is to take a linearly weighted summation of the outputs of all component networks. The output of the ENN can be written

as: $f_{ensemble} = \sum_{i=1}^N w_i f_i(x)$, in which there are N component networks, $f_i(x)$ is the

output of the i th component network and w_i is the combination weight corresponding to the i th component network. These combination weights are non-negative: $w_i \geq 0$ and have the constraint to sum to one: $\sum_i w_i = 1$. Either uniform or non-uniform distribution can be used for these combination weights.

Hashem (1993) explained that linear methods were often simpler to analyze and easier to implement than non-linear methods. Hashem combined the component

Chapter 3 Ensemble Neural Networks

networks by using equal combination weights, $f_{ensemble} = \frac{1}{N} \sum_{i=1}^N f_i(x)$, where $w_i = \frac{1}{N}$. This set of outputs combined by a uniform weighting is referred as the simple ensemble (or simple averaging method). Perrone and Cooper (1993) demonstrated that this method could reduce MSE by a factor of N (the number of component networks), so the ensemble error will decrease with the increase of the size of ENN. It has the advantage that it is easy to understand and implement and can be shown not to increase the expected error. But in Zhou et al.'s (2002) opinion, assembling many of the available NNs may be better than assembling all of those networks. In other words, the total error of combined networks does not necessarily decrease with an increase of the number of individual network models.

Perrone and Cooper (1993) proposed a generalized ensemble method (GEM) to determine optimal weights using the correlation matrix C . They defined the symmetric correlation matrix as $C_{ij} \equiv E[m_i(x)m_j(x)]$, in which $m_i(x) \equiv T(x) - f_i(x)$ is the error between the target function $T(x)$ and the output of the component network $f_i(x)$. Then they showed the optimal weights are $w_i = \frac{\sum_j C_{ij}^{-1}}{\sum_k \sum_j C_{kj}^{-1}}$. Finally, the equation $f_{ensemble} = \sum_{i=1}^N w_i f_i(x)$ is adopted to obtain the ensemble solution. This ensemble method is sometimes called weighted averaging method, which can efficiently utilize local minima.

The nonlinear ensemble strategy is a promising approach for determining the optimal neural ensemble predictor's weight. Yu et al.(2005) proposed using a nonlinear ensemble forecasting model that combines the time series generalized linear auto-regression (GLAR) model, the artificial NNs model and the hybrid model to predict foreign exchange rates. Different from the previous work, Lai et al.

(2006) proposed a new nonlinear ensemble method with support vector machine regression (SVMR) principle (Vapnik, 1995). Chen et al. (2007) proposed a flexible neural tree ensemble technique, in which the output of ensemble is approximated by a local polynomial model. The details of the nonlinear combining methods can be found in Section 3.3.5

3.2.4 Important Factors for a Perfect ENN

The ENNs are more robust than a single network. Several works (Hashem, 1993; Perrone and Cooper, 1993; Krogh and Vedelsby, 1995; Zhao, 2006; Zhao et al., 2008) have shown that the network ensemble has a generalization error generally smaller than that obtained with a single network and also that the variance of the ensemble is lesser than the variance of a single network. But for designing a perfect ENN, some issues still need to be faced. Such as,

- How to determine the ensemble size;
- The method for designing and training the individual networks;
- The method of combining the individual networks;
- The mechanism for obtaining individual weights for each network if such is the case;
- The measures of performance of the individual networks;
- The methods for encouraging diversity among the members of the ensembles and how to measure such diversity;
- The method of selection of dataset that are used by each network to learn;
- Whether to include regularization terms and their form.

The design of NN ensembles implies making many decisions that have a major impact on the performance of the ensembles.

3.2.5 Number of Examples in Training Set

How many samples are needed for good training? This is a difficult question whose answer depends on several factors. A number of heuristic guidelines for determining the size of the training set have been proposed for a given size of the network.

A rule of thumb, obtained from related statistical problems, is to have at least five to ten times as many training samples as the number of weights to be trained. Baum and Haussler (1989) stated that the number of training examples for a network is approximately the number of weights in the network times the inverse of the accuracy parameter, shown as

$$P > \frac{|W|}{(1 - a)} \quad (3.9)$$

where P is the size of the training set; $|W|$ is the number of weights to be trained; and a is the expected accuracy on the test set. According to this heuristic guideline, about 10 times as many training examples as the weights in the network would be required if one wants to achieve an accuracy level of 90%.

This is the method based on the degrees of freedom to determine the size of the training set. In other words, the number of parameters in the network should be (significantly) less than the number of examples. If there is not enough training data, resampling is a better choice to increase the training data. The details of the resampling can be found in Section 3.2.1. If the training data is much more than the degrees of the freedom in the component networks, using part of the training data to train the network is also feasible, e.g., randomly pick certain percentage of the training data to train the different component network.

3.3 RECENT DEVELOPMENTS IN ENNS AND THEIR LIMITATIONS

3.3.1 Bagging

Two of the most widely used techniques for creating the training data are Bagging and Boosting.

Bagging (short for 'bootstrap aggregation') is proposed by Breiman (1996) based on bootstrap sampling (Efron & Tibshirani, 1993), in which "bootstrap" means that one available sample gives rise to many others by resampling (Figure 3.3). It adds a new training set with N size of data by sampling data uniformly at random N times from the training set data. During the sampling, the repeated data randomly picked can be used in the new training set. Then a component NN is trained with this new resample and this process is repeated until the component NNs are enough in the ENN. The resampled sets are often called bootstrap replicates. Finally, for any new test data, the bagged ensemble output is the simple average output of the component networks (Simple averaging method is introduced in Section 3.2.2). Such statistical resampling techniques are particularly useful where there is a shortage of data.

Figure 3.3 Bagging

3.3.2 Boosting

Bagging is always more accurate than a single network, it is sometimes less accurate than Boosting. Boosting (Figure 3.4) is proposed by Schapire (1990) and improved by Freund (1995). Comparing with Bagging that resamples the data set randomly with a uniform probability distribution, Boosting has a non-uniform distribution. It generates a set of component NNs whose training sets are determined by the performance of former component network. Since this method needs a large amount of data, Freund and Schapire (1996) proposed AdaBoost (adaptive boosting algorithm) to avoid this problem. In Adaboost method, samples are initially picked with replacement to train the component network in a manner similar to bagging. Subsequently, the probability of picking this pattern is adjusted down for “easy” patterns. Depending on how well this first weak learner performs on training pattern, the probability of picking this pattern as part of the training set for the next weak learner is adjusted to be lower or remains the same. It is adjusted to be lower if the predicted value is close to the target value in the case of regression. Thus, to increase the number of rounds of Boosting, more attention is paid to “hard” patterns. For any new test patterns, the Boosting ensemble output is the weighted averaging output of the component networks, the weight of a component network is

determined by its performance on its training data. Boosting builds an ensemble of weak learners, each new member of the ensemble conditioned on the performance of the previous built member of the ensemble. Sometimes Boosting can be less accurate than a single network. The reason is that Boosting ensemble network may overfit noisy data sets, thus decreasing its performance.

Figure 3.4 Boosting

3.3.3 Mixtures of Experts

The mixtures of experts architecture is a widely investigated paradigm for creating a combination of estimators (Jacobs et al., 1991; Jacobs, 1995; Jacobs and Tanner, 1999). The principle underlying the architecture is that certain estimators will be able to ‘specialise’ to particular parts of the input space. A gating network receives the same inputs as the component estimators, but its outputs are used as the combination weights. The gating network is responsible for learning the appropriate weighted combination of the specialized estimators (experts) for any given input. A graphical representation of the mixtures of experts’ model is shown in Figure 3.5. The model consists of a set of modules referred to as expert networks. Expert network i maps its input x to an output μ_i , and the total output of the model is given

by $\mu = \sum_i g_i \mu_i$.

Figure 3.5 A mixture of experts model

The expert networks map their inputs to their outputs in a two-step process. First, each expert multiplies the vector $\mathbf{x}$ by a vector of parameters u_i , and the resulting scalar is denoted as η_i : $\eta_i = u_i^T \mathbf{x}$. Second, η_i is mapped to the expert output μ_i by a monotonic, continuous function f . The gating network also forms its outputs in two steps. First, it computes the intermediate variables ξ_i as the inner product of the vector $\mathbf{x}$ and the vector of parameters v_i : $\xi_i = v_i^T \mathbf{x}$. Second, ξ_i is mapped to the gating output g_i . This mapping is performed by using a generalization of the logistic function: $g_i = e^{\xi_i} / \sum_j e^{\xi_j}$. Therefore, the probability distribution of the experts can be combined with its own prior distribution. The major difficulty with combining expert opinions is that these opinions tend to be correlated or dependent.

3.3.4 Negative Correlation Learning

Negative correlation (NC) learning (Liu, 1998) is an efficient ensemble training method which can easily be implemented on top of standard back-propagation in feed forward networks. It incorporates a measure of ensemble diversity into the error function of each network: thus each network not only decreases its error on the function, but also increases its diversity from other network errors. The procedure has the following form: take a set of NNs and a training pattern set, each pattern in the set is presented and back-propagated on, simultaneously, by the networks.

In the standard back-propagation algorithm, the error function for the output layer nodes is $\frac{1}{2}(F_i(n) - t(n))^2$, where $F_i(n)$ is the output of network i on pattern n , and $t(n)$ is the desired response for that pattern. In NC-learning, the error function becomes $\frac{1}{2}(F_i(n) - t(n))^2 + \lambda p_i(n)$ where $p_i(n)$ is $(F_i(n) - F(n)) \sum_{j \neq i} (F_j(n) - F(n))$

and λ is an adjustable strength parameter for the penalty. $F(n)$ is the output of the ensemble on pattern. A common ensemble output function is a simple average of the

networks in the ensemble, i.e., $F(n) = \frac{1}{N} \sum_{i=1}^N F_i(n)$. In this case they have an overall

error function of $\frac{1}{2}(F_i(n) - t(n))^2 - \lambda(F_i(n) - F(n))^2$. As can be seen from this equation, each network receives lower error for moving its response closer to the target response, and away from the mean response of all the other networks—this is a trade-off, controlled by the penalty strength parameter, λ . When $\lambda = 0$, the networks ignore the other errors, and this is termed independent training, equivalent to not using NC at all.

3.3.5 Recent Developments

The ENN originates from Hansen and Salamon's work (1990), which showed that the generalization ability of an NN system can be significantly improved through ensembling a number of NNs. Hansen and Salamon suggested training a group of NNs of the same architecture but initialized with different connection-weights. Then, a screened subset of the trained NNs is used in making the final classification decision by some voting schemes.

Wolpert (1992) used the term 'stacked generalisation' to refer to both the method of stacking classifiers, and the method of creating a set of ensemble members by training of different partitions of the data. Under stacked generalization, a nonlinear net learns how to combine the networks with weights that vary over the feature space. The outputs from a set of level 0 generalisers are used as the inputs to a level 1 generaliser, which is trained to produce the appropriate output. It is also possible to view other methods of combining, such as average, as instances of stacking with a simple level 1 generaliser. The same idea has been adapted to regression tasks, where it is termed as 'stacked regression'.

Krogh and Vedelsby (1995) proved that the ensemble error could be divided into a term that depends on the generalization errors of the individual networks and another term that contain all correlations between the networks. It was shown that apart from getting the component members of the ensemble to generalize well, it is important for generalization that the individuals' disagreement (called ambiguity or diversity) should be as much as possible. They also proved that the generalization error of the ensemble was always smaller than the (weight) average of the ensemble errors.

Optiz and Shavlik (1996) suggested that an effective ensemble should consist of a set of networks that are not only highly correct, but make their errors on different parts of the input space as well. They presented an algorithm, ADDEMUP (Accurate and Diverse Ensemble Maker giving United Predictions), that used genetic algorithms (GA) to generate a population of NNs which are highly accurate, while having minimal overlap on where they make their errors. It defines the fitness of each population member to be as follows:

$$Fitness_i = Accuracy_i + \lambda \cdot Diversity_i = (1 - E_i) + \lambda \cdot D_i, \quad \text{where } \lambda \text{ defines the}$$

tradeoff between accuracy and diversity, $1 - E_i$ is network i 's validation-set accuracy (or training-set accuracy if a validation set is not used). ADDEMUP adds new networks to the population and calculates their fitness, then prunes the population to the N most-fit members, which it defines to be its current ensemble, and repeats this process. In this approach, the output of the networks is obtained by taking a weighted sum of the output of each network, where each weight

$$w_i = \frac{1 - E_i}{\sum_k (1 - E_k)}$$

is based on the validation-set accuracy of the network. But this

approach can not make sure to obtain the correct answer, since sometimes GA will lose the useful information.

Rosen (1996) described a method that allows training an ENN by backpropagation, not only to reproduce a desired output, but also to have their errors linearly decorrelated with the other networks. Trained individual networks are then linearly combined. Networks are forced to be decorrelated with each other by means of a training algorithm that incorporates an error decorrelation penalty term designed to encourage networks to make errors which are decorrelated from those made by other nets. However, Rosen's algorithm still trains the individual networks sequentially. One major disadvantage of this algorithm is that training a network in

an ensemble cannot affect the networks trained previously in the ensemble, so that the errors of the individual networks are not necessarily negatively correlated.

Liu et al. (2000) presented evolutionary ensembles with negative correlation learning (EENCL) for designing NN ensembles automatically based on negative correlation learning and evolutionary learning. EENCL extended Rosen's work to simultaneous training of negatively correlated NNs. The idea of EENCL is to encourage different component NNs in the ensemble to learn different parts or aspects of the training data. EENCL introduces a correlation penalty term into the error function of each individual network in the ensemble so that all the networks can be trained simultaneously and interactively on the same training data set.

Compared with the early stopping method that is applied by monitoring the generalization capabilities of the previous-stage aggregate predictor plus the network being currently trained, Granitto et al. (2001) proposed the late stopping method, in which they retained the simplicity of independent network training and only the validation process becomes slightly more involved, leading in general to some controlled overtraining of the individual networks. Their method was a stepwise construction of the ensemble, where each network is selected at a time and only its parameters have to be saved. Thus, it largely reduces the computational burden compared to other algorithms like NeuralBAG (Carney and Cunningham, 1999) or the method (Naftaly et al., 1997) (which require saving the intermediate networks during training, since the selection of stopping points for the ensemble members is performed only at the end of all the training processes.). Later, Granitto et al. (2005) termed this algorithm as SECA (Stepwise Ensemble Construction Algorithm).

Zhou et al. (2002) analyzed the relationship between the ensemble and its

component NNs, which revealed that it may be a better choice to ensemble many instead of all the available NNs. This theory may be useful in designing powerful ensemble approaches. They presented GASEN (Genetic Algorithm based Selective ENsemble), which use GA to select a suitable subset of all the trained nets to build the ensemble. Compared with some popular ensemble approaches such as Bagging and Boosting, GASEN can generate NN ensembles with smaller sizes but stronger generalization ability. However GA needs more computational time and adds more random parameters on selection, crossover and mutation. Usually, the better results can be found by using GA, but it can not make sure to obtain the best one.

Ren and Zhao (2002) proposed a new algorithm which combines the AIC with the golden-section optimization technique to find the optimal architecture for single-hidden layer feedforward NNs. They have verified that the modified AIC criterion is in close agreement with the network generalization. It is observed that as long as proximity to global minimum solution is found for each configuration of the network, the AIC function of the networks over the whole domain is unimodal. The golden-section search method is very effective and computationally time-saving, especially for large size or complex problems. And the proposed optimization algorithm is applied to the modeling of the concrete strength under triaxial stresses.

Islam et al. (2003) proposed a new constructive algorithm, called constructive NN ensemble (CNNE), for training cooperative NN ensembles. It determined automatically not only the number of NNs in an ensemble, but also the number of hidden nodes in individual networks. CNNE adopted negative correlation learning to promote and maintain diversity among individual networks. The criteria for growing NNs and the ensemble are based on an NN contribution to reducing the ensemble's overall error, rather than in reducing its own error. CNNE performed very well for most problems, except for the letter recognition problem. But this

approach may induce the ENN model more complex and may not identify the optimal ENN.

Zhang and Zhou (2004) introduced the NN ensemble techniques to solve multi-instance learning problems. The notion of multi-instance learning was proposed by Dietterch et al. (1977) in their investigation of drug activity prediction. In multi-instance learning, the training set is composed of many bags each containing many instances. If a bag contains at least one positive instance then it is labeled as a positive bag. Otherwise it is labeled as a negative bag. The labels of the training bags are known, but those of the training instances are unknown. The task is to learn something from the training set for correctly labeling unseen bags. Due to its unique characteristics and extensive applicability, multi-instance learning has been regarded as a new learning framework parallel to supervised learning, unsupervised learning, and reinforcement learning (Maron, 1998).

Lagaros et al. (2005) proposed an adaptive strategy for NNs training. With the evolution strategies (ES) optimization procedure, the adaptive strategy improves the prediction reliability of an NN configuration substantially. The use of NN-based schemes to predict the outcome of constraints checks during the ES procedure can accelerate the overall optimization process. The proposed algorithm uses finite element (FE) to obtain the necessary input/output data for NN training and use the NN to predict the response of the structure in terms of objective and constraint functions' value. The proposed algorithm combining ES and NN provide high – quality results in affordable computing time.

Yu et al. (2005) proposed a PCA (principal component analysis) and ANN (artificial neural network) based nonlinear ensemble model for nonlinear combining forecasting. To summarize, the proposed nonlinear ensemble model consists of four

stages. Generally speaking, construct an original forecasting matrix according to the selected individual forecasts in the first stage. In the second stage, the PCA technique is used to deal with the original forecasting matrix and a new forecasting matrix is obtained. In the third stage, based upon the judgments of PCA results, the number of individual forecasting models is determined. And in the final stage, an ANN model is developed to ensemble different individual forecasts, at the same time the corresponding forecasting results are obtained.

Lai et al. (2006) proposed a new nonlinear ensemble method with support vector machine regression (SVMR) principle (Vapnik, 1995). Generally speaking, an SVMR-based nonlinear ensemble forecasting model can be viewed as a nonlinear information processing system that can be represented as:

$$y = f(x_1, x_2, \dots, x_n) \quad (3.10)$$

where $(x_1, x_2, \dots, x_n)$ is the output of individual NN predictors, y is the aggregated output, $f(\cdot)$ is nonlinear function determined by SVMR.

Zhao (2006) presented a strength model of steel columns under elevated temperatures using the ANN. The many influencing parameters make it difficult to build an analytical steel strength model. Being a flexible model building method, the ANN is an ideal tool to construct the complex relationship between the input and the output parameters accurately. A hybrid NN, which combines the sigmoid neurons and the radial basis function neurons at the hidden layer, is proposed to better map the input–output relationship both locally and globally. The use of the GA approach in searching the best-hidden neurons makes the hybrid NN less likely to be trapped in local minima than the traditional gradient-based search algorithms. The GA based hybrid NN is applied to model the strength of steel columns under fire.

Chapter 3 Ensemble Neural Networks

Chen et al. (2007) investigated how the seemingly chaotic behaviour of stock markets could be well represented using flexible neural tree (FNT) ensemble technique. The structure and parameters of FNT are optimized using genetic programming (GP) like tree structure-based evolutionary algorithm and particle swarm optimization (PSO) algorithms, respectively. A good ensemble model is formulated by the local weighted polynomial regression (LWPR) (Moore et al., 1997). In this paper, the final output of FNT ensemble is approximated by a local polynomial model, i.e., $f_{LWPR} = \sum_{i=1}^M \beta_i t_i(x)$, where t_i is a function that produces the i th term in the polynomial and β is the vector of weight term. For example, with two inputs and a quadratic local model there are six terms: $t_1(x)=1$, $t_2(x)=x_1$, $t_3(x)=x_2$, $t_4(x)=x_1x_1$, $t_5(x)=x_1x_2$, $t_6(x)=x_2x_2$.

Li et al. (2008) proposed a damage identification method based on the combination of ANN, Dempster-Shafer (D-S) evidence theory-based information fusion and the Shannon entropy, to form a weighted and selective information fusion technique to reduce the impact of uncertainties on damage identification. The initial damage decision is first made by several individuals. Optimal weighting coefficients obtained by GA, are assigned to ANNs. The decision obtained from the ANN with the largest weighting coefficient is the most reliable. Damage identification based D-S evidence theory is carried out by combining the decision of the ANN with the largest weighting coefficient with that of the ANN with the second largest weighting coefficient. Next, Shannon entropies of the decisions of the ANN with the largest weighting coefficient and that obtained by the information fusion are calculated to measure the uncertainty level of the decisions, respectively. The decision with smallest entropy remains in the next information fusion operation because this decision has less uncertainty. The decision with smaller entropy will be combined with the decision of the ANN with the third largest weighting coefficient. The

operation is repeated until the last ANN with the minimum-weighting coefficient is fused.

In summary, each method has its limitations, dependent either on the application areas it is suited to, or due to its effectiveness. It is worth mentioning that when a number of NNs are available, many ensemble approaches aim to reduce the MSE of each NN, this may result in overfitting. So, the main objective of this study is to look at the issues of how to reduce overfitting in the ENN. Most of the existing approaches can not effectively balance the accuracy and diversity of the ENN. Thus, how to effectively solve this issue is another objective of this study. Usually, if there is not enough data, many methods can not give satisfying results. So how to obtain the satisfying results of the given problem with limited data is also a problem we are aiming to solve.

3.4 PREDICTION MODEL OF TUNNEL BORING MACHINE PERFORMANCE BY ENNS

3.4.1 Introduction

With the development of tunnel boring machine (TBM), TBM excavation method is now widely used to excavate rock tunnels. Extensive studies have been carried out to establish the relationship between the rock mass quality and TBM performance. In general, the penetration rate primarily reflects the machine efficiency in rock breakage. Four parameters, namely the rock strength, the rock brittleness index, the joint spacing and orientation, are the main influencing factors on the TBM performance. The penetration rate primarily reflects the machine efficiency in rock breakage. The advance rate and utilization are relating to the suitability among the

Chapter 3 Ensemble Neural Networks

machines, the rock mass and the project management. The cutter wear depends on not only the rock mineral composition and texture, tunnel face variation such as mixed face, but also on the characteristics of cutters and the operating factors.

The planning of tunnel projects and selection of construction methods require effective prediction of TBM performance, so the prediction of TBM performance in a given rock mass has been an active research topic over the last decades. Due to the complexity of the interaction between rock mass and TBM, it is not possible to predict the TBM performance theoretically. During the last 30 years, the prediction models have been developed from a single factor model (Graham, 1976; Farmer and Glossop, 1980; Nelson, 1983; Hughes, 1986; O'Rourke et al., 1994) to multiple factors model (Rostami, 1997; Bruland, 1998; Nelson et al., 1999; Barton, 2000; Grima et al., 2000; Bruines, 2001), from simple parameter models like the prediction model based on the rock compressive strength by Graham (1976) to complex probability model by Nelson et al. (1999) and neural-fuzzy model by Grima et al. (2000) and Bruines (2001).

Another approach to predict the TBM performance is based on the rock mass classification system (Sapigni et al., 2002; Ribacchi and Fazio, 2005). Morimoto and Hori (1986) assessed the effect of rock mass on TBM excavation rate for the headrace tunnel of the Hayakido Hydro Power Station. Innaurato et al. (1991) utilized the uniaxial compressive strength (σ_c) and Rock Structure Rating (RSR) to predict TBM penetration rate. The analysis was based on 112 tunnel sections, which can be considered homogeneous with regard to TBM excavation. Laughton and Nelson (1996) classified rock masses into 36 classes, representing combinations of block size rank (four classes), degree of alteration or weathering (three classes), and level of water inflow at the heading (three classes). Alber (1996 and 2000) used rock mass strength (σ_{cm}) to evaluate the specific penetration rate

(mm/revolution/MN/cutter). Integrated Mass System (IMS) has been extensively applied to study correlation between TBM performance and rock mass classes in Hong Kong (McFeat-Smith and Askilrud, 1993; McFeat-Smith, 1999; Grandori et al., 1995). In all these rock mass classification systems, rock strength, joint spacing or frequency and joint conditions were used as the key influencing parameters.

The artificial NN is a computational model for information processing based on biological NNs. It has been successfully applied to a wide range of civil engineering applications, such as in fault detection (Jakubek and Strasser, 2004), concrete strength prediction (Jiang et al., 2003) and steel model under elevated temperature (Zhao, 2006). An ENN is a collection of a finite number of NNs that are trained for the same task. As compared with a single NN, the ENN can provide better generalization, especially when the training data is relatively small.

In this section, an ENN is used to model the boreability of the rock mass based on site information. As the data set is limited, the bootstrap sampling technique is used for the training of each component NN in the ENN structure. The ENN based rock mass boreability prediction model is compared with a nonlinear regression model which was derived from the same data set. The results demonstrate a good accuracy of the ENN model over the regression model.

3.4.2. Database Properties and Regression Analysis

Four rock mass parameters are identified as the influencing factors on the TBM penetration process: rock uniaxial compressive strength (UCS), rock brittleness (B_i), joint spacing (J_v) and joint orientation (α) (Gong, 2006). The rock uniaxial compressive strength affects the formation of crushed zone during TBM penetration process. The higher the rock strength, the more load is required on the cutter. The

Chapter 3 Ensemble Neural Networks

rock brittleness affects the crushed zone size and crack initiation and propagation in the rock indentation and chip formation. The higher rock brittleness results in easier rock fragmentation. Due to the existence of joints, the principal stress field is changed and the rock chipping process is affected. The increase of joint spacing results in the decrease of the penetration rate. The favorite joint angle between tunnel axis and joint plane for the penetration rate is 60° . With the increase or decrease of the angle from 60° , the penetration rate decreases. The rock mass boreability is expressed by the boreability index, defined as the ratio of the applied thrust per cutter to the penetration per revolution (Wanner and Aeberli, 1979; Hamilton and Dollinger, 1979; Sundin and Wanstedt, 1994). It is noted that the boreability index is a normalized cutter force by revolution per minute (RPM) and penetration rate. Thus, it facilitates to compare the performance of different TBMs.

Due to a change in the efficiency of the cutting action at the cutter head in different cutter forces, the boreability index is not a constant (Hamilton and Dollinger, 1979; Gong et al., 2007). Therefore, the boreability index can not represent accurately the rock mass boreability. Based on the in situ tests, a specific rock mass boreability index (SRMBI) (Gong et al., 2007) is defined as the boreability index at the critical point of penetration rate of 1 mm per revolution to stand for the rock mass boreability. It eliminates the influence of the cutter force variation on rock mass boreability. It can be obtained by combining the penetration rate with the TBM operation parameters.

Based on the extensive rock material strength experiments and tunnel face mappings from two Deep Tunnel Sewerage System (DTSS) in Singapore, the parameters that characterize the rock mass boreability were obtained. The rock masses are mainly composed of very closely to widely jointed granite rock masses with different weathering grades (Zhao et al., 2007). The corresponding TBM

Chapter 3 Ensemble Neural Networks

performance data and TBM specifications were collected and analyzed to calculate SRMBI. Although many tunnel face mappings have been conducted during tunnel construction, only part of these were used to establish the database because no corresponding core rock samples were obtained in some locations due to the time-consuming coring process. Some rock experiment results were not used either, because these samples failed along certain structural weak planes.

The input-output mapping for both the regression analysis and NN analysis is aimed to provide the following relationship:

$$SRMBI = f(UCS, J_v, B_i, \alpha) \quad (3.11)$$

The general range of the data corresponding to the five parameters is shown in Table 3.1.

Table 3.1 Range of parameters

	Minimum	Maximum	Data distribution
Orientation angle (α)	10	80	Evenly distributed
Joint spacing (Jv)	0	30	Joint count Jv < 16 for most data
Uniaxial Compressive strength (UCS)	100	260	Most data between 160 – 200 MPa
Rock brittleness (Bi)	8	22	Most data between 14 - 18
Specific rock mass boreability index (SRMBI)	110	240	Most data between 130 – 210 kN/cutter/mm/rev

A nonlinear regression analysis based on the 47 sets of data is carried out (Gong, 2006). The goal of nonlinear regression is to fit a model to these given data. The program finds the best-fit values of the variables in the model which can be interpret scientifically. The nonlinear regression follows the below steps:

- Start with an initial estimated value for each variable in the equation.
- Generate the curve defined by the initial values, and calculate the sum of the

squared residuals.

- c) Adjust the variables to make the curve come closer to the data points.
- d) Stop the calculations when the adjustments make virtually no difference in the sum-of-squares.
- e) Report the best-fit results.

The nonlinear regression analysis derived by Gong et al. (2007) gives:

$$SRMBI = 37.06 \cdot UCS^{0.26} \cdot Bi^{-0.10} \cdot (0.84e^{-0.06 \cdot J_v} + e^{-0.09 \cdot \sin(\alpha + 30)}) \quad (3.12)$$

The coefficient of determination (R^2) obtained is 74.94%. It indicates that the regression model can explain 75% of the total variation in the 47 data sets.

3.4.3 ENN Modelling

In this section, the back-propagation neural networks (BPNNs) are adopted as the component networks. The BPNNs were proposed by Rumelhart et al. (1986), and are multi-layer feed-forward networks combined with a back-propagation learning algorithm. In BPNNs, the main idea is that the error for the units of the hidden layer is determined by back-propagating the error of units of the output layer, which is called back-propagation learning rule. The back-propagation algorithm or generalized delta rule is a rule based on error-correction learning rule. It is a gradient descent method to minimize the total squared error of the output computed by the network. Due to the limited data available, the bootstrap sampling method is employed to create component NNs (Efron and Tibshirani, 1993), in which "bootstrap" means that one available sample gives rise to many others by resampling. It adds a new training set with N size of data by sampling data uniformly at random N times from the total training data. During the sampling, the repeated data randomly picked can be used in the new training set. The resampled

sets are often called bootstrap replicates.

In the proposed ENN, each component network is created several times, and the best weight configuration of this component network will be kept in the ENN. The criterion to choose the best weight configuration of the individual component network is the one with the smallest training MSE. Since good regression ensemble members must be both accurate and diverse, the training of each component network should also have high accuracy and diversity. Based on trial and error procedure, it is found that the BPNN with 2 hidden nodes provide in general the best generalisation, so all component networks have the same 2 hidden nodes.

The ENN algorithm (Figure 3.6) is shown as follows:

1. The data set is split into two parts: the training data and the test data set. The training data set is used for learning of the various component NN models. The test data set is not used in the network training, but used for test the performance of the trained ENN model.
2. Several component networks are used in the ENN. Each component network has one input layer with 4 input nodes, one hidden layer with 2 hidden nodes and one output layer with 1 output node.
3. Run each component network several times randomly. For each run, calculate the MSE of the component network by using the training data set. Then compare with each other to select the best weight configuration of the individual component network that has the smallest training MSE.
4. Combine the ENN with each best component network using the simple

average output of the component networks.

5. The MSE of the test data set will be used as the performance measurement of the ENN.

Figure 3.6 Flowchart of the proposed ENN

3.4.4 Rock Mass Boreability Analysis by ENN

An ENN program is created, and the training and test based on the 44 sets of site data are carried out to predict the specific rock mass boreability index. The detailed data refers to Appendix I. Out of the 44 sets of data, 36 sets are randomly selected as the training data, and the remaining 8 sets as the test data. The training data for each component NN is randomly selected based on the bootstrap sampling technique. For comparison purpose, three types of NNs are used: a single NN with 2 hidden nodes; an ENN with 4 component networks, and an ENN with 10 component networks.

Since the data sets are limited in this TBM example, more data sets are used as the training data sets. The sampling ratio used is set as 0.6. So in this case, the pool for all the training data sets is 36, but the data selected randomly to train each component network is 21. Since the hidden nodes in each component network are the same, the bootstrap sampling technique is used to randomly select the training data sets to increase the diversity of the component networks. In the other words, for the first component network, it randomly selects 21 training data set from the pool of 36 training data; for the second component network, another 21 training data sets are picked from the same pool to train the network; and so on. Since 4 and 10 component networks are used to ensemble the NN, all the training data sets are almost covered. Thus, 36 training data sets are used to determine the training comparisons between the measure data and the ENN prediction. Another 8 test data sets exclude of the training data set pool are used to test the ENN. So, the output results can reflect the distribution of the SRMBI.

Twenty runs are carried out for each type of NN, and their statistic results are

Chapter 3 Ensemble Neural Networks

summarized in Table 3.2. “Minimum” is the minimum MSE value of these twenty MSE of the results. “Mean” is the average MSE value of these twenty MSE. “S.D.” is the standard deviation of these twenty MSE. So, the lower value of the “Mean” stands for better performance of the ENN model. In Table 3.2, it can be observed that both ENNs have better accuracy than the single NN. The single NN is quite sensitive to the initial weights assigned for each run, which explains the large standard deviation for the single NN. With the increase of the component networks from 4 to 10, both the training and the test results have been improved, indicating a better generalisation. The corresponding R^2 for the Ensemble-4 is 0.76 for the training and 0.78 for the test, and R^2 for the Ensemble-10 is 0.78 for the training and 0.81 for the test. Both ENN results (R^2) are better than the nonlinear regression analysis which has the R^2 0.75. A typical NN test and training comparisons between the measure data and the NN prediction are shown in Figures 3.7 and 3.8.

**Table 3.2 TBM rock mass borability index (kN/cutter/mm/rev) modeling
(20 runs)**

		Single	Ensemble-4	Ensemble-10
Test-MSE	Minimum	123.12	71.68	71.63
	Mean	256.07	175.00	127.47
	S.D.	155.58	92.44	50.70
Train-MSE	Minimum	112.58	111.35	120.09
	Mean	246.43	167.67	140.43
	S.D.	190.08	28.43	8.32

Figure 3.7 Comparison between the test measured and predicted specific rock mass boreability index

Figure 3.8 Comparison between the training measured and predicted specific rock mass boreability index

To study the behavior of these parameters that govern the SRMBI, four parametric studies are performed. When the effect of one parameter on the SRMBI is under study, the other three parameters remain constant, so that the correlation between the studied parameter and the SRMBI can be obtained.

Figure 3.9 shows the relationship between the rock uniaxial compressive strength and the SRMBI. In general, the SRMBI increases with increasing UCS. The analysis shows that UCS has a significant effect on the specific rock mass boreability index. The relationship can be divided into three sections. The first section is for UCS less than 100 MPa, the SRMBI increases slowly within this range. For the second section with $100\text{MPa} < \text{UCS} < 200 \text{ MPa}$, there is a rapid increase in SRMBI. For the third section with $\text{UCS} > 200 \text{ MPa}$, the SRMBI increase slows down again.

Figure 3.9 The influence of rock strength on specific rock mass boreability index

When the rolling cutter indents the rock, the stress exerted must be higher than the

rock strength. Thus, the rock strength is directly relevant to the performance of TBM. Figure 3.9 shows that with increasing the rock strength the cutter indentation process gets more difficult.

The relationship between the brittleness index and the SRMBI is shown in Figure 3.10. The effect of rock brittleness index on TBM penetration process was studied by Gong and Zhao (2007). The brittleness index is defined as the ratio of rock compressive strength to tensile strength. From Figure 3.10, the SRMBI decreases with the increase in the rock brittleness index. The results show that with increasing rock brittleness index the cutter indentation process gets easier. And the analysis shows that the brittleness index has a greater effect on the SRMBI at higher values of UCS.

Figure 3.10 The influence of rock brittleness index on specific rock mass boreability index

Figure 3.11 shows the relationship between the volumetric joint count and the SRMBI. Joint spacing is normally measured along a specific direction (scan line) for all discontinuities and represented by the mean spacing of all discontinuities

along the scan line. The volumetric joint count (J_v) is defined as the sum of the number of joints per cubic meter for each joint set present. When J_v varies from 0 to 45, the SRMBI varies in a large range. With J_v more than 30, the effect of J_v on the SRMBI becomes less significant. From Figure 3.11, it can be found that with the increase of the J_v value, the cutters are easier to process.

Figure 3.11 The influence of volumetric joint count on specific rock mass boreability index

It can be observed from Figure 3.12 that the joint orientation angle (α) has relatively small influence on the SRMBI as compared with the other three parameters. With the increase of the angle from 0° to 90° , the SRMBI decreases at the beginning and then increases. It reaches the minimum value at the angle around 60° . So from Figure 3.12, it can be found that the easiest direction of the cutters process is the angle with 60 degree.

Figure 3.12 The influence of joint orientation on specific rock mass boreability index

Based on the ENN simulation results (Figures 3.9-3.12), the influence of rock mass parameters on the SRMBI can be summarized as follows:

1. UCS and J_v are the two most influencing rock mass parameters on the SRMBI.
2. The influence of B_i on the SRMBI is more significant for higher UCS (i.e. UCS=300).
3. α is the least influencing rock mass parameter on the SRMBI.
4. The smaller SRMBI (i.e. <140) can be achieved by the following parameters: $UCS \leq 200$, $J_v \geq 25$, $B_i \geq 15$, and $30 \leq \alpha \leq 90$.

3.4.5 Conclusions

Many engineering problems have complex relationship among various parameters, so it is difficult to build a pure mathematical model. NN offers an alternative tool to establish the relationship between those identified parameters. In this section, an ENN is developed to establish the complex nonlinear relationship between the specific rock mass boreability index and the four influencing rock mass properties: rock compressive strength, rock brittleness, joint spacing and joint orientation. Of the four rock mass parameters, rock uniaxial compressive strength and volumetric joint count have predominant effects on the SRMBI. The parametric studies based on the ENN are consistent with the situ measurements by Bruland (1998).

It should be noted that the statistical data are only collected from granitic rock masses in T05 and T06, Singapore. The machine configurations are fixed in these TBMs used in these two tunnels. Based on the above reasons, special attention must be paid when the model is used to predict the penetration rate in other rock masses or using different machine types. In this model, the effect of the in situ stress on the penetration rate is not taken into account.

CHAPTER 4 DESIGN OF ENSEMBLE NEURAL NETWORK WITH AKAIKE INFORMATION CRITERION

4.1 INTRODUCTION

The artificial NN is a mathematical or computational model for information processing based on biological NNs (McCulloch and Pitts, 1943). It is considered to be one of the intelligent tools to understand the complex problems. It has been successfully applied to a wide range of engineering applications, such as in prediction model of tunnel boring machine performance (Zhao et al., 2007), steel model under elevated temperature (Zhao, 2006), fault detection (Jakubek and Strasser, 2004), concrete strength prediction (Jiang et al., 2003), face recognition (Aitkenhead and McDonald, 2003); color adjustment (Puerto and Ghalia, 2002), injection molding control (Kenig et al., 2001) and bicycle derailleur control (Lin and Tseng, 2000).

The ENN originates from Hansen and Salamon's work (1990), which showed that the generalization ability of an NN system can be significantly improved through ensembling a number of NNs. Since this approach behaves remarkably well, the ENN has been applied to many areas, such as in pattern recognition (Giacinto and Roli, 2001), medical diagnosis (Hayashi and Setiono, 2002), climate prediction

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

(Cannon and Whitfield, 2002), marine propeller modeling (Reich and Barai, 2000).

In general, an ENN is constructed in three steps: creating component networks, selecting appropriate ensemble members and combining component networks into an ENN. For creating component networks, good regression or classification component networks must be both accurate and diverse. To find networks with different generalization ability, a number of training parameters can be manipulated, including the initial condition, the training data, the typology of the nets, and the training algorithm (Sharkey, 1999). After training, each component network can generate its own result. Select a number of component networks to combine may increase the performance of the ENN.

After a set of component networks has been selected, the methods to combine these networks have to be considered. From the beginning of the 90's, several methods have been proposed. Hashem (1993) provided a method to find optimal linear combinations of the members of an ensemble by using equal combination weights. Perrone and Cooper (1993) proposed a generalized ensemble method (GEM) to determine the optimal weights using the correlation matrix. Rosen (1996) described a method that allows training an ensemble of networks by backpropagation, and a penalty term is designed to force networks to be decorrelated with each other. These methods mentioned above are linear combining methods. The nonlinear ensemble strategy is a promising approach for determining the optimal neural ensemble predictor's weight. The details of the nonlinear combining methods can be seen in Section 3.3.5

It is worth mentioning that when a number of NNs are available, most of the ensemble approaches aim to reduce the MSE of each component NN, thus, they may lead to an ENN with unnecessary complexity and unstable performance. The

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

complexity of the ENN model may increase the computational time and lead to overfitting. This chapter aims to reduce the overfitting through the use of the Akaike information criterion. Two theoretical examples and two practical examples are used to demonstrate the accuracy of the proposed AIC based ENN approach.

The proposed method reduces each component network's error first, and then balances their contributions to the ENN by using the AIC based weights in this chapter. Since AIC is a performance measure, it also can be used to determine the number of the component networks and the best component network size. The key idea can be found in Section 7.3.1.1.

4.2 AKAIKE INFORMATION CRITERION IN MODEL SELECTION

The AIC, which was introduced more than 30 years ago by Akaike, is an information criterion for the identification of an optimal model from a class of competing models. The AIC belongs to the indirect approaches since it penalizes the model complexity. For a conventional least squares regression with normally distributed errors, one can compute the AIC with the following formula (where arbitrary constants have been deleted) (Akaike, 1973)

$$AIC = n \log(\hat{\sigma}^2) + 2K \quad (4.1)$$

where $\hat{\sigma}^2 = \sum \varepsilon_i^2 / n$; n is the sample size; ε_i is the estimated residual for a particular candidate model; and K is the number of estimated parameters in the model (i.e., number of variables and the intercept).

The AIC penalizes for the addition of parameters, and thus selects a model that fits well but has a minimum number of parameters. For small sample sizes (i.e., $n/K <$

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

40), Sugiura (1978) proposed AIC_c , which is a corrected version of AIC justified in the context of linear regression with normal errors. The AIC_c can be expressed as:

$$AIC_c = AIC + \frac{2K(K+1)}{n-K-1} \quad (4.2)$$

As the sample size increases, the last term of the AIC_c approaches to zero, thus the AIC_c will yield the same conclusion as the AIC (Burnham and Anderson, 2002).

Two of the most important properties of AIC when it is used for model selection are:

- 1) If only one model form is used, i.e., the number of parameters is fixed, then the AIC will result in the maximum likelihood estimator solution;
- 2) If the modeling is performed in two different model forms with the same values of the maximum likelihood estimator, the model with the smaller number of parameters will be selected.

Two measures associated with the AIC can be used to compare models: the Delta_AIC and Akaike_weights. The Delta_AIC (Δ_i), is a measure of each model relative to the best model, and can be calculated as

$$\Delta_i = AIC_i - \min AIC \quad (4.3)$$

where AIC_i is the AIC value for the i th model and $\min AIC$ is the AIC value of the best model.

The Akaike_weight (w_i) provides another measure of the strength of evidence for each model, and represents the ratio of Δ_i values for each model with respect to the whole set of m candidate models. The Akaike_weight can be expressed as

$$w_i = \frac{\exp(-\Delta_i / 2)}{\sum_{j=1}^m \exp(-\Delta_j / 2)} \quad (4.4)$$

In fact, Equation (4.4) has changed the scale of the Δ_i on a scale of 1 (i.e., $\sum w_i = 1$), where w_i represents the probability that the model is the best among the whole set of candidate models (Burnham and Anderson, 2002).

4.3 DESIGN ENN USING AIC

4.3.1 Creating the Component Networks

Creation of the component networks can be divided into two steps. The first step is to create the training data, the cross validation data and the test data, and the second step is to create the component networks.

For creating the data sets, some common ratios of training data to the test data and the cross validation data to the training data will be used in the analyses. The data selected uniformly or randomly are according to the problem properties. Since the AIC is adopted as a measurement of model accuracy, one should ensure that the same data set is used for each model, to make the AIC comparison meaningful.

In this research, BPNNs are adopted. The BPNNs are multi-layer feed-forward networks combined with a back-propagation learning algorithm. The back-propagation algorithm is a gradient descent method based on error-correction learning rule. Although back-propagation can be applied to networks with any number of layers, it has been shown by Hornik et al. (1989) and Hartman et al. (1990) that one layer of hidden nodes suffices to approximate any function with finitely many discontinuities to arbitrary precision, provided the activation functions of the hidden nodes are non-linear.

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

For selecting component networks, each component network is created several times, and the best weight configuration of this component network will be kept in the ENN. The criterion to choose the best weight configuration of the individual component network is the one with the smallest training MSE. Since good regression ensemble members must be both accurate and diverse, the training of each component network should also have high accuracy and diversity. Thus, different number of hidden nodes will be used in different component networks.

A simple procedure to define the number of hidden nodes in each component network will be used. First, the best hidden node number in a single NN is worked out by the trial and error method. Since the small test MSE and the small AIC value indicates sufficient training and proper number of parameters, to choose the best hidden nodes number is to find the single NN with the smallest test MSE and the smallest AIC value. And this number is selected as the maximum hidden nodes number of the component network. Then several other networks with the number of hidden nodes less than the maximum one are added. The gap of the hidden nodes number between any two component networks should be as big as possible to increase the diversity. And the minimum hidden nodes of component networks would be as small as possible but with sufficient accuracy. An upper boundary for the number of parameters that could be incorporated in the model is determined by the fact that it is not possible to determine more parameters than the number of samples in the data set (Ren and Zhao, 2002). The boundary for the number of hidden nodes is thus determined as (for 1 output node case):

$$N_h < (N_{tr} - 1) / (N_i + 2) \quad (4.5)$$

where N_i is the number of the input nodes of the component network; N_h is the number of the hidden nodes; and N_{tr} is the number of the training data.

One key decision in NN architecture is the selection of the numbers of the hidden

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

layers and hidden nodes in each hidden layers. If too many weights are employed, the NN may be over fitted, but fewer weights definitely cannot fit properly. Thus, the AIC can be introduced to assist in determining the best NN structure. The AIC measurement rewards a network with low MSE, at the same time penalizes networks with a large number of weights. For problems with a limited amount of data, this measurement can be used to determine the network that is least likely to over-fit the data. The idea is that the number of hidden processing elements should be adjusted in order to minimize the AIC value to find the best compromise between network structure size and training set error.

The formulas for the AIC value in ENNs are given in Equations (4.1) and (4.2), in which the K value can be directly determined by each component network structure. The K value is the number of the total parameters, i.e., the number of parameters in the NN model plus one for $\hat{\sigma}^2$. Since each hidden node and output node have a bias in each component network, the K value for each component network in the problem is defined as:

$$K = N_i \times N_h + N_h \times N_o + N_h + N_o + 1 \quad (4.6)$$

where N_o is the number of the output nodes. The first two terms are the number of the all weights in the component network, and the next two terms are the numbers of the all biases in this component network. Usually, there are several input nodes and one output node.

4.3.2 Combining the Networks with AIC

For the proposed ENN, each component network will be trained several times with the same training data set. And the AIC value of each best component network will be calculated.

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

In a regression analysis, the range of Delta_AIC (Δ_i) in general is quite small due to the small variation of the number of parameters. For typical regression models, a $\Delta_i < 2$ suggests substantial evidence for the model, $3 < \Delta_i < 7$ indicates that the model has considerably less support, whereas a $\Delta_i > 10$ indicates that the model is very unlikely. The best performance component network has the minimum AIC value. Since the delta_AIC value of the best performance component network is zero, delta_AIC can not be used to determine the weight in NN directly. Otherwise, the weight of the best component network is zero.

And the Akaike_weight (w_i) can be employed to combine the various regression models if necessary. However, for the NN models, due to the large amount of parameters, the range of Δ_i is usually very large, so the common rules used for the regression analysis can not be used directly. Otherwise, many component networks will have very small (approach to zero) weights, thus, the component networks with very small weight will lose their networks' contributions to the ENN. Therefore, a modified AIC measurement is proposed to determine the component network's weights in this section.

To consider the contributions of all the component networks to the ENN, a modified delta_AIC is proposed as follows:

$$(\Delta_m)_i = 1 + \frac{\Delta_i - \Delta_{\min}}{\Delta_{\max} - \Delta_{\min}} \cdot \beta \quad (4.7)$$

where $\Delta_{\min}$ and $\Delta_{\max}$ are the delta_AIC value of the best and worst component network, respectively; $(\Delta_m)_i$ is the modified delta_AIC value of i th component network; β is a constant. The value of β determines the range of the modified delta_AIC. A small β represents a more uniform Δ_m , distribution, whereas a large

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

β value corresponds to a more diverse Δ_m distribution. As β is a measure of diversity, the following formula is proposed to determine β :

$$\beta = \frac{Div_{\max}}{Div_{\min}} \quad (4.8)$$

where $Div_{\max}$ is the maximum diversity value of the component network and $Div_{\min}$ is the minimum diversity value of the component network. The diversity of the component network is calculated by training data. For each training data, the i th component network has the output O_i . Average all these different outputs ($\bar{O}$) of the different component network, then the diversity of the i th component network can be shown as follows:

$$Div_i = \frac{(O_i - \bar{O})^2}{N_{tr}} \quad (4.9)$$

The model with the smallest AIC value is the optimal model, thus, the weights of the component networks should be in inverse proportional to the modified delta_AIC values. According to all the weights sum to 1, i.e., $\sum_i \lambda_i = 1$ and each weight is not less than 0, i.e. $\lambda_i \geq 0$, the ENN weights are proposed as follows:

$$\lambda_i = \frac{\frac{1}{(\Delta_m)_i}}{\sum_{j=1}^m \frac{1}{(\Delta_m)_j}} \quad (4.10)$$

where λ_i is the weight of i th component network. When β equals to 0, a uniform $(\Delta_m)_i$ which represents the simple average ENN is obtained. When β approaches to infinity, only one $(\Delta_m)_i$ will be meaningful, which corresponds to the single NN.

4.3.3 Algorithm

The major steps of AIC based ENN are shown in Figure 4.1, which are explained further as follows.

1. The data set is split into three parts: the training data, the cross validation data and the test data set. The training data set is used for learning of the various component NN models. The cross validation data can be used to detect when overfitting starts during training of the NN. The test data set is not used in the network training, but used for testing the performance of the trained ENN model.
2. Three or four component networks are used in the ENN. Each component network has one input layer, one hidden layer and one output layer. The number of the input/output nodes is selected according to the problem's input/output attributes. The number of the hidden nodes usually needs to be optimized to get the best ENN. Each component network will be trained by the same training data set.
3. Run each component network several times randomly. For each run, calculate the MSE of the component network by using the training data set. Then compare with each other to select the best weight configuration of the individual component network that has the smallest training MSE.
4. Determine β value. Then calculate the modified delta_AIC value.
5. Construct the ENN with each best component network. Use the modified delta_AIC value of each component network to determine the weight of each component network in the ENN.

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

6. The MSE of the test data set will be used as the performance measurement of the ENN.

Figure 4.1 Flowchart of AIC based ENN

4.4 COMPUTATIONAL EXPERIMENTS

To verify the performance of the ENN proposed in this chapter, four computational experiments are carried out by an ENN program written in MATLAB. Two theoretical functions — the peak function and the Friedman function are applied first, and followed by two practical examples — the modeling of stress-strain-time relationship of mudstone and the prediction of the peak particle velocity (PPV) damage criterion for rock mass. For comparison purpose, a simple averaging ENN which has the same AIC based ENN structure and a single NN which uses the best hidden nodes number are also simulated using the same data.

4.4.1 Peak Function Approximation

The peak function, which is shown in Figure 4.2, is a sample function of two variables and obtained by translating and scaling Gaussian distributions. It is a typical complex two-dimensional function used as demonstration in MATLAB as following:

$$Z = 3(1-x)^2 e^{(-x^2-(y+1)^2)} - 10(x/5 - x^3 - y^5) e^{(-x^2-y^2)} - e^{(-(x+1)^2-y^2)} / 3 \quad (4.11)$$

Figure 4.2 Peak function

The peak function with normally distributed noise (mean 0, variance 0.05) is used to generate the training data, the cross validation data and the test data. First, 11×11 evenly distributed data along both the x-axis and the y-axis are selected from the domain $[-3, 3]$ as the training data for the simulation. Another two 10×10 evenly distributed points from the same domain are used as the cross validation data and the test data. The maximum training epoch of each network is set to 30 for without the cross validation data set. With the cross validation data set, the maximum training epoch is set to 100. There are 121 examples in the training data set and 100 examples in the test data set, and all these training data will be used to train all the component networks in the ENN.

Totally 3 kinds of NNs are employed to solve this problem. Depending on this

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

function, the number of the input nodes is 2, and the number of the output nodes is 1. The optimal number of the hidden nodes is selected as 15 by the trial and error method. Therefore, the single NN uses 15 hidden nodes in its hidden layer. In the other two ENNs, there are 3 component networks. The numbers of hidden nodes in the component networks are 11, 13 and 15, respectively. Each component network is trained 4 times randomly to find the best weight configuration. The simple averaging ENN and the AIC based ENN will then combine the component networks with the best weight configuration. For the simple averaging ENN, the output of the ENN is combined with the simple averaging method (noted as Ave-ENN), and the AIC based ensemble method (noted as AIC-ENN) uses the modified Δ_{AIC} value to determine the networks' weights. For a fair comparison, the single NN also obtain the best result from the 4 random times for the comparison.

The predicted value of the AIC-ENN for the test data and training data with the cross validation method are shown in Figures 4.3 and 4.4. The statistical performance on the training data set and test data set of 20 runs for these 3 different methods are shown in Table 4.1. From Table 4.1, it can be observed that both ENNs have better accuracy than the single NN. These results demonstrate the better generalization property of the ENN, which has been verified by many others. These results using cross validation data during training are better than those without cross validation data set. Between the Ave-ENN and the AIC-ENN, the latter has much smaller MSE and the SD for the test data, indicating a better generalization capability. Thus, this example demonstrated that the AIC based weighted ENN outperforms both the single NN and the simple averaging ENN.

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

Figure 4.3 Comparison between the actual and predicted peak function test results of AIC-ENN

Figure 4.4 Comparison between the actual and predicted peak function training results of AIC-ENN

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

Table 4.1 Results of twenty runs on peak function with 3 component networks

Without cross validation		Single	Ave-ENN	AIC-ENN
Test-MSE	Minimum	0.187	0.248	0.217
	Mean	0.472	0.327	0.310
	S.D.	0.288	0.066	0.065
Train-MSE	Minimum	0.106	0.108	0.103
	Mean	0.188	0.169	0.168
	S.D.	0.069	0.038	0.044
With cross validation		Single	Ave-ENN	AIC-ENN
Test-MSE	Minimum	0.212	0.205	0.187
	Mean	0.283	0.252	0.246
	S.D.	0.058	0.030	0.034
Train-MSE	Minimum	0.075	0.102	0.074
	Mean	0.167	0.150	0.144
	S.D.	0.049	0.024	0.032

4.4.2 Friedman Function Approximation

Friedman #1 is a nonlinear prediction problem which was used by Friedman (1991) in his work on multivariate adaptive regression splines (MARS). It has 5 independent predictor variables that are uniform in $[0, 1]$. The function used is as follows:

$$Y = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 \quad (4.12)$$

The Friedman #1 with normally distributed noise (mean 0, variance 1) is used to test the AIC based ENN.

The data from the x_1 -axis to the x_5 -axis are selected from the domain $[0, 1]$ randomly for the simulation. 200 training data, 200 cross validation data and 200 test data are generated. The maximum training epoch of each network is set to 30

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

for without the cross validation data set; and set to 100 for with the cross validation data set. The 3 kinds of NNs used for the first example are selected again to solve this problem.

After the same processing as in the peak function, the number of the input nodes is 5, and the number of the output nodes is 1. The single NN uses 10 hidden nodes in its hidden layer. The single NN is trained 5 times randomly to find the best results for comparison. In the other two ENNs, there are 3 component networks. The numbers of hidden nodes in component network are 4, 7, and 10, respectively. After choosing the best weight configuration of each component network from 5 random runs, the simple averaging ensemble method and the AIC based ensemble method combine these three component networks with their respective methods.

The predicted value of the AIC-ENN for the test data and training data are shown in Figures 4.5 and 4.6. Table 4.2 shows the corresponding MSE value of the test data and the training data with 20 random runs with and without the cross validation data during training. From Table 4.2, it can be observed that the AIC-ENN provides the best generalization in terms of the MSE and the SD. The better results can be obtained by using the cross validation data. It can be noted that the mean values of the train-MSE are quite close among the three networks. However, the mean values of the test-MSE for the three networks differ a lot. The relative small SD for both ENNs indicates the main advantage of the ENN, i.e. the consistency of the NN simulation.

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

**Figure 4.5 Comparison between the actual and predicted Friedman #1 function
test results of AIC-ENN**

**Figure 4.6 Comparison between the actual and predicted Friedman #1 function
training results of AIC-ENN**

*Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion***Table 4.2 Results of twenty runs on Friedman #1 function with 3 component networks**

Without cross validation		Single	Ave-ENN	AIC-ENN
Test-MSE	Minimum	1.546	1.899	1.504
	Mean	3.100	2.599	2.373
	S.D.	1.268	0.479	0.506
Train-MSE	Minimum	0.699	0.927	0.839
	Mean	1.359	1.386	1.227
	S.D.	0.440	0.242	0.254
With cross validation		Single	Ave-ENN	AIC-ENN
Test-MSE	Minimum	1.331	1.505	1.498
	Mean	2.597	2.402	2.134
	S.D.	1.157	0.702	0.737
Train-MSE	Minimum	0.709	0.897	0.783
	Mean	1.330	1.437	1.193
	S.D.	0.769	0.500	0.504

4.4.3 The Stress-Strain-Time Relationship Modeling of Mudstone

The strain softening behavior (Figure 4.7) is an important aspect of soil properties. Many material models have been proposed to describe the stress-strain relationship after the peak strength of the soil-like materials. For mudstone (or soft rock), a modified Cam Clay model (Figure 4.8) has been used to estimate the stress-strain-time relationship (Liao, et al., 1998). The nonlinear constitutive law of soft rock is very complex, especially when the strain rate effect is considered. Therefore, it is not convenient to build a mathematical expression of the stress-strain-time relationship. Being a flexible modeling tool, the artificial NN provides an excellent alternative to establish the material model based on the experimental data.

Figure 4.7 Strain softening behavior (Liao, et al., 1998)

Figure 4.8 Modified Cam Clay model (Liao, et al., 1998)

In general, there are four key parameters in the stress-strain-time relationship: strain ϵ , mean effective stress p' , deviator stress q and strain rate $\dot{\epsilon}$. So the proposed ENN model aims to build a functional relationship of $q = f(\epsilon, \dot{\epsilon}, p')$, namely three

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

inputs and one output. In order to compare the performance of the proposed AIC based ENN, the other two NNs (single NN and Ave-ENN) are used for simulations also.

Triaxial experiments were conducted on diatomaceous mudstones (Liao, et al., 1998). The mudstone samples have the following properties: specific gravity 2.183 g/cm^3 , natural water contents 119.6%, liquid limit 172.7%, plastic limit 94.7 and compressive index 1.458. The cylindrical specimens have a diameter 5 cm and height of 10 cm. The specimens were completely saturated. The consolidated and undrained triaxial tests were carried out under strain rate from 0.0044%/min to 1.75%/min, and the consolidation pressure is 2.5MPa. In total, there are 204 sets of experimental data. The detailed experimental data refers to Appendix II. One third of the experimental data (i.e. 68) are chosen as the training data, another one third data (i.e. 68) are selected as the cross validation data, and the remaining one third (i.e. 68) is used for NN test. The maximum training epoch of each network is set to 30 for without the cross validation data set and 100 for with the cross validation data set.

After a series of trial and error, it is found that the optimal hidden nodes number for the single NN is 11. In the other two ENNs, there are 4 component networks. The numbers of hidden nodes in each component network are 5, 7, 9 and 11, respectively. Each component network is trained 4 times randomly to find the best weight configuration. The single NN uses the best result from the 4 random runs.

The MSE of the training data set and test data set of the 3 methods for this problem are shown in Table 4.3. From Table 4.3, it can be found that the results of both the Ave-ENN and the AIC-ENN are more accurate than that of the single NN. And both the single NN and the ENNs with the cross validation data during training have

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

the better performance than those without the cross validation data set. From the Test-MSE, it can be seen that the AIC based ENN with the cross validation data during the training obtained the best performance of all.

Table 4.3 Results of twenty runs on mudstone deviator stress (kPa) modeling with 4 component networks

Without cross validation		Single	Ave-ENN	AIC-ENN
Test-MSE	Minimum	1555.7	1777.9	1880.2
	Mean	3377.5	2410.3	2356.0
	S.D.	1765.9	508.1	411.8
Train-MSE	Minimum	176.6	309.5	398.7
	Mean	344.0	451.2	581.0
	S.D.	96.5	81.5	123.5
With cross validation		Single	Ave-ENN	AIC-ENN
Test-MSE	Minimum	1609.1	1650.9	1607.4
	Mean	2811.2	2183.4	2160.9
	S.D.	892.9	267.6	296.6
Train-MSE	Minimum	291.6	585.1	428.4
	Mean	785.3	850.3	766.1
	S.D.	257.3	171.1	183.3

Figures 4.9 and 4.10 show the test and training predicted value with the cross validation data of the AIC-ENN. Again, similar conclusions to the two analytical functions can be observed: the ENNs have better accuracy than the single NN, and the best results are obtained by the AIC-ENN.

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

Figure 4.9 Comparison between the measured and predicted mudstone modeling test results of AIC-ENN

Figure 4.10 Comparison between the measured and predicted mudstone modeling training results of AIC-ENN

4.4.4 The Prediction of Peak Particle Velocity Damage Criterion for Rock Mass

When an explosive detonates in a blast hole, instantaneously huge amount of energy in forms of pressure and temperature liberates. A proportion of this total energy is utilized for actual breakage and displacement of rock mass and the rest of the energy is spent in undesirable side effects like ground vibration, air blasts, noises, back breaks etc. The ground vibration may induce damage to the surface structure or the surrounding rock. Frequency and PPV are most commonly used parameters for assessment of ground vibrations and also the level of structure damage. However, it is quite difficult to establish a universally acceptable PPV damage criterion for rock mass because it depends on many factors such as the material properties of the intact rock, the joint spacing, joint distribution and orientation. Thus, the different threshold PPV of the safe level vibration was reported in the literature.

Comprehensive study of tunnel damage for sandstone is the US Army's Underground Explosion Tests (UET), as reported by Hendron (1977). The damage is classified into four groups: Intermittent failure, local failure, general failure and tight closure. The tunnel PPV damage criterion for different damage zones was reported. Rupert and Clark (1977) reported that the threshold PPV value is 50mm/s when only minor damage in the form of localized thin spalls and collapse of previous fractured coal ribs. Kidybinski (1986) concluded that damage to underground coal mine openings in the form of small roof falls or floor heave occur when the PPV lies in the range of 50-100mm/s and large roof falls at PPV of 100-200mm/s. Fourie and Green (1993) reported that the PPV of 110mm/s produced only minor damage and extensive damage resulted when PPV was 390mm/s.

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

Adhikari et al. (1994) concluded that the PPV of 153mm/s did not cause any damage to the underground workings for the case of rock mass rating (RMR) of the rock mass is 60. For the case of RMR is 38, the allowable values of PPV were 52mm/s. Persson (1997) reported the PPV damage criterion for Swedish hard rock for five level of damage, i.e., incipient swelling, incipient damage, fragmentation, good fragmentation and crushing. Li and Huang (1994) reported the PPV damage criterion for hard rock and soft rock respectively. The tensile strength and compressive strength of the rock are also closely related to the threshold PPV value. Threshold values of PPV at different RMR of roof rock of seven coal mines were reported by Singh (2002).

The threshold value of PPV is dependent on many factors: the level of damage, the material parameter of the intact rock and the physical properties of the rock mass. And each of the above-mentioned parameters includes many factors. The level damage was divided into three to five levels according to the previous study mentioned above. The material parameter of the intact rock includes the rock type, the static and dynamic tensile and compressive strength. The physical properties of the rock mass include the numbers of discontinuities such as cracks and joints, their properties such as positions, orientations, strength and stiffness etc.

In this section, the threshold value of PPV for the slight damage level is predicted. Because there are numerous numbers of discontinuities in a rock mass and it is impossible to be known exactly, the rock mass classification systems have been used to estimate the rock mass properties. The rock mass rating (RMR) value was used in the present study. RMR classification has six parameters to be considered, namely: (1) uniaxial compressive strength (σ_{ci}) of the intact rock, (2) joint spacing, (3) rock quality designation (RQD), (4) condition of the joints, (5) water

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

flow/pressure and (6) the inclination of the discontinuities. Hence, the input parameters will be defined by three parameters, tensile strength of the intact rock, the compressive strength of the intact rock and RMR of the rock mass. Strictly speaking, only two out of the four parameters are independent parameters because the compressive strength of the intact rock has considered in the RMR system. However, the same RMR of the rock mass do have the very different compressive strength of the intact rock due to the other factors considered in the system, therefore all these three parameters are considered in the present study.

The range of values of different input parameters has been decided by the previous field test published by the various researchers (Hao et al., 2001; Singh, 2002; Singh and Seshagiri, 2005). It is difficult to determine all the input parameters which have influence the threshold PPV. The tensile strength, compressive strength or the RMR of the ground site will be used as the input parameters of the networks, since all these three parameters are not independent and can be estimated according to the certain rock mass. The input parameters for network and their range are listed in Table 4.4.

Table 4.4 Parameters for network and their range

	Parameter	Range
Input parameters	Tensile strength(MPa)	0.6-16.1
	Compressive strength(MPa)	18.0-186.0
	RMR	20-95
Output parameter	PPV(m/s)	0.065-1.0

In this study, total of 47 sets of data are used to predict the threshold PPV. The detailed data refers to Appendix III. One third of the data (i.e. 16) are chosen as the training data, another one third data (i.e. 16) are selected as the cross validation data,

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

and the remaining one third (i.e. 15) is used for NN test. The maximum training epoch of each network is limited to 100. The maximum training epoch of each network is set to 30 for without the cross validation data set. When there is no cross validation data, the training data is increasing to two thirds (16 for training add 16 for cross validation i.e. 32) and the test data is kept same as the data used in the training with the cross validation.

Since there is so limited training data available with 3 inputs and 1 output, it is found that the optimal hidden nodes number for the single NN is 3. As the number of hidden node is small, there is no more space to decrease the hidden nodes to differ the structure of the component networks. In this application, the number of hidden node for the different component networks in the ENNs is chose as the same. Therefore, there are 3 component networks in the other two ENNs. The numbers of hidden nodes in each component network are 3, 3 and 3, respectively. Although the hidden nodes are the same for the component networks, the initial conditions (i.e., the initial weights and biases in the structure of the network) are quite different. Therefore, there is diversity among the component networks. Each component network is trained 4 times randomly to find the best weight configuration. The single NN uses the best result from the 4 random runs.

Figures 4.11 and 4.12 show the test and training predicted value with the cross validation data of the AIC-ENN.

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

Figure 4.11 Comparison between the measured and predicted PPV modeling test results of AIC-ENN

Figure 4.12 Comparison between the measured and predicted PPV modeling training results of AIC-ENN

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

Table 4.5 lists the weights of the component networks and β value of the AIC-ENN in each run. The β value is close to 4, both with and without the cross validation data sets. In the simple averaging ENN (Ave-ENN), the weight of each component network is 0.33. Table 4.5 shows the weights of the component networks in AIC-ENN are quite different, although the number of the hidden nodes in each component network is same. For example, in the first run, the weights of the component networks in AIC-ENN with the cross validation data set are 0.105, 0.210, and 0.685, respectively. That means the performance of the component networks are different, the results of third component network is the best, and there are about 68.5% performance of the AIC-ENN from the third component network in the test.

Table 4.5 Weights of the component networks and β value in AIC-ENN in each run on PPV (m/s) modeling

run	with cross validation				without cross validation			
	hidden nodes of component network			β	hidden nodes of component network			β
	3	3	3		3	3	3	
1	0.105	0.210	0.685	5.51	0.538	0.109	0.353	3.95
2	0.088	0.568	0.344	5.47	0.280	0.620	0.101	5.15
3	0.658	0.190	0.152	3.34	0.642	0.186	0.172	2.73
4	0.284	0.122	0.594	3.85	0.098	0.466	0.436	3.73
5	0.617	0.264	0.119	4.18	0.297	0.587	0.116	4.05
6	0.186	0.557	0.258	2.00	0.243	0.568	0.189	2.00
7	0.179	0.663	0.158	3.19	0.372	0.520	0.108	3.81
8	0.610	0.183	0.207	2.33	0.380	0.107	0.513	3.78
9	0.257	0.581	0.161	2.60	0.628	0.249	0.123	4.11
10	0.149	0.294	0.557	2.74	0.086	0.437	0.477	4.53
11	0.585	0.180	0.235	2.25	0.337	0.545	0.118	3.62
12	0.112	0.320	0.568	4.07	0.164	0.547	0.290	2.34
13	0.387	0.100	0.513	4.15	0.161	0.667	0.173	3.14
14	0.201	0.207	0.592	1.94	0.710	0.178	0.112	5.32
15	0.172	0.634	0.195	2.70	0.118	0.308	0.574	3.86
16	0.100	0.336	0.564	4.67	0.099	0.402	0.499	4.02
17	0.284	0.115	0.601	4.23	0.065	0.107	0.827	11.60
18	0.100	0.775	0.125	6.72	0.105	0.524	0.371	3.98
19	0.136	0.635	0.229	3.68	0.157	0.677	0.166	3.32
20	0.624	0.123	0.254	4.09	0.375	0.105	0.520	3.93
average	0.292	0.353	0.356	3.69	0.293	0.395	0.312	4.15
min	0.088	0.100	0.119	1.94	0.065	0.105	0.101	2.00
max	0.658	0.775	0.685	6.72	0.710	0.677	0.827	11.60

Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion

Twenty runs are carried out for each type of NN, and their statistic results are summarized in Table 4.6. It can be observed that both ENNs have better accuracy than the single NN. The single NN is quite sensitive to the initial weights assigned for each run, which explains the large standard deviation for the single NN. Again, similar conclusions to the two analytical functions can be observed: the ENNs have better accuracy than the single NN, and the best results are obtained by the AIC-ENN, where there is limited data can be used.

Table 4.6 Results of twenty runs on PPV (m/s) modeling with 3 component networks

Without cross validation		Single	Ave-ENN	AIC-ENN
Test-MSE ($\times 10^{-3}$)	Minimum	0.57	0.75	1.77
	Mean	6.70	3.20	2.87
	S.D.	8.28	2.91	0.77
Train-MSE ($\times 10^{-3}$)	Minimum	0.02	0.02	0.01
	Mean	1.47	0.49	0.27
	S.D.	1.90	0.81	0.63
With cross validation		Single	Ave-ENN	AIC-ENN
Test-MSE ($\times 10^{-3}$)	Minimum	0.28	0.16	0.11
	Mean	5.50	2.42	1.99
	S.D.	7.19	2.96	2.99
Train-MSE ($\times 10^{-3}$)	Minimum	0.06	0.11	0.03
	Mean	2.31	1.29	0.65
	S.D.	2.79	1.21	1.12

Table 4.7 shows the predicted threshold PPV for granite rock mass. The tensile strength and compressive strength of the intact rock are 16.1MPa and 186MPa respectively. The range of the RMR lies in 40-95.

*Chapter 4 Design of Ensemble Neural Network with Akaike Information Criterion***Table 4.7 Predicted threshold PPV (m/s) for granite rock mass**

RMR of granite rock mass	95	90	85	80	75	70	65	60	55	50	45	40
Predicted threshold PPV	0.92	0.89	0.85	0.79	0.70	0.68	0.65	0.60	0.55	0.54	0.53	0.50

4.5 CONCLUSIONS

As an information criterion, the AIC is used to balance the model complexity with model accuracy. This chapter proposes a new ensemble NN based on the AIC. Using the AIC to combine these best component networks enables us to balance the ensemble network's accuracy and penalizing the model's complexity, and to create a simple, accurate and stable ENN.

The two theoretical functions and two civil engineering problems with various input dimensions are used to verify the performance of the proposed AIC based ENN. Compared the training without cross validation to the training with the cross validation, the results with the cross validation are better than those without the cross validation. From these results, it also can be found that the proposed AIC based ENN outperforms other methods. These results also showed the range of the proposed ENN applications goes beyond this. It has the potential to be applied to other input-output mapping problems.

CHAPTER 5 DESIGN OF ENSEMBLE NEURAL NETWORK WITH THE ENTROPY

5.1 INTRODUCTION

The information theory based on the concept of entropy has a long history in statistical thermodynamics, quantum physics and communications engineering (Jaynes, 1957; Middleton, 1964). The entropy concept has been defined in various ways (Hartley, 1928; Shannon, 1948; Wiener, 1961; Renyi, 1961) and used to characterize models of communication where signals are mixed with random noise and by using various codes the messages are sent through different information channels. Claude Shannon almost created modern information theory a half-century ago (Shannon, 1948). His work links information to the physical concept of entropy, well-known from thermodynamics and statistical mechanics.

Entropy is one of most elegant concepts in science. Accompanying each progress in the conceptual development of entropy is a big forward step in science. Entropy was first introduced into science as a thermodynamic concept in 1865 for solving the problem of irreversible process. Defining entropy as a measure of the unavailability of a system's thermal energy for conversion into mechanical work, Clausius phrased the second thermodynamic law by claiming that the entropy of an isolated system would never decrease. In 1877, Boltzman gave interpretation of entropy in the framework of statistics. Entropy as a mathematical concept appeared first in Shannon's paper (1948) on information theory. This is a quantum jump, having great impact on modern communication theory. Another important progress for

Chapter 5 Design of Ensemble Neural Network with the Entropy

mathematical entropy was made by Kullback (1959) in 1950s. Entropy is thus an elegant tool widely used by both mathematicians and physicists.

Entropy is yet one of the most difficult concepts in science. Confusions often arise about its definition and applicability due to its abstract trait. This results from the phenomena, known as disorder or uncertainty, described by entropy. In fact, entropy is a measure of disorder. There are some examples of applications of entropy in the specific field of NNs.

Schraudolph (1995) had motivated entropy as an objective for unsupervised learning from the minimum description length and projection pursuit frameworks. Its optimization in an NN is nontrivial since entropy depends on the probability density, which is not explicit in an empirical data sample.

NN architecture optimization is often a critical issue, particularly when VLSI implementation is considered. Pelagotti and Piuri (1997) proposed a new minimization method for multilayered feedforward NNs and an original approach to their synthesis, both based on the analysis of the information quantity (entropy) flowing through the network. A layer is described as an information filter which selects the relevant characteristics until the complete classification is performed. The basic incremental synthesis method, including the supervised training procedure, is derived to design application-tailored neural paradigms with good generalization capability.

Fiori (1999) presented a study of polynomial function-link neural units that learn through an information-theoretic-based criterion. First the structure of the neuron is presented and the unsupervised learning theory is explained and discussed, with particular attention being paid to its probability density function and cumulative distribution function approximation capability. Then an NN formed by such neurons (the polynomial function-link artificial NN) is shown to be able to separate out linearly mixed heterokurtic source signals, i.e. signals endowed with either positive or negative kurtoses.

Chapter 5 Design of Ensemble Neural Network with the Entropy

The prediction of a time series using an NN involves an optimum state-space reconstruction. Chatterjee (2001) reconstructed the state space of the daily 10.7-cm solar radio flux by using an information theory approach. A multi-layer feed-forward neural net is used for short-term prediction of the time series. The convergence of the synaptic weights is obtained partially by simulated annealing and partially by the ‘quick prop’ variation of back-propagation. The result gives a reasonably accurate short-term prediction.

Ng et al. (2003) defined entropy as a term used in the learning phase of an NN. As learning progresses, more hidden nodes get into saturation. The early creation of such hidden nodes may impair generalisation. Hence an entropy approach is proposed to dampen the early creation of such nodes by using a new computation called entropy cycle. Entropy learning also helps to increase the importance of relevant nodes while dampening the less important nodes. At the end of learning, the less important nodes can then be pruned to reduce the memory requirements of the NN.

Yuan et al. (2003) proposed a new method for optimizing the number of hidden neurons based on information entropy. Firstly, an initial NN with enough hidden neurons should be trained by a set of training samples. Secondly, the activation values of hidden neurons should be calculated by inputting the training samples that can be identified correctly by the trained NN. Thirdly, all kinds of partitions should be tried and its information gain should be calculated, and then a decision tree for correctly dividing the whole sample space can be constructed. Finally, the important and related hidden neurons that are included in the tree can be found by searching the whole tree, and other redundant hidden neurons can be deleted. Thus, the number of hidden neurons can be decided.

Chakik et al. (2004) presented a comprehensive maximum entropy procedure for the classification tasks. This maximum entropy is applied successfully to the problem of estimating the probability distribution function of a class with a specific pattern, which is viewed as a probabilistic model handling the classification task.

Chapter 5 Design of Ensemble Neural Network with the Entropy

They proposed an algorithm allowing to construct a non-linear discriminating surface using the maximum entropy procedure.

Santos et al. (2004) presented a new way of performing classification by using the entropy of the error between the output of the MLP and the desired targets, as the function to minimize. The minimization of the entropy of the error implies a constant value for the errors. This, in general, does not imply that the value of the error is zero. In regression, this problem is solved by making a shift of the final result such that its average equals the average value of the desired target. They proved that, under mild conditions, this algorithm, when used in a classification problem, makes the error converge to zero and can thus be used in classification.

Estevez et al. (2005) presented a cross-entropy approach to mapping high-dimensional data into a low-dimensional space embedding. The method allows to project simultaneously the input data and the codebook vectors, obtained with the neural gas quantizer algorithm, into a low-dimensional output space. Neural gas is a kind of self-organizing NNs. The aim of this approach is to preserve the relationship defined by the neural gas neighborhood function for each pair of input and codebook vectors. A cost function based on the cross-entropy between input and output probabilities is minimized by using a Newton-Raphson method.

Liu et al. (2006) presented a maximum-entropy learning algorithm based on radial basis function NNs to control the chaotic system. The algorithm optimizes the object function to derive learning rule of central vectors, and uses the clustering function of network hidden layers. It improves the regression and learning ability of NNs. The numerical experiment of ferroresonance system testifies the effectiveness and feasibility of using the algorithm to control chaos in neutral grounded system.

Wilhelm and Hollunder (2007) presented a new information theoretic approach for network characterizations. It is developed to describe the general type of networks with n nodes and L directed and weighted links, i.e., it also works for the simpler undirected and unweighted networks. The new information theoretic measures for network characterizations are based on a transmitter–receiver analogy of effluxes

Chapter 5 Design of Ensemble Neural Network with the Entropy

and influxes. Based on these measures, they classify networks as either complex or non-complex and as either democracy or dictatorship networks. Directed networks, in particular, are furthermore classified as either information spreading or information collecting networks.

Examples of these applications are the works that try to use entropy to determine and define the complexity of the NN by defining bounds for it or just by generating the NN based on entropy or yet by performing NN architecture optimization and pruning. So, entropy and information theory can also be combined with NNs to solve some real problems.

5.2 ENTROPY

In early 19th century, entropy was introduced in the context of efficiency of heat engines. In information theory, entropy is a very important concept. The entropy of a random variable is defined in terms of its probability distribution and can be shown to be a good measure of randomness or uncertainty. The concept of entropy has been used in several branches of science, but its most significant applications to date have been in the areas of data communication and signal processing.

Shannon and Weaver (1964) gave a measure of uncertainty known as Shannon's entropy, having the same mathematical expression as entropy in statistical mechanics. According to the second law of thermodynamics, entropy never decreases in closed systems. Entropy is a measure of the disorder or complexity of a system. Shannon's insight as to the equivalence of entropy and information spawned a half-century of extensive research into information theory. Let $\mathbf{x}$ be a random variable with sample space $X = \{x_1, x_2, \dots, x_n\}$ and the probability mass associated with the value x_i is p_i i.e., $P(X = x_i) = p_i$, $i=1, 2, \dots, n$. The entropy $H(X) \equiv H_n(p)$ of the ensemble $\{(x_1, p_1), (x_2, p_2), \dots, (x_n, p_n)\}$ as defined by Shannon is given by the expression

$$H(X) \equiv H_n(p) = H_n(p_1, p_2, \dots, p_n) = -c \sum_{i=1}^n p_i \log p_i \quad (5.1)$$

Chapter 5 Design of Ensemble Neural Network with the Entropy

where c is an arbitrary positive constant. The base of the logarithm is arbitrary. If logarithm base is 2, entropy is measured in bits; if the base is e , the entropy is in nats (for natural units). Shannon's entropy is not the only characterization of information contained in a random sample, but Shannon's entropy has found numerous applications (applications of which the following are very important for statistical and economic fields). The measure H , that Shannon used to measure the uncertainty of a collection of events, reaches a maximum when $p_1=p_2=\dots=p_n=1/n$ or, in other words, when the probabilities are uniform.

The maximum entropy formalism published by Jaynes in 1957 is a fundamental concept in information theory. The maximum entropy formalism is used to determine the probabilities underlying a random process from any available statistical data about the process. The resulting maximum entropy probability distribution corresponds to a distribution which is consistent with the given partial information but has maximum uncertainty or entropy associated with it.

For illustrating Jaynes' principle, a discrete random variable X is given. For obtaining the 'most objective' probability distribution of X , the maximum entropy principle has been used in the following procedure:

$$\max \quad S(\lambda) = -\sum_{i=1}^m \lambda_i \ln \lambda_i \quad (5.2)$$

$$\text{subject to} \quad \sum_{i=1}^m \lambda_i = 1 \quad (5.3)$$

$$\text{and the constrains} \quad \sum_{i=1}^m \lambda_i f_j(x_i) = \bar{f}_j, \quad j=1,2,\dots,m \quad (5.4)$$

where $f_j(x)$ is a given function of x . Using the method of Lagrange's multipliers, the resulting distribution is $\lambda_i(x) = e^{-\alpha_0 - \alpha_1 f_1(x_i) - \alpha_2 f_2(x_i) - \dots - \alpha_m f_m(x_i)}$, $i=1,2,\dots,n$, where $\alpha_0, \alpha_1, \dots, \alpha_m$ are Lagrangian multipliers which are determined from $(m+1)$ constraints in Equations (5.3) and (5.4).

It can be proved that Lagrange's method yields global maximum and the distribution so obtained has greater entropy than any other satisfying the given

Chapter 5 Design of Ensemble Neural Network with the Entropy

constrains (Jaynes, 1957). The wide applicability of Jaynes' principle in a number of fields as observed by Tribus (1978) is attributed to: "Jaynes' principle shows that if Shannon's measure is taken to be the measure of uncertainty, not just a measure, the formal results of statistical mechanical reasoning can be carried over to other fields".

5.3 NEWTON'S METHOD

Newton's method is the classic algorithm for finding roots of functions. It is often introduced in the calculus sequence as an application of the derivative of a function. The fundamental idea in Newton's method is to use the tangent line approximation to the function f at the point $(x_0, f(x_0))$. The point-slope formula for the equation of the straight line gives

$$\frac{y - y_0}{x - x_0} = f'(x_0) \quad (5.5)$$

Thus the straight line with equation is as following:

$$y = l_0(x) = f(x_0) + f'(x_0)(x - x_0) \quad (5.6)$$

To find where this crosses the axis, set $y=0$ and solve for x :

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \quad (5.7)$$

Call this new approximate value x_1 , and note that it is much closer to the root α . now continue the process with another straight line to get

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \quad (5.8)$$

Or generally,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (5.9)$$

This is Newton's method. It is based on a very simple idea, one that is fundamental and repeated time and again in the derivation of numerical methods.

Newton's method is not a global method. There are examples for which convergence will be poor or even for which convergence doesn't occur. Usually,

Chapter 5 Design of Ensemble Neural Network with the Entropy

this can be cured by obtaining a better initial guess, but sometimes x_0 have to be taken very close to α in order to obtain convergence.

Ideally, it needs to be stopped when the error, $\alpha - x_n$ is sufficiently small. Of course, it cannot be used since the value of α is not known. However, as long as f' is not zero near the root, this error can be related to a computable quantity. The Mean Value Theorem is shown that

$$f(\alpha) - f(x_n) = f'(c_n)(\alpha - x_n) \quad (5.10)$$

where c_n is some value between α and x_n .

Newton's method also can be used to solve nonlinear equations. Assume there are the nonlinear equations:

$$\begin{cases} u(x, y) = 0 \\ v(x, y) = 0 \end{cases} \quad (5.11)$$

approximation at the point $P_0(x_0, y_0)$, and

$$\begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{vmatrix}_{P_0} \neq 0 \quad (5.12)$$

The equations of iteration are as following:

$$\begin{aligned} x_{n+1} &= x_n + \frac{1}{J_n} \begin{vmatrix} \frac{\partial u}{\partial y} & u \\ \frac{\partial v}{\partial y} & v \end{vmatrix}_{P_n} \\ y_{n+1} &= y_n + \frac{1}{J_n} \begin{vmatrix} u & \frac{\partial u}{\partial x} \\ v & \frac{\partial v}{\partial x} \end{vmatrix}_{P_n} \end{aligned} \quad (5.13)$$

where P_n is the point (x_n, y_n) ; J_n , the Jacobian matrix, is the matrix of partial derivatives of the functions at the point P_n as following:

$$J_n = \begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{vmatrix}_{P_n}$$

5.4 DESIGN ENN USING ENTROPY AND NEWTON'S METHOD

The main idea is to use entropy concept to combine the component networks. To use this concept, three parts of the problem should be optimized at the same time. At first, to maximize the entropy of the combining weights of the whole ENN. Secondly, to minimize the error between the mean output of the ENN and the mean target value. And the last is to minimize the difference of the standard deviation of the output of the ENN and the standard deviation of the target value. This will be benefiting the whole ENN. The proposed entropy based NN should outperform the single NN and simple averaging ENN.

$$\begin{aligned}
 \text{Problem} \quad & \max \quad S(P) = -\sum_{i=1}^m P_i \ln P_i \\
 & \min \quad \begin{aligned} & \mu_{ENNoutput} - \mu_{target} \\ & \sigma_{ENNoutput} - \sigma_{target} \end{aligned} \\
 & \text{subject to} \quad \sum_{i=1}^m P_i = 1, \quad P_i > 0
 \end{aligned} \tag{5.14}$$

where $S(P)$ is the entropy value of the combining weights of the whole ENN;

P_i is the i th component network's weight of the ENN;

m is the number of the component networks;

$\mu_{target} = \frac{1}{n} \sum_{j=1}^n T_j$ is the mean value of the target, where n is the number of sets of

input data to determine the component weights; T_j is the target value with the j th input data set;

$\mu_{ENNoutput} = \frac{1}{n} \sum_{j=1}^n \sum_{i=1}^m P_i f_{ij}(x)$ is the mean value of the output of the ENN, where

$f_{ij}(x)$ is the output of the i th component NNs using the j th input data set;

$\sum_{i=1}^m P_i f_{ij}(x)$ is the output of the ENN using the j th input data set;

Chapter 5 Design of Ensemble Neural Network with the Entropy

$\sigma_{target} = \sqrt{\frac{1}{n} \sum_{j=1}^n (T_j - \mu_{target})^2}$ is the standard deviation of the target;

$\sigma_{ENNOutput} = \sum_{i=1}^m P_i \cdot \sigma_i$ is the standard deviation of the ENN,

where $\sigma_i = \sqrt{\frac{1}{n} \sum_{j=1}^n (f_{ij} - \mu_i)^2}$ is the standard deviation of the output of the i th

component network; $\mu_i = \frac{1}{n} \sum_{j=1}^n f_{ij}(x)$ is the mean value of the output of the i th component network.

In the opinion of statistics, when $X_1, X_2, \dots, X_m$ are dependent random variables,

$\sigma_{ENNOutput}^2 = \sum_{i=1}^m P_i^2 \cdot \sigma_i^2 + 2 \cdot \sum_{i \neq j} P_i \cdot P_j \cdot \text{cov}(X_i, X_j)$. Here, the approximation of

$\sigma_{ENNOutput}$ is used, the steps are as following:

$$\begin{aligned} \sigma_{ENNOutput}^2 &= \sum_{i=1}^m P_i^2 \cdot \sigma_i^2 + 2 \cdot \sum_{i \neq j} P_i \cdot P_j \cdot \text{cov}(X_i, X_j) \\ &\leq \sum_{i=1}^m P_i^2 \cdot \sigma_i^2 + 2 \cdot \sum_{i \neq j} P_i \cdot P_j \cdot \sigma_i \cdot \sigma_j = \left(\sum_{i=1}^m P_i \cdot \sigma_i \right)^2 \end{aligned}$$

Generally, the input data sets for the component networks of the ENN are identical or similar, so the outputs of these component networks are not independent. Since the relationship of the standard deviation of the ENN to the standard deviation of the component networks in the ENN cannot be exactly identified, the upper boundary of the standard deviation of the ENN will be defined as the approximate

standard deviation of the ENN in this approach, which is $\sigma_{ENNOutput} = \sum_{i=1}^m P_i \cdot \sigma_i$.

After using Lagrange multiplier as shown in Equation (5.15), the Lagrangian multipliers need to be solved.

$$\text{Max } L(x, P) = -\sum_{i=1}^m P_i \ln P_i - \lambda_0 \left(\sum_{i=1}^m P_i - 1 \right) - \lambda_1 (\mu_{ENNOutput} - \mu_{target}) - \lambda_2 (\sigma_{ENNOutput} - \sigma_{target}) \quad (5.15)$$

where λ_0 , λ_1 and λ_2 are Lagrangian multipliers.

Chapter 5 Design of Ensemble Neural Network with the Entropy

$$\text{Let } \frac{\partial L(x, P)}{\partial P_i} = 0, i=1, 2, \dots, m$$

$$\text{Since } \frac{\partial L(x, P)}{\partial P_i} = -1 - \ln P_i - \lambda_0 - \lambda_1 \cdot \mu_i - \lambda_2 \cdot \sigma_i,$$

$$\ln P_i = -1 - \lambda_0 - \lambda_1 \cdot \mu_i - \lambda_2 \cdot \sigma_i$$

$$\text{the solution of this problem is } P_i = e^{-1-\lambda_0-\lambda_1 \cdot \mu_i - \lambda_2 \cdot \sigma_i}$$

P_i will be used as the component networks' weight to combine them.

$$\text{Let } A = e^{1+\lambda_0}, B = e^{-\lambda_1}, C = e^{-\lambda_2}, \text{ then } P_i = \frac{B^{\mu_i} \cdot C^{\sigma_i}}{A}.$$

$$\text{Since, } \sum_{i=1}^m P_i = 1, A = \sum_{i=1}^m B^{\mu_i} \cdot C^{\sigma_i} \text{ is a normalization factor that converts the relative}$$

$$\text{probabilities into absolute probabilities, then } P_i = \frac{B^{\mu_i} \cdot C^{\sigma_i}}{\sum_{i=1}^m B^{\mu_i} \cdot C^{\sigma_i}}$$

$$\text{Since, } \mu_t = \mu_{ENN} \Rightarrow \mu_t = \sum_{i=1}^m P_i \mu_i$$

$$\sum_{i=1}^m (\mu_t - \mu_i) \cdot B^{\mu_i} \cdot C^{\sigma_i} = 0$$

$$\text{Since, } \sigma_t = \sigma_{ENN} \Rightarrow \sigma_t = \sum_{i=1}^m P_i \sigma_i$$

$$\sum_{i=1}^m (\sigma_t - \sigma_i) \cdot B^{\mu_i} \cdot C^{\sigma_i} = 0$$

Using Newton's method, let

$$\begin{cases} u(B, C) = \sum_{i=1}^m (\mu_t - \mu_i) \cdot B^{\mu_i} \cdot C^{\sigma_i} \\ v(B, C) = \sum_{i=1}^m (\sigma_t - \sigma_i) \cdot B^{\mu_i} \cdot C^{\sigma_i} \end{cases} \quad (5.16)$$

The equations of iteration are as follows:

$$\begin{aligned}
B_{n+1} &= B_n + \frac{1}{J_n} \begin{vmatrix} \frac{\partial u}{\partial C} & u \\ \frac{\partial v}{\partial C} & v \end{vmatrix}_{P_n} \\
C_{n+1} &= C_n + \frac{1}{J_n} \begin{vmatrix} u & \frac{\partial u}{\partial B} \\ v & \frac{\partial v}{\partial B} \end{vmatrix}_{P_n}
\end{aligned} \tag{5.17}$$

where P_n is the point (B_n, C_n) , the original values of B and C (i.e. B_0 and C_0) can influence the performance of the ENN. When B_0 and C_0 are increasing, the difference of the weights of the component networks in the ENN is increasing and the contribution of the best component network is rising. So, the error of the entropy based ENN with big values of B_0 and C_0 becomes smaller than the case with small B_0 and C_0 . And J_n , the Jacobian matrix, is the matrix of partial derivatives of the functions at the point P_n :

$$J_n = \begin{vmatrix} \frac{\partial u}{\partial B} & \frac{\partial u}{\partial C} \\ \frac{\partial v}{\partial B} & \frac{\partial v}{\partial C} \end{vmatrix}_{P_n}$$

where

$$\frac{\partial u}{\partial B} = \sum_{i=1}^m \mu_i \cdot (\mu_t - \mu_i) \cdot B^{\mu_i-1} \cdot C^{\sigma_i}$$

$$\frac{\partial v}{\partial B} = \sum_{i=1}^m \mu_i \cdot (\sigma_t - \sigma_i) \cdot B^{\mu_i-1} \cdot C^{\sigma_i}$$

$$\frac{\partial u}{\partial C} = \sum_{i=1}^m \sigma_i \cdot (\mu_t - \mu_i) \cdot B^{\mu_i} \cdot C^{\sigma_i-1}$$

$$\frac{\partial v}{\partial C} = \sum_{i=1}^m \sigma_i \cdot (\sigma_t - \sigma_i) \cdot B^{\mu_i} \cdot C^{\sigma_i-1}$$

5.5 ALGORITHM

The major steps of entropy based ENN are illustrated as follows.

1. The data set is split into two parts: the training data set and the test data set. The training data set is used for learning of the various component NN models.

Chapter 5 Design of Ensemble Neural Network with the Entropy

The test data set is not used in the network training, but used for testing the performance of the trained ENN model.

2. Four component networks are used in the ENN. Each component network has one input layer, one hidden layer and one output layer. The number of the input/output nodes is selected based on the problem's input/output attributes. The number of the hidden nodes usually needs to be optimized to get the best ENN.
3. Run each component network several times randomly. For each run, calculate the MSE of the component network by using the training data set. Then compare with each other to select the best weight configuration of the individual component network that has the smallest training MSE.
4. Use Newton's method to find the solution of entropy equations.
5. Construct the ENN with each best component network. Use the modified entropy value of each component network to determine the weight of each component network in the ENN.
6. The MSE of the test data set will be used as the performance measurement of the ENN.

5.6 COMPUTATIONAL EXPERIMENTS

To verify the performance of the ENN proposed in this chapter, three computational experiments are carried out by an ENN program written in MATLAB. Two theoretical functions — the peak function and Friedman function are applied first, and followed by one practical example — the modelling of PPV damage criterion for rock mass. For comparison purpose, a simple averaging ENN which has the same entropy based ENN structure and a single NN which uses the best hidden nodes number are also simulated using the same data.

5.6.1 Peak Function Approximation

The peak function, which is shown in Figure 4.3, is a sample function of two variables and obtained by translating and scaling Gaussian distributions. It is a typical complex two-dimensional function used as demonstration in MATLAB in Equation (4.11).

The peak function with normally distributed noise (mean 0, variance 0.05) is used to generate the training data and the test data. First, 11×11 evenly distributed data along both the x-axis and the y-axis are selected from the domain $[-3, 3]$ as the training data for the simulation. Another one 10×10 evenly distributed points from the same domain are used as the test data. The maximum training epoch of each network is set to 30. There are 121 examples in the training data set and 100 test examples, and all these training data will be used to train all the component networks in the ENN.

Totally 3 kinds of NNs are employed to solve this problem. Depending on this function, the number of the input nodes is 2, and the number of the output nodes is 1. The optimal number of the hidden nodes is selected as 15 by the trial and error method. Therefore, the single NN uses 15 hidden nodes in its hidden layer. In the other two ENNs, there are 4 component networks. The numbers of hidden nodes in the component networks are 9, 11, 13 and 15, respectively. Each component network is trained 4 times randomly to find the best weight configuration. The simple averaging ENN and the entropy based ENN will then combine the component networks with the best weight configuration. For the simple averaging ENN, the output of the ensemble networks is combined with the simple averaging method (noted as Ave-ENN), and the entropy based ensemble method (noted as EN-ENN) uses the modified entropy value to determine the networks' weights. For a fair comparison, the single NN also obtain the best result from the 4 random times for the comparison.

Chapter 5 Design of Ensemble Neural Network with the Entropy

The performance of the EN-ENN for the test data and training data are shown in Figures 5.1 and 5.2. And all data points are within a narrow band of the 45° line. The statistical performance on the training data set and test data set of 20 runs for EN-ENN with the different original values of B and C are shown in Table 5.1. From Table 5.1, the error of the entropy based ENN is decreasing when the original values of B and C increasing.

Figure 5.1 Comparison between the actual and predicted peak function test results of EN-ENN

Figure 5.2 Comparison between the actual and predicted peak function training results of EN-ENN

Table 5.1 Results of twenty runs in EN-ENN with different B_0 and C_0 in peak function

EN-ENN	$B_0=C_0=$	3	10	100	1000
Test-MSE	Minimum	0.247	0.241	0.232	0.227
	Mean	0.338	0.334	0.329	0.326
	S.D.	0.055	0.055	0.055	0.057
Train-MSE	Minimum	0.132	0.127	0.121	0.116
	Mean	0.175	0.171	0.166	0.161
	S.D.	0.028	0.029	0.029	0.031

The statistical performance on the training data set and test data set of 20 runs for these 3 different methods are shown in Table 5.2. Single 9 in the table is used to denote the single NN with 9 hidden nodes. From Table 5.2, it can be observed that both ENNs have better accuracy than the single NN. For the single component networks, the network with higher number of the hidden nodes has the better performance. When these 4 component networks combined, the performance of

Chapter 5 Design of Ensemble Neural Network with the Entropy

ENNs becomes better than that of the single one. Those results demonstrate the better generalization property of the ENN, which has been verified by many others. Between the Ave-ENN and the EN-ENN, the latter has much smaller MSE and the SD for the test data, indicating a better generalization capability. Thus, this example demonstrated that the entropy based weighted ENN outperforms both the single NN and the simple averaging ENN.

Table 5.2 Results of twenty runs on peak function with 4 component networks

		Single 9	Single 11	Single 13	Single 15	Ave- ENN	EN- ENN
Test-MSE	Minimum	0.486	0.289	0.324	0.234	0.269	0.247
	Mean	0.818	0.669	0.495	0.460	0.360	0.338
	S.D.	0.236	0.294	0.126	0.201	0.056	0.055
Train-MSE	Minimum	0.418	0.217	0.171	0.100	0.157	0.132
	Mean	0.557	0.357	0.271	0.177	0.196	0.175
	S.D.	0.088	0.060	0.066	0.044	0.027	0.028

The weights of the 4 component networks using the entropy based ENN are listed in Table 5.3. From Table 5.3, it can be observed that the contribution of the component network is depending on network performance, which is related to the number of hidden node, and the original values of B and C in the Newton's method (expressed by B_0 and C_0) affect the weight value of each component network. Theoretically, B_0 and C_0 are picked randomly in the proceeding of the method, and they do not have specific affect on the final results. In the current case study, increasing the original values of B and C, the difference of the weights between the component networks is increasing. This induces the results to become better with the higher value of the B_0 and C_0 .

In other words, the component network in the best performance gives more contribution to the ENN with the higher values of B_0 and C_0 . At the same time, the worst network will decrease the contribution to the ENN when the values of B_0 and C_0 increase. For example, the best performance network in this entropy based ENN is the network with 15 hidden nodes (the mean value of the test MSE in single

Chapter 5 Design of Ensemble Neural Network with the Entropy

network with 15 hidden nodes is 0.460 in Table 5.2.), the worst performance of the network is the network with 9 hidden nodes (the mean value of the test MSE in single network with 9 hidden nodes is 0.818 in Table 5.2.). When the values of B_0 and C_0 are 3, the average weight of the component network with 15 hidden nodes is 0.311, at the same time, the average weight of the component network with 9 hidden nodes is 0.178. When B_0 and C_0 increases to 1000, the average weight of the best component network is up to 0.385, and the average weight of the worst component network is reduced to 0.116.

Since the contribution of the best component network in the entropy based ENN increases from 0.311 to 0.385, and the effect of the worst component network decreases from 0.178 to 0.116, the performance of the entropy based ENN is improved. From Table 5.1, it can be found that the mean value of test MSE is reduced from 0.338 to 0.326 when B_0 and C_0 are increasing from 3 to 1000.

Chapter 5 Design of Ensemble Neural Network with the Entropy

Table 5.3 Weights of the component networks in EN-ENN in peak function

run	B₀=C₀=3 hidden nodes of the component networks				B₀=C₀=10 hidden nodes of the component networks			
	9	11	13	15	9	11	13	15
1	0.104	0.245	0.294	0.357	0.097	0.243	0.296	0.364
2	0.161	0.306	0.237	0.296	0.141	0.319	0.228	0.312
3	0.144	0.208	0.320	0.328	0.126	0.200	0.333	0.342
4	0.186	0.213	0.267	0.334	0.173	0.198	0.270	0.359
5	0.214	0.241	0.260	0.285	0.200	0.242	0.262	0.296
6	0.177	0.248	0.272	0.303	0.159	0.249	0.284	0.309
7	0.179	0.243	0.260	0.318	0.167	0.242	0.260	0.331
8	0.193	0.255	0.256	0.296	0.165	0.258	0.261	0.316
9	0.223	0.253	0.263	0.261	0.221	0.254	0.262	0.263
10	0.210	0.230	0.261	0.299	0.196	0.229	0.261	0.314
11	0.182	0.235	0.280	0.303	0.169	0.231	0.286	0.314
12	0.202	0.220	0.285	0.293	0.186	0.205	0.289	0.320
13	0.184	0.233	0.248	0.335	0.171	0.230	0.248	0.352
14	0.157	0.250	0.298	0.295	0.128	0.248	0.319	0.305
15	0.182	0.222	0.262	0.334	0.158	0.207	0.262	0.373
16	0.176	0.215	0.319	0.290	0.163	0.208	0.333	0.296
17	0.185	0.219	0.263	0.333	0.173	0.208	0.263	0.356
18	0.153	0.256	0.249	0.342	0.136	0.251	0.249	0.364
19	0.208	0.196	0.289	0.306	0.201	0.183	0.298	0.319
20	0.130	0.252	0.309	0.309	0.114	0.248	0.313	0.325
average	0.178	0.237	0.275	0.311	0.162	0.233	0.279	0.326
min	0.104	0.196	0.237	0.261	0.097	0.183	0.228	0.263
max	0.223	0.306	0.320	0.357	0.221	0.319	0.333	0.373
run	B₀=C₀=100 hidden nodes of the component networks				B₀=C₀=1000 hidden nodes of the component networks			
	9	11	13	15	9	11	13	15
1	0.082	0.238	0.300	0.381	0.067	0.231	0.303	0.398
2	0.107	0.342	0.211	0.339	0.081	0.362	0.193	0.365
3	0.096	0.184	0.356	0.365	0.073	0.167	0.376	0.385
4	0.149	0.172	0.272	0.408	0.126	0.147	0.270	0.458
5	0.176	0.244	0.264	0.316	0.155	0.245	0.264	0.336
6	0.129	0.249	0.306	0.317	0.103	0.246	0.328	0.323
7	0.147	0.238	0.259	0.356	0.128	0.234	0.257	0.382
8	0.121	0.261	0.266	0.353	0.086	0.259	0.266	0.389
9	0.223	0.256	0.256	0.265	0.262	0.257	0.231	0.250
10	0.170	0.227	0.259	0.344	0.147	0.224	0.256	0.373
11	0.145	0.224	0.297	0.334	0.125	0.215	0.306	0.354
12	0.157	0.176	0.293	0.374	0.130	0.149	0.291	0.430
13	0.146	0.223	0.248	0.383	0.125	0.214	0.246	0.415
14	0.085	0.240	0.357	0.319	0.056	0.227	0.392	0.326
15	0.119	0.177	0.256	0.448	0.088	0.148	0.243	0.520
16	0.140	0.195	0.358	0.307	0.119	0.182	0.383	0.316
17	0.176	0.200	0.260	0.364	0.118	0.157	0.250	0.475
18	0.109	0.237	0.247	0.407	0.086	0.223	0.241	0.451
19	0.185	0.160	0.313	0.342	0.170	0.138	0.326	0.365
20	0.088	0.238	0.319	0.355	0.067	0.227	0.323	0.384
average	0.137	0.224	0.285	0.354	0.116	0.213	0.287	0.385
min	0.082	0.160	0.211	0.265	0.056	0.138	0.193	0.250
max	0.223	0.342	0.358	0.448	0.262	0.362	0.392	0.520

5.6.2 Friedman Function Approximation

Friedman #1 is a nonlinear prediction problem which was used by Friedman (1991) in his work on multivariate adaptive regression splines (MARS). It has 5 independent predictor variables that are uniform in $[0, 1]$. The function used is as Equation (4.12). The Friedman #1 with normally distributed noise (mean 0, variance 1) is used to test the entropy based ENN.

First, $5 \times 5 \times 5 \times 5 \times 5$ evenly distributed data along both the x-axis and the y-axis are selected from the domain $[0, 1]$ as the training data for the simulation. Another $4 \times 4 \times 4 \times 4 \times 4$ evenly distributed points from the same domain are used as the test data. The maximum training epoch of each network is set to 30. There are 3125 examples in the training data set, and 1024 examples in the test data set. The 3 kinds of NNs used for the first example are selected again to solve this problem.

After the same processing as in the peak function, the number of the input nodes is 5, and the number of the output nodes is 1. The single NN uses 10 hidden nodes in its hidden layer. The single NN is trained 4 times randomly to find the best results for comparison. In the other two ENNs, there are 4 component networks. The numbers of hidden nodes in component network are 4, 6, 8 and 10, respectively. After choosing the best weight configuration of each component network from 4 random runs, the simple averaging ensemble method and the entropy based ensemble method combine these four component networks with their respective methods.

Table 5.4 shows the corresponding MSE value of the test data and the training data with 20 runs during training. From Table 5.4, it can be observed that the EN-ENN provides the best generalization in terms of the MSE and the SD. The relative small SD for both ENNs indicates the main advantage of the ENN, i.e. the consistency of the NN simulation. The comparison between the actual and predicted test results

Chapter 5 Design of Ensemble Neural Network with the Entropy

and training results of EN-ENN are shown in Figures 5.3 and 5.4. All these thousands of the data points are within a narrow band of the 45° line.

Table 5.4 Results of twenty runs on Friedman #1 function with 4 component networks

		Single 4	Single 6	Single 8	Single 10	Ave- ENN	EN- ENN
Test-MSE	Minimum	2.941	1.075	1.540	1.052	1.410	1.285
	Mean	5.693	3.419	4.567	2.889	1.948	1.862
	S.D.	2.064	2.076	2.403	1.679	0.455	0.437
Train-MSE	Minimum	2.382	0.226	0.188	0.041	0.481	0.181
	Mean	5.155	2.398	1.183	0.357	0.963	0.584
	S.D.	1.381	1.648	0.648	0.383	0.358	0.296

Figure 5.3 Comparison between the actual and predicted Friedman #1 function test results of EN-ENN

Figure 5.4 Comparison between the actual and predicted Friedman #1 function training results of EN-ENN

5.6.3 The Prediction of Peak Particle Velocity Damage Criterion for Rock Mass

The prediction of the PPV damage criterion for rock mass is applied to prove the performance of the ENN again in this chapter. The details of the PPV model can be found in Section 4.4.4.

In this section, total of 47 groups data are used to predict the threshold PPV. Two thirds of the data (i.e. 32) were chosen as the training data, and the remaining one third (i.e. 15) is used for NN test. There is no cross validation. The maximum training epoch of each network is set to 30. Since there is limited training data available with 3 inputs and 1 output, it is found that the optimal hidden nodes number for the single NN is 3. In this application, the hidden node numbers of the different component networks in the ENNs are chosen as same, i.e. 3. Therefore,

Chapter 5 Design of Ensemble Neural Network with the Entropy

there are 4 component networks in the other two ENNs. The numbers of hidden nodes in each component network are 3, 3, 3 and 3, respectively. Each component network is trained 4 times randomly to find the best weight configuration. The single NN uses the best result from the 4 random runs.

Twenty runs are carried out for EN-ENN with each different original values of B and C, and their statistic results are summarized in Table 5.5. From Table 5.5, the MSE of the entropy based ENN is decreasing when the original values of B and C increasing.

Table 5.5 Results of twenty runs in EN-ENN with different B0 and C0 in PPV (m/s) modelling

EN-ENN	B ₀ =C ₀ =	3	10	100	1000
	Minimum	0.927	0.927	0.928	0.929
Test-MSE	Mean	2.262	2.256	2.247	2.237
($\times 10^{-3}$)	S.D.	1.147	1.141	1.129	1.118
	Minimum	0.036	0.036	0.036	0.036
Train-MSE	Mean	0.390	0.388	0.382	0.377
($\times 10^{-3}$)	S.D.	0.357	0.355	0.351	0.347

Figures 5.5 and 5.6 show the performance of the entropy based ENN in the test and training data sets. Table 5.6 lists the weights of the component networks of the EN-ENN in each run. In the simple averaging ENN (Ave-ENN), the weight of each component network is 0.25. Table 5.5 shows the weights of the component networks in EN-ENN are a little different with the same number of the hidden nodes in each component network. From the average weights values, it can be observed that the first component network is performed best, and the second one is performed worst. When the original values of the B and C increasing, the contribution of the first network to the ENN is increasing and the contribution of the second network to the ENN is decreasing.

Chapter 5 Design of Ensemble Neural Network with the Entropy

Figure 5.5 Comparison between the measured and predicted PPV modelling test results of EN-ENN

Figure 5.6 Comparison between the measured and predicted PPV modelling training results of EN-ENN

Chapter 5 Design of Ensemble Neural Network with the Entropy

Table 5.6 Weights of the component networks in EN-ENN in PPV modelling

run	B₀=C₀=3 hidden nodes of the component networks				B₀=C₀=10 hidden nodes of the component networks			
	3	3	3	3	3	3	3	3
1	0.256	0.245	0.256	0.243	0.257	0.244	0.257	0.242
2	0.260	0.238	0.246	0.256	0.262	0.236	0.245	0.257
3	0.254	0.239	0.254	0.252	0.254	0.239	0.255	0.252
4	0.256	0.256	0.238	0.250	0.258	0.258	0.235	0.248
5	0.257	0.250	0.257	0.236	0.258	0.250	0.259	0.233
6	0.260	0.250	0.250	0.239	0.262	0.251	0.250	0.237
7	0.252	0.249	0.249	0.249	0.253	0.249	0.249	0.249
8	0.254	0.243	0.252	0.252	0.254	0.241	0.252	0.252
9	0.260	0.226	0.258	0.256	0.261	0.222	0.260	0.257
10	0.250	0.251	0.248	0.252	0.250	0.251	0.247	0.252
11	0.253	0.250	0.247	0.249	0.253	0.250	0.248	0.249
12	0.249	0.249	0.251	0.250	0.248	0.250	0.252	0.250
13	0.256	0.238	0.258	0.248	0.258	0.234	0.261	0.248
14	0.251	0.251	0.251	0.247	0.251	0.252	0.251	0.246
15	0.249	0.255	0.249	0.247	0.250	0.256	0.249	0.245
16	0.262	0.262	0.223	0.254	0.264	0.264	0.217	0.255
17	0.255	0.241	0.252	0.253	0.255	0.241	0.252	0.253
18	0.251	0.249	0.250	0.250	0.251	0.249	0.250	0.250
19	0.251	0.249	0.248	0.252	0.252	0.249	0.248	0.252
20	0.253	0.245	0.252	0.249	0.254	0.245	0.253	0.249
average	0.254	0.247	0.250	0.249	0.255	0.246	0.249	0.249
min	0.249	0.226	0.223	0.236	0.248	0.222	0.217	0.233
max	0.262	0.262	0.258	0.256	0.264	0.264	0.261	0.257
run	B₀=C₀=100 hidden nodes of the component networks				B₀=C₀=1000 hidden nodes of the component networks			
	3	3	3	3	3	3	3	3
1	0.259	0.242	0.258	0.241	0.261	0.240	0.260	0.240
2	0.266	0.231	0.243	0.260	0.269	0.227	0.242	0.262
3	0.255	0.237	0.256	0.253	0.256	0.235	0.256	0.253
4	0.262	0.262	0.230	0.246	0.266	0.266	0.226	0.243
5	0.261	0.251	0.262	0.227	0.264	0.251	0.265	0.221
6	0.266	0.251	0.250	0.233	0.269	0.251	0.250	0.229
7	0.254	0.249	0.248	0.249	0.256	0.249	0.248	0.248
8	0.256	0.238	0.253	0.253	0.258	0.235	0.254	0.254
9	0.264	0.215	0.262	0.259	0.267	0.208	0.265	0.261
10	0.250	0.251	0.247	0.252	0.250	0.251	0.247	0.253
11	0.254	0.249	0.249	0.248	0.255	0.249	0.250	0.247
12	0.247	0.250	0.252	0.250	0.246	0.251	0.253	0.251
13	0.261	0.226	0.266	0.248	0.264	0.219	0.271	0.247
14	0.251	0.253	0.252	0.245	0.252	0.254	0.252	0.243
15	0.251	0.260	0.247	0.242	0.253	0.263	0.246	0.239
16	0.269	0.269	0.205	0.257	0.274	0.274	0.193	0.259
17	0.256	0.240	0.251	0.253	0.256	0.240	0.251	0.253
18	0.251	0.249	0.250	0.250	0.251	0.249	0.251	0.250
19	0.252	0.248	0.247	0.252	0.253	0.248	0.247	0.253
20	0.256	0.244	0.254	0.247	0.257	0.242	0.255	0.246
average	0.257	0.246	0.249	0.248	0.259	0.245	0.249	0.247
min	0.247	0.215	0.205	0.227	0.246	0.208	0.193	0.221
max	0.269	0.269	0.266	0.260	0.274	0.274	0.271	0.262

Chapter 5 Design of Ensemble Neural Network with the Entropy

Twenty runs are carried out for each type of NN, and their statistic results are summarized in Table 5.7. It can be observed that both ENNs have better accuracy than the single NN. The single NN is quite sensitive to the initial weights assigned for each run, which explains the large standard deviation for the single NN. From Table 5.7, it can be found that the single component networks performance differently even with the same number of the hidden nodes. This is because the performance of the network depends on some other factors, e.g. initial weights in the network. Again, similar conclusions to the two analytical functions can be observed: the ENNs have better accuracy than the single NN, and the best results are obtained by the entropy based ENN, where there is so limited data can be used.

Table 5.7 Results of twenty runs on PPV (m/s) with 4 component networks

		Single 3	Single 3	Single 3	Single 3	Ave- ENN	EN- ENN
Test-MSE ($\times 10^{-3}$)	Minimum	0.57	0.35	0.15	0.59	0.92	0.93
	Mean	2.88	4.56	3.35	7.47	2.28	2.26
	S.D.	2.46	4.60	2.42	12.03	1.17	1.15
Train-MSE ($\times 10^{-3}$)	Minimum	0.01	0.02	0.01	0.02	0.04	0.04
	Mean	0.24	1.62	0.69	1.10	0.41	0.39
	S.D.	0.42	1.54	1.12	1.49	0.37	0.36

5.7 CONCLUSIONS

This chapter presents an entropy based ENN which improves the performance of the ENN. Three computational experiments with various input dimensions, which are peak function, Friedman function and PPV modelling, are used to verify the performance of the proposed ENN. From these results, it can be found that the proposed entropy based ENN outperforms other methods. It is hoped that the entropy based ENN could be an alternative tool for other input-output mapping problems.

CHAPTER 6 DESIGN OF ENSEMBLE NEURAL NETWORK USING THE AIC AND ENTROPY

6.1 INTRODUCTION

Wolpert (1996) proposed “no free lunch theorem” which stated that if one inducer is better than another in some domains, then there are necessarily other domains in which this relationship is reversed. The “no free lunch theorem” implies that for a given problem, a certain approach can yield more information from the same data than other approaches.

The dilemma of what method to choose becomes even greater, if other factors such as comprehensibility are taken into consideration. For instance, for a specific domain, an entropy based ENN may outperform an AIC based ENN in accuracy. However, from the comprehensibility aspect, AIC based ENN is considered better. In other words, even if the researcher knows that ENN is more accurate, he still has a dilemma what method to use. For example, the entropy based ENN focuses on more accuracy, but the complexity of the network structure will be ignored by the ENN, so the weight distribution of the component networks in the ENN is proportional to the complexity of the networks. The more complex the network, the more time needs it to be trained. But the AIC based ENN pays more attention to the complexity of the networks; it penalizes the complexity of the network. Usually, it will give less weight to the best performance component network than to others, since the best component network is commonly the most complex one. So if there is a method to combine advantages of both methods, it will be a better choice to use.

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

Since both of the AIC based ENN and the entropy based ENN focus on the optimal combining weight distribution of the whole ENN, to adopt both AIC and entropy to improve the optimization of the contribution of each component NN is feasible. Entropy is a powerful tool to find the good performance component NN from the random system, and AIC can be used to limit the complexity of the component network in the ENN. So to combine AIC and entropy to improve the ENN performance can create a promising, simple, accurate and stable ENN. The studies to combine two or three different methods to improve the performance of the NN started from the beginning of the 20 century.

Ren and Zhao (2002) proposed a new algorithm which combines the AIC with the golden-section optimization technique to find the optimal architecture for single-hidden layer feedforward NNs. They have verified that the modified AIC criterion is in close agreement with the network generalization. It is observed that as long as proximity to global minimum solution is found for each configuration of the network, the AIC function of the networks over the whole domain is unimodal. The golden-section search method is very effective and computationally time-saving, especially for large size or complex problems. So the AIC and the golden-section optimization technique are combined to the NN. And the proposed optimization algorithm is applied to the modelling of the concrete strength under triaxial stresses.

Jiang et al. (2003) proposed a structural modular neural network, by combining the back-propagation (BP) neurons and the radial basis function (RBF) neurons at the hidden layer to construct a better input-output mapping both locally and globally. The use of the GA in searching the best hidden neurons makes the structural modular neural network less likely to be trapped in local minima than the traditional gradient-based search algorithms. They verified the modular approach is more accurate than using the RBFNN or the BPNN alone.

With the development of electric market reform, short-term load forecasting (STLF) has been paid more and more attention. Sun et al. (2005) presented a hybrid model to integrate information entropy and data mining theory with NN to establish a new short-term load forecasting model. First, information entropy theory is used to

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

select relevant ones from all influential factors; the results are used as inputs of NN. Secondly, according to the features of power load, the typical historical load data samples were selected as the training set which have the same weather characteristic as the certain forecasting day by using data mining theory. Finally, Elman NN forecasting model is constructed combining the reduced factors and typical training set. The presented model can effectively improve forecasting accuracy.

Li et al. (2008) proposed a damage identification method based on the combination of ANN, Dempster–Shafer (D–S) evidence theory-based information fusion and the Shannon entropy, to form a weighted and selective information fusion technique to reduce the impact of uncertainties on damage identification. Several individual ANNs with different inputs are created first. Optimal weighting coefficients of the ANNs are obtained by GA. The decision obtained from the ANN with the largest weighting coefficient is the most reliable. Damage identification based D–S evidence theory is carried out by combining the decision of the ANN with the largest weighting coefficient with that of the ANN with the second largest weighting coefficient. Next, Shannon entropies of the decisions of the ANN with the largest weighting coefficient and that obtained by the information fusion are calculated to measure the uncertainty level of the decisions, respectively. The decision with smallest entropy remains in the next information fusion operation because this decision has less uncertainty. The decision with smaller entropy will be combined with the decision of the ANN with the third largest weighting coefficient. The operation is repeated until the last ANN with the minimum-weighting coefficient is fused.

Examples of these applications are the works that try to improve NN performance by using two or more methods combined to the networks, or one method to combine two different structure networks. So combining AIC and entropy to the ENN will be a feasible way to improve the performance of the proposed ENN.

6.2 DESIGN ENN USING AIC-ENTROPY AND NEWTON'S METHOD

Maximum entropy principle (Jaynes, 1957) can be regarded as a powerful quantitative tool for inference in probabilistic systems when only partial information is given. And AIC analysis is able to pick up the most suitable one from a group of candidate models. AIC is an effective method that can measure a network with low error but penalize networks with a large number of free parameters. For the proposed ENN, to combine the entropy and AIC is a better choice if it is feasible.

The creative idea is to use entropy concept to combine the modified delta_AIC to the ENN. To use this concept, this problem should optimize four parts at the same time. First part is to maximize the entropy of the combining weights of the whole ENN. Second part is to maximize the modified Delta_AIC of the whole ENN. The first part and the second part are calculated as the entropy value of the ENN in this problem. To minimize the error between the mean output of the ENN and the mean target value is third part. And the last part is to minimize the difference between the standard deviation of the output of the ENN and the standard deviation of the target value. After considering all these four parts together, the proposed AIC and entropy based ENN should outperform the single NN, the simple averaging ENN, the AIC based ENN and the entropy based ENN.

$$\begin{aligned}
 \text{Problem} \quad & \max \quad S(P) = -\sum_{i=1}^m P_i \cdot \ln P_i - \alpha \cdot \sum_{i=1}^m P_i \cdot (\Delta_{AIC})_i \\
 & \min \quad \begin{aligned} & \mu_{ENNoutput} - \mu_{target} \\ & \sigma_{ENNoutput} - \sigma_{target} \end{aligned} \\
 & \text{subject to} \quad \sum_{i=1}^m P_i = 1, \quad P_i > 0
 \end{aligned} \tag{6.1}$$

where $S(P)$ is the entropy value of the whole ENN;

P_i is the i th component network's weight of the ENN; α is the constant; and m is the number of the component networks,

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

$\mu_{target} = \frac{1}{n} \sum_{j=1}^n T_j$ is the mean value of the target; $\mu_{ENNoutput} = \frac{1}{n} \sum_{j=1}^n \sum_{i=1}^m P_i f_{ij}(x)$ is the mean value of the output of the ENN; $\sigma_{target} = \sqrt{\frac{1}{n} \sum_{j=1}^n (T_j - \mu_{target})^2}$ is the standard deviation of the target; and $\sigma_{ENNoutput} = \sum_{i=1}^m P_i \cdot \sigma_i$ is the standard deviation of the ENN output.

In the opinion of statistics, when $X_1, X_2, \dots, X_m$ are dependent random variables,

$$\sigma_{ENNoutput}^2 = \sum_{i=1}^m P_i^2 \cdot \sigma_i^2 + 2 \cdot \sum_{i \neq j} P_i \cdot P_j \cdot \text{cov}(X_i, X_j) . \quad \text{Here, the approximation of } \sigma_{ENNoutput} \text{ is used as } \sum_{i=1}^m P_i \cdot \sigma_i , \text{ the detailed steps can be found in Section 5.4.}$$

In AIC, the Delta_AIC (Δ_i), is a measure of each model relative to the best model, and can be calculated in Equation (4.3). To consider the contributions of all the component networks to the ENN, a modified delta_AIC is proposed as follows:

$$(\Delta_{AIC})_i = \theta + \frac{\Delta_i}{\Delta_{\max}} \cdot \beta \quad (6.2)$$

where $\Delta_{\max}$ is the delta_AIC value of the worst component network; Δ_i is the modified delta_AIC value of i th component network; θ and β are constants. The value of θ is used to adjust the different degree of the modified delta_AIC. The value of β determines the range of the modified delta_AIC. A small β represents a more uniform Δ_m distribution, whereas a large β value corresponds to a more diverse Δ_m distribution. As β is a measure of diversity, the Equation (4.8) is proposed to determine the value of β .

After using Lagrange multiplier as shown in Equation (6.3), the Lagrangian multipliers need to be solved.

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

Max

$$L(x, P) = -\sum_{i=1}^m P_i \ln P_i - \alpha \cdot \sum_{i=1}^m P_i \cdot (\Delta_{AIC})_i - \lambda_0 \left(\sum_{i=1}^m P_i - 1 \right) - \lambda_1 (\mu_{ENNOutput} - \mu_{target}) - \lambda_2 (\sigma_{ENNOutput} - \sigma_{target}) \quad (6.3)$$

where λ_0 , λ_1 and λ_2 are Lagrangian multipliers.

$$\text{Let } \frac{\partial L(x, P)}{\partial P_i} = 0, i=1, 2, \dots, m$$

$$\text{Since } \frac{\partial L(x, P)}{\partial P_i} = -1 - \ln P_i - \alpha \cdot (\Delta_{AIC})_i - \lambda_0 - \lambda_1 \cdot \mu_i - \lambda_2 \cdot \sigma_i \quad (6.4)$$

$$\ln P_i = -1 - \lambda_0 - \alpha \cdot (\Delta_{AIC})_i - \lambda_1 \cdot \mu_i - \lambda_2 \cdot \sigma_i$$

the solution of this problem is

$$P_i = e^{-1 - \lambda_0 - \alpha \cdot (\Delta_{AIC})_i - \lambda_1 \cdot \mu_i - \lambda_2 \cdot \sigma_i}$$

P_i will be used as the component networks' weight to combine them.

$$\text{Let } A = e^{1 + \lambda_0}, B = e^{-\lambda_1}, C = e^{-\lambda_2}, D = e^{-\alpha}, \text{ then } P_i = \frac{B^{\mu_i} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i}}{A}$$

Since, $\sum_{i=1}^m P_i = 1$, $A = \sum_{i=1}^m B^{\mu_i} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i}$ is a normalization factor that converts the

relative probabilities into absolute probabilities, then $P_i = \frac{B^{\mu_i} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i}}{\sum_{i=1}^m B^{\mu_i} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i}}$

$$\text{Since, } \mu_t = \mu_{ENN} \Rightarrow \mu_t = \sum_{i=1}^m P_i \mu_i$$

$$\sum_{i=1}^m (\mu_t - \mu_i) \cdot B^{\mu_i} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i} = 0$$

$$\text{Since, } \sigma_t = \sigma_{ENN} \Rightarrow \sigma_t = \sum_{i=1}^m P_i \sigma_i$$

$$\sum_{i=1}^m (\sigma_t - \sigma_i) \cdot B^{\mu_i} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i} = 0$$

Using Newton's method, let

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

$$\begin{cases} u(B, C) = \sum_{i=1}^m (\mu_t - \mu_i) \cdot B^{\mu_i} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i} \\ v(B, C) = \sum_{i=1}^m (\sigma_t - \sigma_i) \cdot B^{\mu_i} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i} \end{cases} \quad (6.5)$$

The equations of iteration are same as follows:

$$\begin{aligned} B_{n+1} &= B_n + \frac{1}{J_n} \begin{vmatrix} \frac{\partial u}{\partial C} & u \\ \frac{\partial v}{\partial C} & v \end{vmatrix}_{P_n} \\ C_{n+1} &= C_n + \frac{1}{J_n} \begin{vmatrix} u & \frac{\partial u}{\partial B} \\ v & \frac{\partial v}{\partial B} \end{vmatrix}_{P_n} \end{aligned} \quad (6.6)$$

where P_n is the point (B_n, C_n) ; and J_n , the Jacobian matrix, is the matrix of partial derivatives of the functions at the point P_n :

$$J_n = \begin{vmatrix} \frac{\partial u}{\partial B} & \frac{\partial u}{\partial C} \\ \frac{\partial v}{\partial B} & \frac{\partial v}{\partial C} \end{vmatrix}_{P_n}$$

where

$$\frac{\partial u}{\partial B} = \sum_{i=1}^m \mu_i \cdot (\mu_t - \mu_i) \cdot B^{\mu_i-1} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i}$$

$$\frac{\partial v}{\partial B} = \sum_{i=1}^m \mu_i \cdot (\sigma_t - \sigma_i) \cdot B^{\mu_i-1} \cdot C^{\sigma_i} \cdot D^{(\Delta_{AIC})_i}$$

$$\frac{\partial u}{\partial C} = \sum_{i=1}^m \sigma_i \cdot (\mu_t - \mu_i) \cdot B^{\mu_i} \cdot C^{\sigma_i-1} \cdot D^{(\Delta_{AIC})_i}$$

$$\frac{\partial v}{\partial C} = \sum_{i=1}^m \sigma_i \cdot (\sigma_t - \sigma_i) \cdot B^{\mu_i} \cdot C^{\sigma_i-1} \cdot D^{(\Delta_{AIC})_i}$$

6.3 ALGORITHM

The major steps of the AIC and entropy based ENN can be illustrated as follows.

1. The data set is split into two parts: the training data and the test data set. The training data set is used for learning of the various component network models.

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

The test data set is not used in the network training, but used for testing the performance of the trained ENN model.

2. Four component networks are used in the ENN. Each component network has one input layer, one hidden layer and one output layer. The number of the input/output nodes is selected according to the problem's input/output attributes. The number of the hidden nodes usually needs to be optimized to get the best ENN. Each component network will be trained by the same training data set.
3. Run each component network several times randomly. For each run, calculate the MSE of the component network by using the training data set. Then compare with each other to select the best weight configuration of the individual component network that has the smallest training MSE.
4. According to the diversity of the component networks, to determine the modified ΔAIC value. Add these modified ΔAIC value to the entropy problem. Use Newton's method to find the solution of entropy equations.
5. Construct the ENN with each best component network. Use the modified entropy value of each component network to determine the weight of each component network in the ENN.
6. The MSE of the test data set will be used as the performance measurement of the ENN.

6.4 COMPUTATIONAL EXPERIMENTS

To verify the performance of the ENN proposed in this chapter, two computational experiments are carried out by an ENN program written in MATLAB. Two theoretical functions — the peak function and Friedman function are applied. For comparison purpose, a simple averaging ENN which has the same AIC-entropy

based ENN structure; the AIC based ENN; the entropy based ENN and a single NN which uses the best hidden nodes number are also simulated using the same data.

6.4.1 Peak Function Approximation

The peak function, which is shown in Figure 4.3, is a sample function of two variables and obtained by translating and scaling Gaussian distributions. It is a typical complex two-dimensional function used as demonstration in MATLAB in Equation (4.11).

The peak function with normally distributed noise (mean 0, variance 0.05) is used to generate the training data and the test data. First, 11×11 evenly distributed data along both the x-axis and the y-axis are selected from the domain $[-3, 3]$ as the training data for the simulation. Another 10×10 evenly distributed points from the same domain are used as the test data. The maximum training epoch of each network is set to 30.

Totally 5 kinds of NNs are employed to solve this problem. Depending on this function, the number of the input nodes is 2, and the number of the output nodes is 1. The optimal number of the hidden nodes is selected as 15 by the trial and error method. Therefore, the single NN uses 15 hidden nodes in its hidden layer. In the other four ENNs, there are 4 component networks. The numbers of hidden nodes in the component networks are 9, 11, 13 and 15, respectively. Each component network is trained 4 times randomly to find the best weight configuration. The four ENNs combine the component networks with the best weight configuration. For the simple averaging ENN, the output of the ensemble networks is combined with the simple averaging method (noted as Ave-ENN), the entropy based ensemble method (noted as EN-ENN) uses the modified entropy value to determine the networks' weights, the AIC based ensemble method (noted as AIC-ENN) used the modified ΔAIC value to determine the weights of the component networks, and the AIC-entropy based ENN (noted as AIC-EN-ENN) is combine the AIC and entropy to

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

determine the weights. For a fair comparison, the single NN also obtain the best result from the 4 random times for the comparison.

Figures 6.1 and 6.2 show the comparison between the actual and predicted test results and training results of AIC-EN-ENN. All the data points are within a narrow band of the 45° line.

Figure 6.1 Comparison between the actual and predicted peak function test results of AIC-EN-ENN

Figure 6.2 Comparison between the actual and predicted peak function training results of AIC-EN-ENN

The weights of the 4 component networks using the AIC-entropy based ENN are listed in Table 6.1. From Table 6.1, it can be observed, the contribution of the component network is depending on network performance, which is related to the number of hidden nodes. And the original values of B and C affect the weight value of each component network. Increase the original values of B and C, the difference of the weights between the component networks is increasing. In other words, the component network in the best performance, whose number of hidden node is 15, gives more contribution to the ENN with the higher values of B and C. At the same time, the worst network with 9 hidden nodes decrease the contribution to the ENN when the values of B and C increase. Comparing to the Table 5.3, in EN-ENN, there are 2 component networks with average weights that are below 0.25, but in AIC-EN-ENN, there is only 1 component network with the average weights below 0.25 except for the original values of B and C is 1000. For example, when the values of B_0 and C_0 are equal to 3, the average weights of component networks are 0.178, 0.237, 0.275 and 0.311 in Table 5.3, the average weights are 0.170, 0.268,

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

0.281 and 0.281 correspondingly in Table 6.1; when the values of B_0 and C_0 are equal to 100, the average weights of component networks are 0.137, 0.224, 0.285 and 0.354 in Table 5.3, the average weights are 0.131, 0.253, 0.292 and 0.324 correspondingly in Table 6.1. That means for the entropy based ENN, the performance of 2 component networks are below the average performance (i.e. 0.25), but for AIC-EN-ENN, only 1 component network performance is lower than average level. So, the AIC-EN-ENN can give the more reasonable weights to the component networks, and it focuses more on the contribution of the better performance of the component network.

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

Table 6.1 Weights of the component networks in AIC-EN-ENN in peak function

run	B₀=C₀=3 hidden nodes of the component networks				B₀=C₀=10 hidden nodes of the component networks			
	9	11	13	15	9	11	13	15
1	0.088	0.331	0.233	0.347	0.082	0.329	0.235	0.355
2	0.139	0.473	0.179	0.209	0.120	0.490	0.171	0.219
3	0.086	0.162	0.396	0.356	0.074	0.154	0.407	0.366
4	0.202	0.281	0.222	0.294	0.190	0.265	0.226	0.319
5	0.244	0.271	0.263	0.222	0.229	0.274	0.265	0.231
6	0.115	0.245	0.190	0.452	0.102	0.244	0.197	0.457
7	0.247	0.303	0.248	0.203	0.233	0.304	0.250	0.213
8	0.126	0.354	0.287	0.233	0.107	0.356	0.290	0.247
9	0.116	0.285	0.363	0.237	0.111	0.286	0.364	0.240
10	0.270	0.184	0.283	0.263	0.253	0.185	0.284	0.278
11	0.212	0.249	0.290	0.249	0.197	0.246	0.298	0.259
12	0.194	0.172	0.401	0.234	0.178	0.160	0.406	0.256
13	0.247	0.250	0.225	0.279	0.230	0.248	0.227	0.295
14	0.143	0.316	0.248	0.293	0.117	0.314	0.266	0.304
15	0.108	0.299	0.277	0.316	0.092	0.277	0.277	0.354
16	0.173	0.190	0.416	0.221	0.160	0.184	0.432	0.225
17	0.190	0.232	0.173	0.405	0.182	0.219	0.170	0.429
18	0.143	0.304	0.178	0.375	0.128	0.296	0.177	0.399
19	0.285	0.170	0.324	0.222	0.275	0.159	0.334	0.232
20	0.078	0.285	0.432	0.206	0.072	0.280	0.426	0.222
average	0.170	0.268	0.281	0.281	0.157	0.263	0.285	0.295
min	0.078	0.162	0.173	0.203	0.072	0.154	0.170	0.213
max	0.285	0.473	0.432	0.452	0.275	0.490	0.432	0.457
run	B₀=C₀=100 hidden nodes of the component networks				B₀=C₀=1000 hidden nodes of the component networks			
	9	11	13	15	9	11	13	15
1	0.068	0.321	0.239	0.372	0.056	0.313	0.241	0.390
2	0.090	0.518	0.156	0.235	0.068	0.542	0.141	0.250
3	0.055	0.138	0.425	0.382	0.041	0.123	0.440	0.395
4	0.166	0.233	0.232	0.369	0.143	0.203	0.234	0.420
5	0.203	0.279	0.269	0.249	0.180	0.282	0.272	0.267
6	0.082	0.242	0.211	0.465	0.065	0.239	0.225	0.471
7	0.208	0.305	0.254	0.234	0.184	0.304	0.256	0.255
8	0.077	0.356	0.293	0.275	0.055	0.352	0.292	0.302
9	0.101	0.287	0.366	0.246	0.093	0.288	0.367	0.252
10	0.223	0.186	0.286	0.306	0.195	0.185	0.285	0.335
11	0.171	0.240	0.311	0.278	0.148	0.233	0.323	0.296
12	0.151	0.138	0.412	0.300	0.126	0.117	0.411	0.346
13	0.200	0.244	0.230	0.327	0.173	0.238	0.231	0.358
14	0.078	0.305	0.299	0.318	0.052	0.291	0.330	0.328
15	0.069	0.236	0.270	0.426	0.050	0.197	0.257	0.497
16	0.136	0.171	0.462	0.231	0.115	0.158	0.490	0.236
17	0.126	0.175	0.164	0.535	0.093	0.143	0.154	0.610
18	0.102	0.279	0.175	0.444	0.080	0.260	0.170	0.490
19	0.256	0.140	0.354	0.251	0.237	0.122	0.372	0.270
20	0.057	0.269	0.430	0.244	0.043	0.257	0.435	0.265
average	0.131	0.253	0.292	0.324	0.110	0.242	0.296	0.352
min	0.055	0.138	0.156	0.231	0.041	0.117	0.141	0.236
max	0.256	0.518	0.462	0.535	0.237	0.542	0.490	0.610

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

The statistical performance on the training data set and test data set of 20 runs for AIC-EN-ENN with the different original values of B and C are shown in Table 6.2. From Table 6.2, the error of the AIC-EN-ENN is decreasing when the original value of B and C increasing. Usually, the value of B and C is problem depending. Comparing to the Table 5.2, it can be found that the performance of the AIC-EN-ENN is better than the performance of the EN-ENN.

Table 6.2 Results of twenty runs in AIC-EN-ENN with different B_0 and C_0 in peak function

AIC-EN-ENN	$B_0=C_0=$	3	10	100	1000
Test-MSE	Minimum	0.240	0.236	0.229	0.225
	Mean	0.336	0.333	0.328	0.324
	S.D.	0.056	0.055	0.054	0.053
Train-MSE	Minimum	0.119	0.116	0.113	0.111
	Mean	0.174	0.171	0.165	0.160
	S.D.	0.035	0.034	0.033	0.032

The statistical performance on the training data set and test data set of 20 runs for these 5 different methods are shown in Table 6.3. From Table 6.3, it can be observed that the ENNs have better accuracy than the single NN. When these 4 component networks are combined, the performance of ENNs becomes better than that of the single one. These results demonstrate the better generalization property of the ENN, and have been verified by many others. Among the Ave-ENN, EN-ENN and AIC-EN-ENN, the latest has the smallest MSE for the test data, indicating a best generalization capability. In this example, the entropy based weighted ENN is better than the AIC based weighted ENN. Thus, this example demonstrated that the AIC-entropy based weighted ENN outperforms all the single NN, simple averaging ENN, AIC based weighted ENN and entropy based weighted ENN.

Table 6.3 Results of twenty runs on peak function with 4 component networks

		Single 15	Ave- ENN	EN- ENN	AIC- ENN	AIC-EN- ENN
Test-MSE	Minimum	0.234	0.269	0.247	0.255	0.240
	Mean	0.460	0.360	0.338	0.356	0.336
	S.D.	0.201	0.056	0.055	0.059	0.056
Train-MSE	Minimum	0.100	0.157	0.132	0.143	0.119
	Mean	0.177	0.196	0.175	0.192	0.174
	S.D.	0.044	0.027	0.028	0.035	0.035

6.4.2 Friedman Function Approximation

Friedman #1 is a nonlinear prediction problem which was used by Friedman (1991) in his work on multivariate adaptive regression splines (MARS). It has 5 independent predictor variables that are uniform in $[0, 1]$. The function used is in Equation (4.12). The Friedman #1 with normally distributed noise (mean 0, variance 1) is used to test the ENN.

The data from the x_1 -axis to the x_5 -axis are selected from the domain $[0, 1]$ randomly for the simulation. 200 training data and 200 test data are generated. The maximum training epoch of each network is set to 30 for without the cross validation data set. All 5 kinds of NNs used for the first example are adopted again to solve this problem.

After the same processing as in the first example, the number of the input nodes is 5, and the number of the output nodes is 1. The single NN uses 10 hidden nodes in its hidden layer. The single NN is trained 4 times randomly to find the best results for comparison. In the other four ENNs, there are 4 component networks. The numbers of hidden nodes in component network are 4, 6, 8 and 10, respectively. After choosing the best weight configuration of each component network from 4 random runs, the four ensemble methods combine these four component networks with their respective methods.

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

Figures 6.3 and 6.4 show the performance of the AIC-EN-ENN in the test and training results. Table 6.4 shows the corresponding MSE value of the test data and the training data of 20 random runs without the cross validation data during training. From Table 6.4, it can be observed that the AIC-EN-ENN provides the best generalization in terms of the MSE. In this case, the AIC based weighted ENN outperforms the entropy weighted ENN. But the AIC-EN-ENN improves the performance of the AIC-ENN. According to these results, it can be found that the ENN using both entropy and AIC is about to get the best results. This is due to the fact that the proposed method combines the advantages of both methods.

Figure 6.3 Comparison between the actual and predicted Friedman #1 function test results of AIC-EN-ENN

Figure 6.4 Comparison between the actual and predicted Friedman #1 function training results of AIC-EN-ENN

Table 6.4 Results of twenty runs on Friedman #1 function with 4 component networks

		Single	Ave- ENN	EN- ENN	AIC- ENN	AIC-EN- ENN
Test-MSE	Minimum	1.379	1.682	1.552	1.389	1.381
	Mean	3.259	2.580	2.451	2.336	2.324
	S.D.	1.079	0.637	0.677	0.706	0.707
Train-MSE	Minimum	0.670	0.871	0.731	0.731	0.696
	Mean	1.418	1.393	1.285	1.288	1.236
	S.D.	0.525	0.339	0.362	0.406	0.388

Table 6.5 shows the weights of the AIC-ENN, EN-ENN and AIC-EN-ENN, respectively. From the Table 6.5, it can be found that the same component network can be combined with different weights to get the different results. For the AIC-ENN, it focuses on balance the error and number of the parameters, so the weights of the component network with 6 hidden nodes get the highest value, its average weight reaches to 0.309. The weights of other three component networks are from

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

0.220 to 0.251. For the entropy based ENN, it focuses more on the error, so the weight of the component network with 10 hidden nodes, which is the best performance of the component network, reaches to 0.307, other weights of the three component network are between 0.197 to 0.256. With the same parameter of the AIC-ENN and EN-ENN, the AIC-EN-ENN is combining the advantage of the AIC-ENN and EN-ENN. In the AIC-EN-ENN, the weights of the component networks with 6 and 10 hidden nodes are the higher; their average weights are 0.273 and 0.285, respectively. The weights of the component networks with 4 and 8 hidden nodes are lower; their average weights are 0.181 and 0.260. The component network with 4 hidden nodes is not very good for this function, so the contribution to the AIC-EN-ENN is controlled to lower than that both in the AIC-ENN and in the entropy-ENN. In the AIC-ENN and entropy-ENN, there are 2 component networks with average weights that are below 0.25, but in AIC-EN-ENN, there is an only 1 component network with the average weights below 0.25. So, the AIC-EN-ENN can give the more reasonable weights to the component networks.

Chapter 6 Design of Ensemble Neural Network Using the AIC and Entropy

Table 6.5 Weights of the component networks in AIC-EN-ENN in Friedman function

run	AIC-ENN hidden nodes of the component networks				EN-ENN hidden nodes of the component networks			
	4	6	8	10	4	6	8	10
1	0.215	0.148	0.460	0.176	0.224	0.175	0.329	0.272
2	0.154	0.532	0.162	0.152	0.200	0.305	0.194	0.301
3	0.103	0.549	0.252	0.096	0.197	0.279	0.276	0.248
4	0.176	0.383	0.299	0.142	0.315	0.198	0.238	0.248
5	0.181	0.171	0.426	0.222	0.168	0.192	0.346	0.294
6	0.246	0.165	0.197	0.392	0.239	0.257	0.264	0.241
7	0.151	0.294	0.374	0.181	0.183	0.279	0.298	0.239
8	0.407	0.171	0.210	0.212	0.192	0.219	0.292	0.297
9	0.228	0.237	0.393	0.142	0.247	0.263	0.238	0.252
10	0.208	0.428	0.166	0.198	0.147	0.088	0.319	0.447
11	0.372	0.181	0.303	0.144	0.217	0.233	0.286	0.265
12	0.294	0.381	0.124	0.201	0.146	0.277	0.275	0.302
13	0.181	0.425	0.186	0.209	0.175	0.297	0.160	0.368
14	0.145	0.161	0.280	0.414	0.194	0.241	0.267	0.299
15	0.102	0.456	0.328	0.114	0.171	0.274	0.282	0.273
16	0.421	0.205	0.170	0.204	0.251	0.213	0.191	0.344
17	0.072	0.440	0.239	0.249	0.184	0.269	0.274	0.274
18	0.163	0.208	0.175	0.454	0.208	0.241	0.204	0.347
19	0.115	0.485	0.171	0.229	0.161	0.289	0.242	0.308
20	0.458	0.166	0.114	0.262	0.121	0.214	0.134	0.531
average	0.220	0.309	0.251	0.220	0.197	0.240	0.256	0.307
min	0.072	0.148	0.114	0.096	0.121	0.088	0.134	0.239
max	0.458	0.549	0.460	0.454	0.315	0.305	0.346	0.531
run	AIC-EN-ENN							
	4	6	8	10				
1	0.219	0.129	0.428	0.224				
2	0.129	0.495	0.145	0.232				
3	0.159	0.419	0.303	0.119				
4	0.275	0.257	0.280	0.189				
5	0.152	0.150	0.400	0.299				
6	0.267	0.219	0.237	0.278				
7	0.150	0.302	0.338	0.211				
8	0.250	0.186	0.279	0.286				
9	0.239	0.255	0.329	0.177				
10	0.167	0.144	0.279	0.410				
11	0.266	0.207	0.327	0.200				
12	0.170	0.353	0.185	0.293				
13	0.144	0.377	0.134	0.345				
14	0.137	0.188	0.293	0.381				
15	0.077	0.399	0.367	0.158				
16	0.324	0.199	0.157	0.319				
17	0.038	0.368	0.294	0.300				
18	0.162	0.222	0.167	0.449				
19	0.076	0.415	0.190	0.320				
20	0.224	0.185	0.078	0.513				
average	0.181	0.273	0.260	0.285				
min	0.038	0.129	0.078	0.119				
max	0.324	0.495	0.428	0.513				

6.5 CONCLUSIONS

Determination of model complexity in an NN is crucial in NN design. The AIC based ENN can balance the error and the number of the free parameters in each component network, and entropy based ENN can improve the performance of the ENN. To combine the advantage of both AIC based ENN and entropy based ENN, this chapter proposes a new ensemble NN which adopt both the AIC and the entropy, and use Newton's method to help find the weights of the component network.

The two computational experiments (peak function and Friedman function) with various input dimensions are used to verify the performance of the proposed ENN. From these results, it can be found that the proposed AIC-entropy based ENN outperforms other methods. Although the proposed method has been used for the theoretical functions, the results also showed the potential of the proposed ENN to be applied to other kinds of civil engineering problems.

CHAPTER 7 CONCLUSIONS AND FUTURE WORK

7.1 SUMMARY

For a good ENN, the main idea is to create the component networks with both accuracy and diversity, then combine these component networks to obtain the good performance.

The present research work emphasizes the following tasks:

- (1) the development of the AIC based ENN to apply to the civil engineering problems;
- (2) the development of the entropy based ENN to apply to the civil engineering problems;
- (3) the improvement of the proposed entropy based ENN by using AIC to consider the complexity of the component networks.

The AIC based ENN has been developed in Chapter 4 to predict the strength of the intact rock. AIC is a performance measure that can be used to help determine the best network size. It measures a network with low error but penalizes networks with a large number of free parameters. So, using AIC to determine the contributions of all the component networks and combining them is a feasible method. AIC concept has been used to investigate the performance of the ENNs. A peak function and Friedman function are used first as the computational experiment to verify the

Chapter 7 Conclusions and Future Work

accuracy of the proposed model. Then, the proposed ENN is applied to model the stress-strain-time relationship of mudstone and predict PPV damage criterion for the rock mass. All examples show that the proposed AIC based ENN method has better performance and is more accurate than other popular ENN method.

The entropy based ENN has been developed in Chapter 5. Entropy is a measure of the disorder or complexity of a system that can be used to help determine the best component networks. It measures a network performance by low error and low difference of the standard deviation between the output and target. Combined by the Lagrange's method and Newton's method, using the entropy to determine the contributions of each component network is a feasible method. A peak function and Friedman function are used first as the computational experiment to verify the accuracy of the proposed model. Then, the proposed ENN is applied to model the PPV damage criterion for the rock mass again. All examples show that the proposed entropy based method has better performance and is more accurate than other popular ENN method.

In Chapter 6, the AIC based ENN and the entropy based ENN are compared. AIC based ENN is able to balance the error and the number of the free parameters in the NN. The entropy based ENN more focuses on the performance of the NN. For peak function, the entropy based ENN is better than AIC based ENN, but for Friedman function, the AIC based ENN is the better one. So a new ensemble method by combining both the AIC and the entropy is proposed. And these two examples show that the proposed AIC-entropy based ENN has better performance and is more accurate than other ENN methods.

All these proposed ENN systems are written in Matlab in this thesis.

7.2 CONCLUSIONS

This Ph.D. thesis aims to develop new algorithms for ENN to avoid the shortcoming of the traditional ENNs. A new AIC based ENN, a new entropy based ENN and a new combined AIC and entropy based ENN have been proposed to improve the network accuracy (i.e. generalization). These methods and the models developed in this thesis may be considered as an alternative tool for practical input-output mapping problems.

The following conclusions can be drawn from the present work:

1. The ENNs can construct a better input-output mapping than the single NN. Compared to the single NN, the ENNs can reduce the risk of overfitting. At the same time, the ENNs can get more accurate and stable results than the single NN. Even with the limited data sets available for the analysis, the bootstrap sampling technique can help ENN to obtain the more accurate results.
2. The AIC is used as an information criterion for measuring performance of each component network in the ENN. The AIC based ENN is proposed and implemented. In this proposed ENN, the AIC is used to balance the model complexity with model accuracy. The AIC is very helpful to save the computation time of the ENN by reducing the model complexity.
3. Compared with the training without cross validation to the training with the cross validation, the results with the cross validation are better than those without the cross validation in both single NN and ENNs. This is because the validation set is very useful to measure the quality of the network during the

training process.

4. Entropy is a very important concept in the information theory. The entropy of a random variable is defined in terms of its probability distribution and can be shown to be a good measure of randomness or uncertainty. The entropy based ENN is proposed and implemented. The proposed entropy based ENN outperforms the single NN and the simple averaging ENN.
5. Compared the AIC based ENN to the entropy based ENN, each proposed ENN has its own advantages. AIC based ENN is able to balance the error and the number of the free parameters in the NN. The entropy based ENN focuses more on the performance of the NN. A new ENN with coupled AIC and entropy is proposed and implemented. It is found that this proposed ENN outperforms the single NN, the simple averaging ENN, the AIC based ENN and the entropy based ENN.
6. Although the proposed methods have so far been used for the theoretical functions and the civil engineering problems, the potential range of applications goes beyond this.

7.3 FUTURE WORK

The principal areas of future work can be divided into the following two main parts:

- (1) improve the model performance with information theory; (2) extend the proposed system to more realistic input-output mapping problems in civil engineering applications.

7.3.1 Improve the Model Performance

The performance of the proposed ENN will be improved. First, AIC will be applied to optimize the number of the component networks and/or the number of the hidden nodes of the component network in the ENN; second, the entropy concept will be applied to optimize the number of the hidden nodes of the component networks in ENN.

7.3.1.1 Optimize the ENNs with AIC

The AIC, a performance measure, can be used to determine the best network size. The AIC measure rewards a network with low MSE but penalizes networks with a large number of weights. For problems with limited data, this measurement can be used to determine the network that is least likely to over-fit the data.

AIC has been used to optimize the ensemble network model by adjusting the component network's weight. The results in Chapter 4 demonstrate that the proposed AIC based ENN is an effective approach whether the network structure size is big or small. In this proposed method, the number of the hidden nodes in each component network and the number of the component networks are given in the examples. Since AIC can find the best compromise between network structure size and training set error, the optimal AIC ensemble network (AICENN) will be obtained automatically with adjusting the K value. Changing the ENN structure will result in varying of K value, which is related to the number of the free parameters in the whole ensemble network.

AICENN will be proposed for deciding when to add hidden nodes in an existing component network and when to halt the construction of component networks. The

Chapter 7 Conclusions and Future Work

major steps of AICENN are shown in Figure 7.1. First, the minimal ENN architecture with two component networks will be created. Second, after training all the component networks, the MSE on the test data will be used to determine the final AICENN. If the test MSE is not acceptable, adding one or few hidden nodes in the component network will be adopted. If AIC does not decrease in the new ensemble networks with adding hidden nodes, one new component network will be added to the ENN. Finally, the optimal AICENN will be created until the AIC does not decrease further.

The main idea of AICENN is to automatically optimize the structure of the ENN, because most of the NNs' algorithms can not create optimal networks' structure automatically. The number of the hidden nodes, the number of the component networks in the ensemble network must be given in the traditional algorithms. Certainly, the detailed algorithm of AICENN still needs to be studied further in the future.

Figure 7.1 Flowchart of AICENN

7.3.1.2 Combine the Selected Networks with the Entropy Method

Entropy has been used to optimize the ensemble network model by adjusting the component network's weight. The results in Chapter 5 demonstrate that the proposed entropy based ENN is an effective approach no matter the network structure size is big or small. In this proposed method, the number of the hidden nodes in each component network and the number of the component networks are given in peak function, Friedman function and PPV modeling. Since entropy can find the best training set error, the optimal entropy ensemble neural network (EENN) will be obtained automatically with deleting the worse networks.

EENN will be proposed for deciding when to delete a component network in an existing ensemble network. The major steps of EENN are shown in Figure 7.2. First, the maximum ENN architecture with more component networks with different number of the hidden nodes will be created. Second, after training all the component networks, the MSE on the training data will be used to determine the final EENN. The limited error is set originally, if the MSE of the network using the training data set is more than the pre-defined error, this network will be deleted from the ENN. Finally, the optimal EENN will be created to be tested.

Figure 7.2 Flowchart of EENN

The major steps of EENN are shown in Figure 7.2, which are illustrated further as follows.

1. The data set is split into two parts: the training data and the test data set. The training data set is used for learning of the various component network models. The test data set is not used in the network training, but used for testing the performance of the trained ENN model.

2. Maximum original component networks (e.g. 10 networks) are used in the ENN. Each component network has one input layer, one hidden layer and one output layer. The number of the input/output nodes is selected according to the problem's input/output attributes. The number of the hidden nodes usually needs to be optimized to get the best ENN. Each component network will be trained by the same training data set.
3. To create the optimal component networks. Run each component network to calculate the MSE of the training data set. If the MSE of the component network is bigger than the pre-defined error, this network is discarded. For each run, calculate the MSE of the component network by using the training data set. Then compare with each other to select the best weight configuration of the individual component network that has the smallest training MSE.
4. Construct the ENN with these optimal component networks. Use the modified entropy value of each component network to determine the weight of each component network in the ENN. And using Newton's method to find the value of solution of entropy equation.
5. The MSE of the test data set will be used as the performance measurement of the ENN.

The main idea is to use entropy concept to combine the entropy ensemble neural network (EENN), i.e., to both minimize the error between the mean output of the ENN and the mean target value.

Chapter 7 Conclusions and Future Work

$$\begin{aligned}
 \text{Problem} \quad & \max \quad S(P) = -\sum_{i=1}^m P_i \ln P_i \\
 & \min \quad \mu_{ENN\text{Output}} - \mu_{\text{target}} \\
 & \text{subject to} \quad \sum_{i=1}^m P_i = 1, \quad P_i > 0
 \end{aligned} \tag{7.1}$$

where P_i is the i th component network's weight of the ENN;

m is the number of the component networks;

$\mu_{\text{target}} = \frac{1}{n} \sum_{j=1}^n T_j$ is the mean value of the target, where n is the number of sets of

input data to determine the component weights; T_j is the target value with the j th input data set;

$\mu_{ENN\text{Output}} = \frac{1}{n} \sum_{j=1}^n \sum_{i=1}^m P_i f_{ij}(x)$ is the mean value of the output of the ENN, where

$f_{ij}(x)$ is the output of the i th component NNs using the j th input data set;

$\sum_{i=1}^m P_i f_{ij}(x)$ is the output of the ENN using the j th input data set.

After using Lagrange multiplier as shown in Equation (7.2), the Lagrangian multipliers need to be solved.

$$\begin{aligned}
 \text{Max} \quad L(x, P) = & -\sum_{i=1}^m P_i \ln P_i - \lambda_0 \left(\sum_{i=1}^m P_i - 1 \right) - \lambda_1 (\mu_{ENN\text{Output}} - \mu_{\text{target}}) \\
 & (7.2)
 \end{aligned}$$

where λ_0 , λ_1 and λ_2 are Lagrangian multipliers.

$$\text{Let } \frac{\partial L(x, P)}{\partial P_i} = 0, \quad i=1, 2, \dots, m$$

Chapter 7 Conclusions and Future Work

Since $\frac{\partial L(x, P)}{\partial P_i} = -1 - \ln P_i - \lambda_0 - \lambda_1 \cdot \mu_i$

$$\ln P_i = -1 - \lambda_0 - \lambda_1 \cdot \mu_i$$

the solution of this problem is

$$P_i = e^{-1-\lambda_0-\lambda_1 \cdot \mu_i}$$

P_i will be used as the component networks' weight to combine them.

Let $A = e^{1+\lambda_0}$, $B = e^{-\lambda_1}$, then $P_i = \frac{B^{\mu_i}}{A}$

Since, $\sum_{i=1}^m P_i = 1$, $A = \sum_{i=1}^m B^{\mu_i}$ is a normalization factor that converts the relative

probabilities into absolute probabilities, then $P_i = \frac{B^{\mu_i}}{\sum_{i=1}^m B^{\mu_i}}$

Since, $\mu_t = \mu_{ENN} \Rightarrow \mu_t = \sum_{i=1}^m P_i \mu_i$

the above equation can be written as following:

$$\mu_t = \mu_{ENN} \Rightarrow \mu_t = \sum_{i=1}^m P_i \mu_i \Rightarrow \mu_t = \sum_{i=1}^m \frac{B^{\mu_i} \cdot \mu_i}{\sum_{i=1}^m B^{\mu_i}} \Rightarrow \mu_t \cdot \sum_{i=1}^m B^{\mu_i} = \sum_{i=1}^m (B^{\mu_i} \cdot \mu_i)$$

$$f(B) = \sum_{i=1}^m B^{\mu_i} \cdot (\mu_t - \mu_i)$$

$$f'(B) = \sum_{i=1}^m B^{(\mu_i-1)} \cdot \mu_i \cdot (\mu_t - \mu_i)$$

Using Newton's method to solve the value of B:

$$B_{n+1} = B_n - \frac{f(B_n)}{f'(B_n)}$$

The main idea of EENN is to automatically optimize the structure of the ENN, because most of the NNs' algorithms can not create optimal networks' structure

Chapter 7 Conclusions and Future Work

automatically. The number of the hidden nodes, the number of the component networks in the ensemble network must be given in the traditional algorithms. Certainly, the detailed algorithm of EENN still needs to be studied further in the future.

In order to demonstrate the EENN performance, the peak function is used, which is a typical complex two-dimensional function (Equation (4.11)). 21×21 evenly distributed data along both the x-axis and the y-axis are selected from the domain $(-3 \leq x \leq 3, -3 \leq y \leq 3)$ as training data for the simulation. Another 20×20 evenly distributed data from the domain $(-2.85 \leq x \leq 3, -2.85 \leq y \leq 3)$ are used as test data. The maximum training epoch of each component network is set to 30. There are 441 examples in the training data set and 400 examples in the test data set. The optimal number of the hidden nodes is selected as 24 by the trial and error method. The single NN has 24 hidden nodes in its hidden layer. In the ENNs, there are several component networks. The component networks in simple averaging ENN are same as those used in EENN. The number of the original component networks in ENN is set to 10. The numbers of hidden nodes in the original component networks are 15, 16, 17, 18, 19, 20, 21, 22, 23 and 24, respectively. Then, the numbers of hidden nodes in each component network are selected by entropy based algorithm. The pre-defined error is 0.01. Iteration of the Newton's method is set to 2. The original value of B is set to 1000. For EENN, the Equation (7.2) will be used to determine the component networks' weights.

There are 20 random runs for each method. MSE is selected as the criterion for comparing. Table 7.1 lists the number of the component networks in ENNs and the weights of the component networks in EENN. Table 7.2 lists the number of hidden nodes of component networks in EENN. From Tables 7.1 and 7.2, it can be found there are 3 to 7 networks are selected to be the component networks in the ensemble

Chapter 7 Conclusions and Future Work

networks. For example, the first run, there are 6 component networks are selected in the ENNs. The numbers of the hidden nodes of the component networks in the ENNs are 16, 17, 21, 22, 23 and 24, respectively, and the weights of the component networks in EENN are 0.175, 0.165, 0.175, 0.162, 0.164 and 0.159, respectively.

Table 7.1 The number of the component networks in ENNs and weights of the component networks in EENN

run	no. of component network	weight 1	weight 2	weight 3	weight 4	weight 5	weight 6	weight 7
1	6	0.175	0.165	0.175	0.162	0.164	0.159	
2	4	0.236	0.245	0.259	0.260			
3	6	0.167	0.163	0.170	0.165	0.172	0.163	
4	6	0.156	0.181	0.161	0.170	0.168	0.163	
5	7	0.143	0.148	0.138	0.142	0.144	0.151	0.134
6	6	0.175	0.168	0.165	0.156	0.172	0.164	
7	6	0.158	0.166	0.173	0.172	0.158	0.173	
8	5	0.182	0.207	0.197	0.201	0.213		
9	3	0.318	0.330	0.352				
10	5	0.192	0.201	0.211	0.195	0.201		
11	5	0.186	0.202	0.204	0.206	0.202		
12	4	0.244	0.248	0.266	0.243			
13	6	0.166	0.163	0.168	0.158	0.173	0.173	
14	5	0.203	0.193	0.204	0.202	0.199		
15	6	0.172	0.161	0.166	0.167	0.165	0.169	
16	4	0.259	0.239	0.262	0.239			
17	4	0.251	0.244	0.259	0.246			
18	4	0.264	0.245	0.242	0.250			
19	5	0.196	0.221	0.200	0.186	0.197		
20	5	0.210	0.199	0.200	0.197	0.194		

Table 7.2 The number of the hidden nodes of the component networks in EENN

run	component network 1	component network 2	component network 3	component network 4	component network 5	component network 6	component network 7
1	16	17	21	22	23	24	
2	15	16	21	23			
3	15	16	17	20	22	24	
4	15	17	19	21	23	24	
5	15	16	18	19	20	22	24
6	17	18	20	21	23	24	
7	15	17	20	21	23	24	
8	16	19	22	23	24		
9	18	20	23				
10	15	17	21	23	24		
11	15	16	21	22	24		
12	15	16	20	23			
13	15	17	20	22	23	24	
14	16	17	19	20	24		
15	16	18	19	20	21	24	
16	17	20	21	22			
17	15	17	19	23			
18	17	19	21	22			
19	15	18	19	20	23		
20	15	16	17	18	23		

The comparison of the MSE in 20 run of these three methods is listed in Table 7.3. From Table 7.3, it can be found that the EENN can obtain the best result in 20 randomly continuous runs, and it is stable to get the good result. Certainly, in order to optimize the entropy algorithm further, changing some parts of the algorithm (such as the improved methods to determine the weight of the component networks) in EENN will be the most promising direction.

Table 7.3 Results of twenty runs on peak function in EENN

		Single 24	Ave-ENN	EENN
Test-MSE	Minimum	0.073213	0.085882	0.085920
	Mean	0.119714	0.116377	0.116028
	S.D.	0.035123	0.026850	0.026907
Train-MSE	Minimum	0.073213	0.066584	0.066646
	Mean	0.118620	0.097229	0.096889
	S.D.	0.037239	0.023734	0.023746

7.3.2 Model Application in Geological Problems

Rock failure is an old research subject for rock mechanics. Brittle fracture is a failure phenomenon that has been intensively investigated in rock physics and geotechnical engineering. Generally, many engineering geological problems are characterized as being complex, dynamic, uncertain and ill-defined. Complexity arises due to various reasons: lack of data, lack of knowledge, and inherently uncertainty in the system, such as rock microstructure. The distribution of microcracks can be described the macroscopic failure behavior.

Statistics of the mechanical and failure properties on the grain scale are often assumed to follow the Weibull distribution (Weibull, 1951) in numerical simulations of failure and damage development. To investigate the microstructural basis for such a statistical model of compressive failure in a brittle rock, the rock material, the microcrack density, length and orientation can be applied to develop a crack model. Since this is a complex model in high dimensions, ENN will be a promising method to create the creative crack model.

PUBLICATIONS

Zhang, Y. and Zhao, Z. Y. (2006), “Development of ensemble neural network by Akaike information criterion”, Proceedings of the third International Conference on Artificial Intelligence in Engineering and Technology ICAIET-2006.

Zhao, Z. Y., Gong, Q. M., Zhang, Y. and Zhao, J. (2007), “Prediction model of tunnel boring machine performance by ensemble neural networks”, Geomechanics and Geoengineering, Vol. 2, No. 2, pp. 123-128.

Zhao, Z. Y., Zhang, Y. and Liao, H. J. (2008), “Design of ensemble neural network using the Akaike information criterion”, Engineering Applications of Artificial Intelligence, Vol. 21, pp. 1182-1188.

Zhao, Z.Y., Zhang, Y. and Wei, X.Y. (2009) “Discontinuous deformation analysis for parallel hole cut blasting in rock mass”, Proceedings of the ninth International Conference on Analysis of Discontinuous Deformations ICADD-9.

Zhao, Z. Y. and Zhang, Y. Dynamic fracturing performance of rock under blast load by ensemble neural network, under preparation.

REFERENCE

Adeli, H. (2001), "Neural Networks in civil Engineering: 1989-2002", Computer-aided Civil and Infrastructure Engineering, Vol.16, pp.126-142.

Adeli, H. and Yeh, C. (1989), "Perceptron learning in engineering design", Microcomputers in civil engineering, Vol. 4, No. 4, pp. 247-256.

Adhikari, G. A., Rajan Babu, H. S. and Venkatesh & A. I. Thresraj. (1994), "Blast damage assessment for underground structures", Proc. National Symp. on emerging mining and ground control technologies, Varanasi, India, pp. 247-255.

Aitkenhead, M. J. and McDonald, A. J. S. (2003), "A neural network face recognition system", Engineering Applications of Artificial Intelligence, Vol. 16, No. 3, pp.167-176.

Akaike, H. (1973), "Information theory and an extension of the maximum likelihood principle", 2nd International Symposium on Information Theory, pp. 267-281.

Alber, M. (1996), Prediction of penetration and utilization for hard rock TBMs. Eurock'96, Barla G.(Ed.), pp.721-725, (Balkema: Rotterdam).

Alber, M. (2000), "Advance Rates of Hard Rock TBMs and Their Effects on Project Economics". Tunnelling And Underground Space Technology, Vol. 15, No. 1, pp. 55-64.

Barron, A. (1992), "Neural net approximation", Proceedings of the Seventh Yale

Reference

Workshop on Adaptive and Learning Systems, Yale University, New Haven, CT, pp. 69–72.

Barton, N. (2000), TBM Tunnelling in jointed and faulted rock, pp.172 (A. A. Balkema: Rotterdam, Brookfield).

Baum, E. and Haussler, D. (1989), “What size net gives valid generalization?”, Neural Computation, pp. 151-160.

Bolt, G. R. (1992), “Fault tolerance in artificial neural networks”, Ph.D. Thesis, York University, Ontario.

Breiman, L. (1996), “Bagging predictors”, Machine Learning, Vol. 24, No. 2, pp. 123-140.

Bruines, P. A. (2001), “The use of neurofuzzy modeling for performance prediction of tunnel boring machines”, Modern Tunnelling Science and Technology, Adachi, Swets & Zeitlinger (Eds) , 2001, pp. 583-588.

Bruland, A. (1998), “Hard rock tunnel boring”, Ph.D. Thesis, Norwegian University of Science and Technology, Trondheim.

Burnham, K. P. and Anderson, D. R. (2002), Model Selection and Multimodel Inference: a practical information-theoretic approach, 2nd edition. Springer-Verlag, New York.

Campbell, N. A., Reece, J. B. and Mitchell, L. G. (1999), Biology, California: Benjamin/Cummings.

Cannon, A. J. and Whitfield, P. H. (2002), “Downscaling recent streamflow conditions in British Columbia, Canada using ensemble neural network modes”. Journal of Hydrology, Vol. 259, pp. 136-151.

Reference

Carney, J. G. and Cunningham, P. (1999), "The NeuralBAG algorithm: Optimizing generalization performance in bagged neural networks", Proceedings of the 7th European Symposium on Artificial Neural Networks, pp. 35-40.

Carney, J. and Cunningham, P. (2000), "Tuning Diversity in Bagged Ensembles", International Journal of Neural Systems, Vol. 10, pp. 267-280.

Chakik, F. El, Shahine, A., Jaam, J. and Hasnah, A. (2004), "An approach for constructing complex discriminating surfaces based on Bayesian interference of the maximum entropy", Information Sciences, Vol. 163 pp. 275-291.

Chatterjee, T. N. (2001), "On the application of information theory to the optimum state-space reconstruction of the short-term solar radio flux (10.7cm), and its prediction via a neural network", Monthly Notices of the Royal Astronomical Society, Vol. 323, pp. 101-108.

Chen, Y. H., Yang, B. and Abraham, A. (2007), "Flexible neural trees ensemble for stock index modeling", Neurocomputing, Vol.70, pp. 697-703.

Cun, Y. L., Bottou, L., Orr, G. B. and Müller, K-R. (1998), Efficient backprop, In G. B. Orr and K-R. Müller, editors, Neural Networks: Tricks of the Trade, Vol. 1524 of Lecture Notes in Computer Science, chapter 1, pp. 9–50. Springer.

Cybenko, G. (1989), "Approximation by superpositions of a sigmoidal function", Mathematics of control, Signal and Systems, Vol. 2, No.4, pp. 303-314.

Dietterich, T. G., Lathrop, R. H. and Lozano-Perez (1977), "Solving the multiple-instance problem with axis-parallel rectangle", Artificial Intelligence, Vol. 89, No.1-2, pp.31-71

Dondeti, S., Kannan, K. and Manavalan, R. (2005), "Genetic algorithm optimized neural networks ensemble for estimation of mefenamic acid and paracetamol in

Reference

tablets”, Acta Chimica Slovenica, Vol. 52, pp. 440-449.

Efron, B. and Tibshirani R. (1993), An Introduction to the Bootstrap, Chapman & Hall, New York.

Esteves, P.A. Figueroa, C. J. and Saito, K. (2005), “Cross-entropy embedding of high-dimensional data using the neural gas model” Neural Networks, Vol. 18, pp. 727-737.

Farmer, I. W. and Glossop, N. H. (1980), “Mechanics of Disc Cutter Penetration”, Tunnels and Tunnelling International , Vol. 12, No. 6, pp. 22-25.

Fausett, L. V. (1994), Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Englewood Cliffs, NJ: Prentice-Hall.

Fiori, S. (1999), “Entropy optimization by the PFANN network: application to blind source separation” Network: Computation in Neural System, Vol. 10, pp. 171-186.

Fourie, A.B. and Green, R.W. (1993), “Damage to underground coal mines caused by surface blasting”, International Journal of Surface Mining and Reclamation, Vol. 7, No.1, pp. 11–16.

Freund, Y. (1995). “Boosting a weak algorithm by majority”, Information and Computation, Vol. 121, No.2, pp. 256-285.

Freund, Y. and Schapire, R. E. (1996), “Experiments with a new boosting algorithm”, Proceedings of the Thirteenth International Conference on Machine Learning, pp. 149-156. Morgan Kaufmann.

Freund, Y. and Schapire, R. E. (1995), “A decision-theoretic generalization of on-line learning and an application to boosting”, Proc. EuroCOLT-94, Barcelona, Spain, Springer, Berlin, pp. 23-37.

Reference

Friedman, J. H. (1991), "Multivariate adaptive regression splines", Annals of Statistics, Vol. 19, No.1, pp. 1-82.

Funahashi, K. I. (1989), "On the approximate realization of continuous mappings by neural networks", Neural networks, Vol. 2, No. 3, pp. 193-192.

Gavin, B., Jeremy, L. Wyatt and Peter Tino (2005), "Managing diversity in regression ensembles", Journal of Machine Learning Research, Vol. 6, pp. 1621-1650.

Gavin Brown, (2004). "Diversity in Neural Network Ensemble", Ph.D. Thesis, University of Birmingham, United Kingdom.

Geman, S., Bienenstock, E. and Doursat, R. (1992), "Neural Networks and the Bias/Variance Dilemma". Neural Computation, Vol. 4, pp. 1-58.

Giacinto, G. and Roli, F. (2001), "Design of effective neural network ensembles for image classification purposes", Image and Vision Computing Journal, Vol. 19, pp. 699-707.

Gong, Q. M. (2006), "Development of a rock mass characteristics model for TBM penetration rate prediction", Ph.D. Thesis, Nanyang Technological University, Singapore.

Gong, Q. M., Zhao, J. and Jiang, Y. S. (2007), "In situ TBM penetration tests and rock mass boreability analysis in hard rock tunnels", Tunnelling and Underground Space Technology, Vol. 22, No. 3, pp. 303-316.

Gong, Q. M., and Zhao, J. (2007), "Influence of rock brittleness on TBM penetration rate in Singapore Granite", Tunnelling and Underground Space Technology, Vol. 22, pp. 317-324.

Graham, P. C. (1976), "Rock exploration for machine manufacturers, in exploration

Reference

for rock engineering”. Bieniawski Z. T. (ed.), Proceedings of the Symposium, Johannesburg, Balkema, Vol. 1, pp.173-180.

Grandori, R., Sem, M., Lembo-Fazio, A. and Ribacchi, R. (1995), “Tunnelling by double shield TBM in the Hong Kong granite”, Proceedings of 8th Int. Cong. for Rock Mechanics, Vol.1, pp.569-574.

Granitto, P. M., Verdes, P. F. and Ceccatto, H. A. (2005), “Neural network ensembles: evaluation of aggregation algorithms”, Artificial Intelligence, Vol. 163, pp. 139-162.

Granitto, P. M., Verdes, P. F., Navone, H. D. and Ceccatto H. A. (2001), “A late-stopping method for optimal aggregation of neural networks”, International Journal of Neural Systems, Vol. 11, No. 3, pp. 305-310.

Grima, M. A., Bruines, P. A. and Verhoef, P. N. W. (2000), “Modeling tunnel boring machine performance by Neuro-fuzzy methods”, Tunnelling and Underground Space Technology, Vol. 15, No. 3, pp. 259-269.

Hamilton, W. H. and Dollinger, G. L. (1979), “Optimizing tunnel boring machine and cutter design for greater boreability”, RETC Proceedings, Atlanta, Vol. 1, pp. 280-296.

Hansen, L. K. and Salamon, P. (1990), “Neural network ensembles”, IEEE Trans. Pattern Analysis Machine Intelligence Vol. 12, No. 10, pp. 993-1001.

Hao, H., Wu, Y., Ma, G., Zhou, Y., (2001), “Characteristics of surface ground motions induced by blasts in jointed rock mass”, Soil Dynamics and Earthquake Engineering, Vol. 21, No. 2, pp. 85-98.

Hartley, R. V. (1928), “Transmission of information,” Bell Systems Technical Journal, Vol. 7, pp.535-563.

Reference

- Hartman, E. J., Keeler, J. D. and Kowalski, J. M. (1990), "Layered neural networks with Gaussian hidden units are universal approximations", Neural Computation, Vol. 2, No. 2, pp. 210-215.
- Hashem, S. (1993), "Optimal Linear Combinations of Neural Networks", Ph.D. Thesis, School of Industrial Engineering, Purdue University, Dec.
- Hassoun, M., Cherkassky, V., Hanson, S., Oja, E., Sarle, W. and Sudjianto, A. (1996), "Panel on neural networks and statistical models", in 'International Conference on Neural Networks, ICNN 96', IEEE, Piscataway, NJ, pp. 18–22.
- Hayashi, Y. and Setiono, R. (2002), "Combining neural network Predictions from medical diagnosis", Computers in Biology and Medicine, Vol. 32, pp. 237-246.
- Haykin, S. S. (1999), Neural networks: a comprehensive foundation, 2nd ed., Prentice Hall.
- Hendron, A. J. (1977), "Engineering of rock blasting on civil projects", In: hall, W.J.(ed.) Structural and Geotechnical Mechanics. A Volume Honoring N. M. Newmark, Prentice-Hall, New Jersey, pp. 242-277
- Hornik, K., Stinchcombe, M. and White, H. (1989), "Multilayer feedforward networks are universal approximators", Neural Networks, Vol. 2, No. 5, pp. 359-366.
- Hughes, H. M. (1986), "The relative cuttability of coal measures rock", Mining Science and Technology, Vol. 3, pp. 95-109.
- Innaurato, N., Mancini, R., Rondena, E. and Zaninetti, A. (1991), "Forecasting and effective TBM performance in a rapid excavation of a tunnel in Italy", Wittke W. (ed.), Proceedings of the 7th International Congress on Rock Mechanics, Aachen, pp.1009-1014.

Reference

Islam, M. Md., Yao X., and Murase, K. (2003), "A constructive algorithm for training cooperative neural network ensembles", IEEE Transactions on Neural Networks, Vol. 14, No.4, pp. 820-834.

Jacobs, R. A. (1995), "Methods for combining experts' probability assessments", Neural computation, Vol. 7, pp. 867-888.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J. and Hinton, G. E. (1991), "Adaptive mixtures of local experts", Neural Computation, Vol. 3, No.1, pp. 79-87.

Jacobsand, R. A. and Tanner, M. A. (1999), Combining Artificial Neural Nets. Springer-Verlag, London.

Jakubek, S. M. and Strasser, T. I. (2004), "Artificial neural networks for fault detection in large-scale data acquisition systems", Engineering Applications of Artificial Intelligence, Vol. 17, pp. 233-248.

Jaynes, E.T. (1957), "Information theory and Statistical Mechanics I", Physical Review, Vol. 106, pp. 620-630.

Jaynes, E.T. (1957), "Information theory and Statistical Mechanics II", Physical Review, Vol. 108, pp. 171-190.

Jiang, N., Zhao, Z.Y. and Ren, L.Q. (2003), "Design of structural modular neural networks with genetic algorithm", Advances in engineering software, Vol. 34, pp. 17-24.

Jolliffe, IT. (1986), Principal component analysis. Berlin: Springer.

Judgy, G. G., Griffiths, W. E., Hill, R. C. and Lee, T. (1980), The theory and practice of econometrics, New York: Wiley.

Kenig, S., Ben-David, A., Omer, M. and Sadeh, A. (2001), "Control of properties in

Reference

injection molding by neural networks”, Engineering Applications of Artificial Intelligence, Vol. 14, pp. 819-823.

Kennedy, J. and Eberhart, R. (1997), “A discrete binary version of the particle swarm optimization”, In Proceedings IEEE international conference on computational cybernetics and simulation.

Khaw, J.F.C., Lim, B.S. and Lim, L.E.N. (1995), “Optimal design of neural networks using the Taguchi method”, Neurocomputing, Vol. 7, pp. 225-245.

Kidybinski, A. (1986), “Design criteria for roadway supports to resist dynamic loads”, International Journal of Mining Engineering, Vol. 4, pp. 91–109.

Kim, C. Y., Bae, G. J., Hong, S. W., Park, C. H., Moon, H. K. and Shin, H. S. (2001), “Neural network based prediction of ground surface settlements due to tunneling”, Computers and Geotechnics, Vol. 28, pp. 517-547.

Krogh, A., and Vedelsby, J., (1995), “Neural network ensembles, cross validation and active learning”, In Tesauro, G., Touretzky, D., & Leen, T., editors, Advances in Neural Information Processing Systems Vol. 7, pp. 231-238 Cambridge, MA. MIT Press.

Kullback, S. (1959), Information Theory and Statistics, Willey & Sons, New York.

Kung, S. Y. and Hwang, J. N. (1988), “An algebraic projection analysis for optimal hidden units size and learning rates in back-propagation learning”, Proceeding of Int. Conf. Neural Networks, San Diego, CA, July 1988, pp. 363-370.

Lagaros, N. D., Charmpis, D. C. and Papadrakakis, M. (2005), “An adaptive neural network strategy for improving the computational performance of evolutionary structural optimization”, Computer Methods in Applied Mechanics and Engineering, Vol. 194, pp. 3374-3393.

Reference

Lai, K. K., Yu, L., Wang, S. Y. and Huang, W. (2006), "A Novel Nonlinear Neural Network Ensemble Model for Financial Time Series Forecasting", International Conference on Computational Science, Vol. 1, pp.790-793.

Laughton, C. and Nelson, P. P. (1996), "The development of rock mass parameters for use in the prediction of tunnel boring machine performance", Barla G. (ed.), Eurock'96, pp.727-733, (Balkema: Rotterdam).

Lawrence, S., Giles, C. L. and Tsoi, A. C. (1996), "What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation", Technical Report, Institute for Advanced Computer Studies, University of Maryland.

Le Cun, Y. (1985), "A Learning Procedure for Asymmetric Threshold Network", Proceedings of Cognitive'85. (In French), Paris, pp. 599-604.

Li, H., Bao, Y. Q. and Ou, J.P. (2008), "Structural damage identification based on integration of information fusion and Shannon entropy", Mechanical Systems and Signal Processing, Vol. 22, pp. 1427-1440.

Li, Z. and Huang, H. (1994), "The calculation of stability of tunnels under the effects of seismic wave of explosions", Proceedings of the 26th Department of Defense Explosives Safety Seminar, USA, Department of Defense Explosives Safety Board.

Li, Y. J. (1999), "A method to directly estimate the number of the hidden neurons in the feedforward neural networks", Journal of Computer, Vol. 22, No.11, pp.1204-1208.

Liao, H., Ning, C., Akaishi, M. and Zhou, L. (1998), "Effects of the time-dependent behavior on strain softening of diatomaceous soft rock", Metals and Materials, Vol. 4, No. 5, pp. 1093-1096.

Lin, T. Y. and Tseng, C. H. (2000), "Optimum design for artificial neural networks:

Reference

an example in a bicycle derailleur system”, Engineering Applications of Artificial Intelligence, Vol.13, pp. 3-14.

Liu, Y. (1998), “Negative Correlation Learning and Evolutionary Neural Network Ensembles”, Ph.D thesis, University College, The University of New South Wales, Australian Defence Force Academy, Canberra, Australia.

Liu, Y., Yao, X. and Higuchi, T. (2000), “Evolutionary ensembles with negative correlation learning”, IEEE Transactions on Evolutionary Computation, Vol. 4, pp. 380-387.

Liu, F., Sun, C. X., Si-ma, W. X., Liao, R. J. and Guo, F. (2006), “ Chaos control of ferroresonance system based on RBF-maximum entropy clustering algorithm”, Physics Letters A, Vol. 357, pp. 218-223.

Maron, O. (1998), “Learning from Ambiguity”, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT.

McCulloch, W. S. and Pitts, W. (1943), “A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133. Reprinted in Anderson & Rosenfeld [1988], pp. 18-28.

McFeat-Smith, I. (1999), “Mechanised tunnelling for Asia”, Workshop Manual, Organized by IMS Tunnel Consultancy LTD.

McFeat-Smith, I. and Askilsrud, O. G. (1993), “Tunnel boring Machines in Hong Kong”, RECT Proceedings, pp.401-413.

Mead, C. A. and Conway, L. (1980), Introduction to VLSI systems, Reading, MA: Addison Wesley.

Middleton, D. (1964), Topics in communication theory, New York: McGraw Hill.

Reference

Moore, A., Schneider, J. and Deng, K. (1997), "Efficient locally weighted polynomial regression predictions", Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 236–244.

Morimoto, T. and Hori, M. (1986), "Performance Characteristics of a Tunnel Boring Machine from the Geomechanical Viewpoint". International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts, Vol. 23, No. 1, pp. 55-66.

Naftaly, U., Intrator, N. and Horn, D. (1997), "Optima ensemble averaging of neural networks", Network: Computation in Neural System, Vol. 8, pp. 283-296.

Nelson, P. P., Yousof, A. Al-Jalil and Laughton, C. (1999), "Improved strategies for TBM performance prediction and project management", RETC Proceedings, pp.963-979.

Nelson, P. P. (1983), "Tunnel boring Machine performance in sedimentary rock", Ph.D. Thesis, the Graduate School of Cornell University, USA.

Ng, G. S., Wahab, A. and Shi, D. (2003), "Entropy learning and relevance criteria for neural network pruning", International Journal of Neural Systems, Vol. 13, No.5, pp.291-305.

O'Rourke J. E., Spring J. E. and Coudray S. V. (1994), "Geotechnical parameters and tunnel boring machine performance at Goodwill Tunnel", California. Nelson & Laubach (eds), Rock Mechanics Models and Measurements Challenges from Industry, Proc. of the 1st North American Rock Mechanics Symposium, The University of Texas at Austin, (A. A. Balkema: Rotterdam).

Opitz, D.W. and Shavlik, J.W. (1996), "Actively searching for an effective neural network ensemble", Connection Science, Vol. 8, No.3-4, pp.337-353.

Reference

Parker, D. B. (1985), "Learning-Logic", TR-47, MIT, Center for Computational Research in Economics and Management Science, Cambridge, MA.

Pelagotti A. and Piuri, V. (1997), "Entropic analysis and incremental synthesis of multilayered feedforward neural networks", International Journal of Neural Systems, Vol. 8, No.5-6, pp. 647–659.

Perrone, M. P. and Cooper, L. N. (1993), "When networks disagree: Ensemble method for neural networks", in: R.J. Mammone (Ed.), Artificial Neural Networks for Speech and Vision, Chapman & Hall, New York, pp. 126-142.

Persson, P. A. (1997), "The relationship between strain energy, rock damage, fragmentation, and throw in rock blasting", International Journal of Blast Fragmentation, Vol. 1, pp. 99-110.

Pineda, F. J. (1988), "Dynamics and Architecture for Neural Computation", In Journal of Complexity 4, 1988.

Puerto, F. D. and Ghalia, M. B. (2002), "White color tracking adjustment in television receivers using neural networks", Engineering Applications of Artificial Intelligence, Vol.15, pp. 601-606.

Reich, Y. and Barai, S. V. (2000), "A methodology for building neural networks models from empirical engineering data", Engineering Applications of Artificial Intelligence, Vol. 13, pp. 685-694.

Ren, L. Q. and Zhao, Z. Y. (2002), "An optimal neural network and concrete strength modeling", Advances in Engineering Software Vol. 33, pp.117-130.

Renyi, A. (1961), "On the measure of entropy and information," Proceedings of Fourth Berkeley Symposium on Mathematical Statistics and Probability, vol.1, pp.541-561.

Reference

Ribacchi R. and Lembo-Fazio A. (2005), “Influence of rock mass parameters on the performance of a TBM in a gneissic formation (Varzo Tunnel)”, Rock Mechanics and Rock Engineering, Vol. 38, No.2, pp. 105-127.

Rocha, M., Cortez, P. and Neves, J. (2007), “Evolution of neural networks for classification and regression”, Neurocomputing, Vol. 70, pp. 2809-2816.

Rosen, B. E. (1996), “Ensemble learning using decorrelated neural networks”, Connection Science, Vol. 8, No. 3-4, pp. 373-383.

Rosenblatt, F. (1958), “The perceptron: a probabilistic model for information storage and organization in the brain”, Psychological Review, Vol. 65, pp. 386-408.

Rostami, J. (1997), “Development of a force estimation model for rock fragmentation with disc cutters through theoretical modeling and physical measurement of crushed zone pressure”, Ph.D. Thesis, Dept. of Mining Engineering, Colorado School of Mines, Golden, Colorado, USA.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), “Learning internal representations by error propagation”, Chapter 8 in Parallel Distributed Processing: Foundation, Vol. 1, MIT Press, Cambridge, MA, pp. 318-362.

Rupert, G. B. and Clark, G. B. (1977), “Criteria for the proximity of surface blasting to underground coal mines”, Proceedings Rock Mechanics Symposium, Colorado School of Mines, pp. 3C31–3C310.

Santos, J.M., Alexandre, L.A. and Marques de S’a. J. (2004), “The Error Entropy Minimization Algorithm for Neural Network Classification”, In Int. Conf. on Recent Advances in Soft Computing, Nottingham, United Kingdom.

Sapigni, M., Berti, M., Bethaz, E., Busillo, A. and Cardone, G. (2002), “TBM performance estimation using rock mass classifications”, International Journal of

Reference

Rock Mechanics & Mining Sciences, Vol. 39, pp. 771-788.

Schapire, R. E. (1990), "The strength of weak learnability", Machine Learning Vol.5, No.2, pp.197-227.

Schraudolph, N. N. (1995), "Optimization of entropy with neural networks", Ph.D. thesis, University of California.

Seibi, A. and Al-Alawi, S. M. (1997), "Prediction of fracture toughness using artificial neural networks (ANNs)", Engineering Fracture Mechanics, Vol. 56, pp. 311-319.

Shannon, C. E. and Weaver W. (1964), The mathematical Theory of Communication, University of Illinois Press, Urbana.

Shannon, C. E. (1948), "A mathematical theory of communication," Bell Systems Technical Journal, Vol. 27, pp. 379-423; 623-656.

Sharkey, A. (Ed.), (1999), Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems. Springer, London.

Sherrod, P. H. (1996), Nonlinear Regression Analysis Program (NLREG), Version 3.2, Phillip H. Sherrod, Nashville, TN.

Sietsma, J. and Dow. R. J. (1988), "newblock Neural net pruning---Why and how." newblock In IEEE International Conference on Neural Networks, pp. 325-333. IEEE, New York, 1988.

Singh, P. K. (2002), "Blast vibration damage to underground coal mines from adjacent open-pit blasting", International Journal of Rock Mechanics and Mining Sciences, Vol.39, No.8, pp. 959-973.

Singh, M., Seshagiri Rao, K., (2005). "Empirical methods to estimate the strength

Reference

of jointed rock masses”, Engineering Geology, Vol. 77, No. 1-2, pp. 127-137.

Sollich, P. and Krogh, A. (1996), “Learning with ensembles: How over-fitting can be useful”, in: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (Eds.), Advances in Neural Information Processing Systems 8, Denver, CO, MIT press, Cambridge, MA, pp. 190-196.

Sugiura, N. (1978), “Further analysis of data by Akaike's information criterion and the finite corrections”, Communications in Statistics A, Vol.7, pp.12-26.

Sun, W., Lu, J. and He, Y. (2005), “Information entropy based neural network model for short-term load forecasting”, In Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005 IEEE/PES, pp. 1–5.

Sundin, N. O. and Wanstedt, S. (1994), “A boreability model for TBM's”, Nelson & Laubach (eds), Rock Mechanics, Balkema, Rotterdam, pp8.

Ting, K. M. (1996), “The characterisation of predictive accuracy and decision combination”. Proceedings of the 13th International Conference on Machine Learning, pp. 498-506.

Tortum, A., Yayla, N., Celik, C. and Gokdag, M. (2007), “the investigation of model selection criteria in artificial neural networks by the Taguchi method”, Physics A Vol. 386 pp. 446-468.

Tribus, M. (1978), Thirty years of information theory in the maximum entropy formalism, R.D. Levine and M. Tribus (eds), M.I.T.Press, pp.1-14.

Tumer, K. and Ghosh, J. (1996), “Error correlation and error reduction in ensemble classifiers”, Connection Science. Special Issue on Combining Artificial Neural: Ensemble Approaches, Vol. 8, No. 3&4, pp. 385-404.

Ueda, N. and Nakano, R. (1996), “Generalization error of ensemble estimators”,

Reference

Proceedings of International Conference on Neural Networks, pp 90-95.

Vapnik, V. (1995), The Nature of Statistical Learning Theory, New York: Springer-Verlag

Wanner, H. and Aeberli, U. (1979), “Tunnelling machine performance in jointed rock”, Proceedings of 4th Congress of the International Society for Rock Mechanics, Montreux, Vol. 1, pp.573-580.

Weibull, W. (1951). “A statistical distribution function of wide applicability” Journal of Applied Mechanics, Vol. 18, pp. 293-297.

Werbos, P. (1974), “Beyond Regressions New Tools for Prediction and Analysis in the Behavioral Science”, M. Sc. Thesis, Applied Mathematics, Harvard University, Boston, MA.

Widrow, B. and Hoff, M. E. (1960), “Adaptive switching circuits”, In 1960 IRE WESCON Conv. Records, Vol. 4, pp. 96–104. Institute of Radio Engineers (now IEEE).

Wiener, N. (1961), Cybernetics, New York: John Wiley.

Wilhelm, T. and Hollunder, J. (2007), “Information theoretic description of networks”, Physica A, Vol. 385, pp. 385-396.

Wolpert, D. (1992). “Stacked generalization”, Neural Networks, Vol.5, pp.241-259.

Wolpert, D. (1996). “The lack of a priori distinctions between learning algorithms”, Neural Computation, Vol.8, pp.1341-1390.

Yu, J. B. and Xi, L. F.(2009), “A neural network ensemble-based model for on-line monitoring and diagnosis of out-of-control signals in multivariate manufacturing processes”, Expert Systems with Applications. Vol. 36, No. 1, pp. 909-921.

Reference

Yu, L., Wang, S. Y. and Lai, K. K. (2005), “A novel nonlinear ensemble forecasting model incorporating GLAR and ANN for foreign exchange rates”, Computers & Operations Research, Vol. 32, pp. 2523-2541.

Yuan, H. C., Xiong, F. L., and Huai, X. Y. (2003), “A method for estimating the number of hidden neurons in feed-forward neural networks based on information entropy”, Computers and Electronics in Agriculture, Vol. 40, pp. 57-64.

Zell, A., Mamier, G., and et al. (1995), SNNS User Manual. University of Stuttgart, 1995. Version 4.1.

Zhang, M. L. and Zhou, Z. H. (2004), “Ensembles of multi-instance neural networks”, Proceedings of the International Conference on Intelligent Information Processing (ICIIP'04), Beijing, China, pp.471-474.

Zhang, M. J. (2000), “A domain independent approach to 2D object detection based on the neural and genetic paradigms”, Ph.D thesis, Department of Computer Science, Faculty of Applied Science RMIT University, Melbourne, Victoria, Australia.

Zhao, J., Gong, Q. M. and Eisensten, Z. (2007), “Tunnelling through a frequently changing and mixed ground: A case history in Singapore”, Tunnelling and Underground Space Technology, Vol. 22, No. 4, pp. 388-400.

Zhao, Z. Y. (2006), “Steel column under fire – a neural network based strength model”, Advances in Engineering Software, Vol. 37, No. 2, pp.97-105.

Zhao, Z. Y., Gong, Q. M., Zhang, Y. and Zhao, J. (2007), “Prediction model of tunnel boring machine performance by ensemble neural networks”, Geomechanics and Geoengineering, Vol. 2, No. 2, pp. 123-128.

Zhao, Z. Y., Zhang, Y. and Liao, H. J. (2008), “Design of ensemble neural network

Reference

using the Akaike information criterion”, Engineering Applications of Artificial Intelligence, Vol. 21, pp. 1182-1188.

Zheng, J. (2009), “Predicting software reliability with neural network ensembles”, Expert Systems with Applications, Vol. 36, No. 2, pp. 2116-2122.

Zhou, Z.H., Wu, J. and Tang, W. (2002), “Ensembling neural networks: many could be better than all”, Artificial Intelligence, Vol. 137, pp. 239-263.

APPENDIX I TBM EXPERIMENT DATA

J_v	α	UCS (MPa)	B_i	SRMBI
0	0	194.86	15.73	210.4
0	0	176.86	14.57	195.17
0.2	70	193.27	17.09	190.02
0.4	50	177.2	13.15	201.96
0.4	10	200.57	14.93	202.33
0.4	35	175.83	14.52	174.63
0.7	70	217.27	19.13	194.4
1.1	40	178.02	17.61	191.59
1.2	5	163.94	12.91	190.36
1.5	62.5	107.92	9.99	164.39
1.7	30	214.34	20.93	198.91
2	30	172.9	16.39	193.74
2.5	30	175.63	13.91	193.57
2.7	45	162.43	14.39	163.05
3	20	178.37	15.72	185.72
3	60	236.51	19.94	158.92
3.3	10	169.47	12.36	172.3
4.5	25	121.6	10.03	178.81
4.5	65	177.04	9.29	182.18
4.6	35	154.6	13.7	189.45
5	75	166.33	16.31	170.76
5	20	172.93	14.48	147.23
5	70	165	14.95	157.52
5.3	20	240.3	19.68	164.17
6	20	169	14.31	147.7
6	35	67.39	6.23	139.87
6	45	149.35	15.87	182.93
6.2	30	180.53	13.31	187.56

Appendix I TBM Experiment Data

6.7	30	228.95	14.12	168.43
7	20	107.59	10	144.89
8.3	67	180.27	14.69	165.45
9.6	20	130.7	10	140.35
10.3	45	164.97	14.42	157.95
10.4	60	165.6	14.72	132.78
10.6	45	201.44	14.45	177.52
10.9	30	212.54	15.43	181.97
11.2	80	170.17	16.24	149.54
12.2	75	148.6	10	139.49
12.3	25	162.06	10.45	154.02
13.3	20	196.5	12.96	160.92
14	75	147.02	10.56	133.35
16.1	45	195.34	15.11	137.43
20	30	103.95	8	134.66
26.7	45	236.79	15.7	146.35

*Appendix II Mudstone Experiment Data***APPENDIX II MUDSTONE EXPERIMENT DATA**

$\tilde{e}$	e	p'	q
0.0044	0	2500	0
0.0044	0.255	2398.7	501.8
0.0044	0.707	2150.3999	943.6
0.0044	1.046	1981.4	1161.6
0.0044	1.512	1797.3	1348.9
0.0044	1.738	1721.7	1400.7
0.0044	2.317	1561.4	1447
0.0044	2.572	1510.7	1470.5
0.0044	3.01	1443.5	1518.1
0.0044	3.476	1366.2	1483.8
0.0044	4.634	1275.5	1519.3
0.0044	4.945	1245.4	1487.8
0.0044	6.09	1184.3	1443.7
0.0044	6.655	1153.2	1408.9
0.0044	7.545	1129.2	1395.5
0.0044	8.138	1118.9	1386.5
0.0044	8.986	1100	1373.7
0.0044	9.551	1086.2	1339.9
0.0044	10.71	1070.8	1322.7
0.0044	11.006	1064.4	1318.3
0.0044	12.151	1037.5	1252.3
0.0044	12.744	1037.9	1268.2
0.0044	13.889	1021.9	1227.5
0.0044	14.468	1021.6	1219.2
0.0175	0.129	2471.6001	51.6
0.0175	0.211	2426.1333	154.6
0.0175	0.603	2212.2334	769.9
0.0175	0.941	2053.6001	1099.8
0.0175	1.618	1750.9	1473.3

Appendix II Mudstone Experiment Data

0.0175	1.943	1647.3667	1595
0.0175	2.768	1432.2667	1682
0.0175	3.323	1345.6	1722.3
0.0175	4.717	1186.8334	1722.1
0.0175	5.543	1122.4333	1682.8
0.0175	7.207	1043.5667	1629.2
0.0175	8.046	1019.1334	1614.4
0.0175	9.711	979.6334	1561.9
0.0175	10.821	945.7	1496.7
0.0175	13.595	899.9333	1374.1
0.0175	14.691	884.9333	1329.1
0.0583	0.223	2427.8333	318.1
0.0583	0.526	2285.7666	756.2
0.0583	1.077	2004.9667	1261.4
0.0583	1.301	1904.1666	1427.9
0.0583	1.826	1701.5	1588.8
0.0583	2.063	1635.6333	1633
0.0583	2.575	1505.9667	1720.1
0.0583	2.799	1469.2333	1763.8
0.0583	3.495	1343	1751.1
0.0583	4.034	1281.7	1764.9
0.0583	5.125	1179.7	1744.8
0.0583	5.663	1154.8334	1758.1
0.0583	6.491	1105.1	1696.8
0.0583	6.99	1077.4667	1664.9
0.0583	8.081	1046.4667	1645.4
0.0583	8.62	1035.9333	1635.7
0.0583	9.71	1004.9667	1571.9
0.0583	10.21	1006.9333	1585.3
0.0583	10.762	984.2	1531.8
0.0583	11.3	976.3666	1500.8
0.0583	12.378	953.4667	1439.6
0.0583	12.917	938.5	1409.4
0.0583	13.994	923.2333	1370.8
0.0583	14.533	911	1341.3
0.175	0.235	2386.5667	575.3
0.175	0.496	2260.1667	1038.2
0.175	1.004	1968.8	1467.9
0.175	1.276	1858.0667	1626.5

Appendix II Mudstone Experiment Data

0.175	1.71	1697.2667	1781.3
0.175	1.958	1622.5667	1857.5
0.175	2.417	1505.7333	1902.4
0.175	2.677	1458.3667	1950.8
0.175	3.024	1399.0333	1970.5
0.175	3.123	1374.8	1941.9
0.175	3.829	1272.4667	1927.7
0.175	4.362	1236.1	1969.5
0.175	5.341	1145.1666	1897.4
0.175	5.874	1117.2	1886.7
0.175	6.915	1065.1333	1840.3
0.175	7.361	1043.9	1806
0.175	8.402	1015.1334	1785.7
0.175	8.935	1004.4	1775.4
0.175	9.964	979.6	1730.4
0.175	10.224	983.8	1750.2
0.175	11.191	956.3666	1682.6
0.175	11.699	958.8333	1697.2
0.175	12.765	941.5333	1652.8
0.175	13.198	930.8333	1620.7
0.175	14.264	903.1667	1530.2
0.175	14.797	897.5	1520.7
0.525	0.271	2451.2	153.6
0.525	0.551	2439.1667	612.5
0.525	0.993	2284.8333	1219.6
0.525	1.26	2164.2334	1495
0.525	1.728	1964.0333	1714.9
0.525	1.995	1860.7333	1785.7
0.525	2.557	1709.5333	1925.5
0.525	2.731	1654.6666	1922
0.525	3.573	1507.8667	1979.6
0.525	4.121	1423.5333	1968.4
0.525	4.964	1350.3	1975.5
0.525	5.498	1309.9	1964.4
0.525	6.621	1242.9	1917
0.525	7.17	1214.1333	1882
0.525	8.012	1178.5667	1841.3
0.525	8.493	1175.8667	1855.1
0.525	9.616	1138.1333	1786

Appendix II Mudstone Experiment Data

0.525	10.178	1129.6333	1774.9
0.525	11.034	1106.2667	1712.3
0.525	11.502	1088.1333	1657.9
0.525	12.638	1063.3334	1590.7
0.525	13.2	1056.3667	1569.8
0.525	14.256	1024.0667	1480.4
0.525	14.831	1018.6667	1442
1.75	0.156	2492.3333	113.8
1.75	0.347	2487.5334	369.4
1.75	0.673	2444.5	933.9
1.75	0.85	2362.8	1214.7
1.75	1.408	2088.6001	1685.4
1.75	1.694	1980.5667	1848.5
1.75	2.252	1810.5667	2005.1
1.75	2.538	1741.7333	2054.8
1.75	3.49	1560.4667	2089.7
1.75	3.79	1521.8334	2090.8
1.75	4.919	1401.2667	2058.8
1.75	5.491	1351.5667	2019.5
1.75	6.824	1277	1964.4
1.75	7.396	1249.6	1926
1.75	8.526	1222.1666	1902.5
1.75	9.097	1208.5	1890.6
1.75	10.159	1168.2333	1791.7
1.75	10.744	1147.4	1729.2
1.75	11.9	1126.6666	1681.7
1.75	12.445	1120.7	1671.3
1.75	13.574	1095.6	1625.1
1.75	14.064	1090.0333	1615.9
0.0044	0.481	2265	723.2
0.0044	1.286	1884.9	1296.8
0.0044	1.978	1662.6	1479.5
0.0044	2.784	1481.3	1521.7
0.0044	4.041	1320.1	1528.8
0.0044	5.496	1209.3	1452.8
0.0044	6.951	1150.6	1430.4
0.0044	8.689	1106.3	1378.2
0.0044	10.13	1070.1	1306.2
0.0044	11.586	1059.1	1309.8

Appendix II Mudstone Experiment Data

0.0044	13.309	1029.6	1235.8
0.0175	0	2500	0
0.0175	0.251	2403.5667	206
0.0175	1.28	1881	1299.9
0.0175	2.268	1561.0333	1665.4
0.0175	4.162	1249.7333	1756.9
0.0175	6.382	1078.4333	1667.8
0.0175	8.872	994.1667	1576.4
0.0175	12.472	918.9	1423.8
0.0583	0	2500	0
0.0583	0.762	2162.8	1046.4
0.0583	1.616	1784.1666	1544
0.0583	2.26	1577.3334	1677.7
0.0583	3.022	1418.4333	1735.9
0.0583	4.586	1229.9333	1778.2
0.0583	5.939	1128.0333	1706.8
0.0583	7.555	1074.5	1700.1
0.0583	9.145	1023.1	1626.3
0.0583	10.486	993.1	1558.5
0.0583	11.865	963.4667	1469.6
0.0583	13.442	935.6667	1400.9
0.175	0	2500	0
0.175	0.756	2110.8	1308
0.175	1.524	1765.0667	1757.6
0.175	2.231	1546.8	1879.2
0.175	2.937	1405.3334	1945.6
0.175	3.297	1344.3334	1938.4
0.175	4.908	1183.5667	1932.2
0.175	6.395	1094.1333	1876.3
0.175	7.882	1042.7	1846.5
0.175	9.431	996.2667	1765.7
0.175	10.67	970.2333	1717
0.175	12.232	939.3666	1638.8
0.175	13.731	909.3333	1563.4
0.525	0	2500	0
0.525	0.725	2400.1001	917.1
0.525	1.541	2029.1333	1617.1
0.525	2.29	1776.6	1855.5
0.525	3.025	1605.8334	1966

Appendix II Mudstone Experiment Data

0.525	4.696	1374.0333	1981
0.525	6.073	1263.0333	1904.2
0.525	7.45	1212.7333	1900
0.525	9.068	1151.5333	1796.8
0.525	10.459	1112.4667	1723.4
0.525	12.063	1082.2	1647.3
0.525	13.775	1041.4	1524.9
1.75	0	2500	0
1.75	0.51	2469.7	651.9
1.75	1.136	2219.5	1492.8
1.75	1.966	1891.6	1955.1
1.75	2.823	1680.5667	2076.5
1.75	4.075	1492.1666	2104.3999
1.75	6.049	1311.7667	1980.8
1.75	7.954	1235.9333	1914.4
1.75	9.682	1190.9667	1852.7
1.75	11.302	1130.4667	1693.1
1.75	13.016	1103.9	1635.6
1.75	14.649	1069.3667	1532

APPENDIX III PPV EXPERIMENT DATA

σ_i	σ_c	RMR	PPV damage criterion(slight damage)
0.6	18	20	0.065
0.8	24.5	30	0.065
1	31	40	0.095
1.25	37.5	50	0.14
1.47	44.2	60	0.175
1.71	60	38	0.43
2	70	60	0.49
2	70	95	0.9
2.1	75	60	0.49385
2.2	77	61	0.49769
2.3	79	62	0.50154
2.4	81	63	0.50538
2.5	83	64	0.50923
2.6	85	65	0.51308
2.7	87	66	0.51692
2.8	89	67	0.52077
2.9	91	68	0.52461
3	93	69	0.52846
3.1	95	70	0.53231
3.2	97	71	0.53615
3.3	100	75	0.54
3.4	110	75.6	0.5447
3.5	114	76.2	0.5494
3.6	118	76.8	0.5541
3.7	122	77.4	0.5588
3.8	126	78	0.5635
3.9	130	78.6	0.5682
4	134	79.2	0.5729

Appendix III PPV Experiment Data

4.1	138	79.8	0.5776
4.2	142	80.4	0.5823
4.3	146	81	0.587
4.4	150	81.6	0.5917
4.5	154	82.2	0.5964
4.6	158	82.8	0.6011
4.7	162	83.4	0.6058
4.8	166	84	0.6105
4.9	170	84.6	0.6152
5	174	85.2	0.62
5.1	180	85.8	0.6343
5.2	183	86.4	0.6486
5.3	186	87	0.6629
5.4	189	87.6	0.6772
5.5	192	88.2	0.6915
5.6	195	88.8	0.7058
5.7	200	95	0.72
6	180	95	1
17.45	150	90	0.9

APPENDIX IV MATLAB PROGRAM

```
% Title: An Ensemble Neural Network (ENN)

% This program is a universal ensemble neural network program, thus it
% can be used to solve many kinds of problems. In this example, it is applied
% to regress the peak function. In this program, it can list the results
% of the single networks, the simple average combined ENN and AIC combined
% ENN. All these three NNs are used cross validation data set to avoid
% over-fitting. For AIC-ENN, the approach scaled delta-AIC to the weight,
% and the beta value is determined by the diversity of the component
% networks outputs.

% This program was developed by Miss Zhang Yun (zh0002un@ntu.edu.sg),
% supervised by Assoc Prof. Zhao Zhi Ye (Nanyang Technological University).

clc                                % clear the command window
clear;
rand('seed', 10);
randn('seed', 10);

ensemblesize=3; % ensemblesize is the number of the component networks.
hiddenminno = 11;
a=2; %a is the difference of the hidden nodes between two component NNs.
setsize=4;    % setsize is the number of run times to find the best net
% structure of the component network.
maxepoch=100;
runno=20;

fprintf('ENN with cross validation    setsize = %g        epoch = %g\n', setsize, maxepoch);
for run=1:1:runno

% x,y are training data %
```

Appendix IV Matlab Program

```

x=-3:0.6:3;
y=x;
[X,Y]=meshgrid(x,y);
X1=reshape(X,1,121);
Y1=reshape(Y,1,121);

%x0,y0 are test data%
x0=-2.7:0.6:3;
y0=x0;
[X0,Y0]=meshgrid(x0,y0);
X10=reshape(X0,1,100);
Y10=reshape(Y0,1,100);

%x2,y2 are cross validation data%
x2=-2.85:0.6:3;
y2=x2;
[X2,Y2]=meshgrid(x2,y2);
X20=reshape(X2,1,100);
Y20=reshape(Y2,1,100);

INDATA=[X1;Y1];
OUTDATA = 3*(1-X1).^2.*exp(-(X1.^2) - (Y1+1).^2) ...
          - 10*(X1/5 - X1.^3 - Y1.^5).*exp(-X1.^2-Y1.^2) ...
          - 1/3*exp(-(X1+1).^2 - Y1.^2)+(0.05^0.5)*randn(size(X1));
P=INDATA;
T=OUTDATA;

TDATA=[X10;Y10];
OTDATA = 3*(1-X10).^2.*exp(-(X10.^2) - (Y10+1).^2) ...
          - 10*(X10/5 - X10.^3 - Y10.^5).*exp(-X10.^2-Y10.^2) ...
          - 1/3*exp(-(X10+1).^2 - Y10.^2)+(0.05^0.5)*randn(size(X10));
te.P=TDATA;
te.T=OTDATA;

CVDATA=[X20;Y20];
OCVDATA = 3*(1-X20).^2.*exp(-(X20.^2) - (Y20+1).^2) ...
          - 10*(X20/5 - X20.^3 - Y20.^5).*exp(-X20.^2-Y20.^2) ...
          - 1/3*exp(-(X20+1).^2 - Y20.^2)+(0.05^0.5)*randn(size(X20));
v.P=CVDATA;
v.T=OCVDATA;

traininput=P;
traintarget=T;

```


Appendix IV Matlab Program

```

testinput=TDATA;
testtarget=OTDATA;
vinput=CVDATA;
vtarget=OCVDATA;

[n,testexpno] = size(testinput);           % 'testexpno' is the
number of test examples, 'n' is useless
[attrno,trainexpno] = size(traininput);    % 'attrno' is the number
of attributes, 'trainexpno' is the number of training examples

% generate the component neural networks
no=0;
nono=zeros(setsize, ensemblesize);
best=zeros(1,ensemblesize);
k=zeros(1,ensemblesize);
hiddenno=zeros(1,ensemblesize);
mse_trw=zeros(1,ensemblesize);
aic=zeros(1,ensemblesize);
for i = 1:ensemblesize
    mse_tre=zeros(1,setattr);
    mse_best=zeros(1,1);
    mse_com=zeros(1,1);
    for e = 1:setsize
        % generate the component training sets
        compinput = zeros(attrno,trainexpno);
        comptarget = zeros(1,trainexpno);
        for j = 1:trainexpno
            % appear = floor(rand * trainexpno) + 1;    % 'appear'
            indicates which example should appear in current component training set
            compinput(:,j) = traininput(:,j);
            comptarget(:,j) = traintarget(:,j);
        end

        % train the component neural networks
        hiddenno(1,i)=hiddenminno+a*(i-1);
        k(1,i)=hiddenno(1,i)*(2+1+1)+2 ; %k is total number of the
parameters include weights and bias
        net = newff(MinMax(compinput), [(hiddenno(1,i)) 1], {'tansig'
'purelin'});
        net.trainParam.epochs = maxepoch;
        net.trainParam.goal = 0.0;
        net.trainParam.mc=0.7;
        net.trainParam.lr=0.05;
    end
end

```

Appendix IV Matlab Program

```

    net.trainParam.show=NaN;
%   net = train(net,compinput,comptarget);
[net, tr] = train(net,compinput,comptarget,[],[],v);
figure(run);
plot(tr.epoch,tr.perf,tr.epoch,tr.vperf)
legend('Training','Validation',-1);
ylabel('Squared Error %d'); xlabel('Epoch')

% save the component neural networks
no=no+1;
netfile = strcat('net',dec2base(no,10));
save(netfile,'net');
nono(e,i)=no;
outpute = sim(net,compinput);
mse_tre(1,e) = mse(outpute - comptarget);
end
best(1,i)=no-setsizes+1;
mse_best=mse_tre(1,1);
for e=2:setsizes
    mse_com=mse_tre(1,e);
    if mse_best>mse_com
        mse_best=mse_com;
        best(1,i)=nono(e,i);
    end
end
mse_trw(1,i)=mse_best;
if trainexpno/k(1,i)<40
    aic(1,i)= trainexpno*log(mse_trw(1,i))+
2*k(1,i)+2*k(1,i)*(k(1,i)+1)/(trainexpno-k(1,i)-1);
else
    aic(1,i)= trainexpno*log(mse_trw(1,i))+2*k(1,i);
end
end

% finding the weight of component network using AIC
aic_value=aic;
weight=zeros(1,ensemblesize);
delta_aic=zeros(1,ensemblesize);
t=zeros;
q=1;
m=zeros;
c1=zeros;
beta1=zeros;

```

Appendix IV Matlab Program

```

aic_new=zeros(1,ensemblesize);
mm=zeros;
mim=zeros;
for i=1:ensemblesize
    q=q*aic(1,i);
    if q<0
        t=t+1;
    end
end
m=min(aic);
for i = 1: ensemblesize
    delta_aic(1,i)=aic(1,i)-m;
end
mm=max(delta_aic(1,:));
mim=min(delta_aic(1,:));
[m,trexpno] = size(traininput);
nnoutput = zeros(ensemblesize,trexpno);
nndiv = zeros(1,ensemblesize);
traindiv = zeros(1,trexpno);
for i = 1:ensemblesize
    netfile = strcat('net',dec2base(best(1,i),10));
    load(netfile);
    nnoutput(i,:) = sim(net,traininput);
end
traindiv=mean(nnoutput);
for i = 1:ensemblesize
    nndiv(1,i)= mse(nnoutput(i,:) - traindiv) ;
end

beta1=max(nndiv)/min(nndiv);
for i = 1: ensemblesize
    aic_new(1,i)=1+(delta_aic(1,i)-mim)/(mm-mim)*beta1 ;
    % aic_new(1,i)=1+delta_aic(1,i)/mm*beta1 ;
end
for i = 1: ensemblesize
    c1= c1+(1/aic_new(1,i));
end
for i = 1: ensemblesize
    weight(1,i)= (1/aic_new(1,i))/c1;
end

% test the ensemble
ensize=ensemblesize;

```

Appendix IV Matlab Program

```

[n,testexpno] = size(testinput);    % 'testexpno' is the number of
test examples, 'n' is useless
[m,trexpno] = size(traininput);
enoutput_tr=zeros(1,trexpno); % training output of ENN by AIC method
enoutput_te=zeros(1,testexpno); % test output of ENN by AIC method
enoutput_trs=zeros(1,trexpno);    % training output of ENN by
simple-averaging method
enoutput_tes=zeros(1,testexpno); % test output of ENN by
simple-averaging method
mse_tesgl=zeros(1,ensize);    % training output MSE of single NN
mse_trsgl=zeros(1,ensize);    % test output MSE of single NN
for i = 1:ensize
    netfile = strcat('net',dec2base(best(1,i),10));
    load(netfile);
    output_te = sim(net,testinput);
    enoutput_te(1,:) = enoutput_te(1,:) + weight(1,i) * output_te;
    mse_tesgl(1,i)=mse(output_te-testtarget);
    sigltest(i,run)=mse_tesgl(1,i);
    enoutput_tes = enoutput_tes + output_te;
end
for i = 1:ensize
    netfile = strcat('net',dec2base(best(1,i),10));
    load(netfile);
    output_tr = sim(net,traininput);
    enoutput_tr(1,:)=enoutput_tr(1,:) + weight(1,i) * output_tr;
    mse_trsgl(1,i)=mse(output_tr-traintarget);
    sigltrain(i,run)=mse_trsgl(1,i);
    enoutput_trs = enoutput_trs + output_tr;
end
enoutput_trs = enoutput_trs / sum(ensize) ;
enoutput_tes = enoutput_tes / sum(ensize) ;
% %    figure(run);
% %    [m,b,r1]=postreg(enoutput_te(1,:),testtarget);
mse_te = mse(enoutput_te(1,:) - testtarget) ;
mse_tr = mse(enoutput_tr(1,:) - traintarget) ;
mse_tes = mse(enoutput_tes - testtarget) ;
mse_trs = mse(enoutput_trs - traintarget) ;
entest(1,run)=mse_te;
simtest (1,run)=mse_tes;
entrain(1,run)=mse_tr;
simtrain(1,run)=mse_trs;
fprintf('AIC run = %g           mse_te = %12.5g           mse_tr
=%12.5g           ', run, mse_te, mse_tr );    fprintf('\n');
```

Appendix IV Matlab Program

```

fprintf('AIC run = %g', run);
fprintf('    weight value    = ');
fprintf(1, ' %12.4g', weight(1,:));    fprintf('\n');
end
fprintf('\n\n');
for i = 1:ensize
    fprintf('hidden nodes=%g    min test sgl=    %g\n',hiddenno(1,i),min(sigltest(i,:)));    fprintf('\n');
    fprintf('hidden nodes=%g    mean test sgl=    %g\n',hiddenno(1,i),mean(sigltest(i,:)));    fprintf('\n');
    fprintf('hidden nodes=%g    SD test sgl=    %g\n',hiddenno(1,i),std(sigltest(i,:)));    fprintf('\n\n');
    fprintf('hidden nodes=%g    min train sgl=    %g\n',hiddenno(1,i),min(sigltrain(i,:)));    fprintf('\n');
    fprintf('hidden nodes=%g    mean train sgl=    %g\n',hiddenno(1,i),mean(sigltrain(i,:)));    fprintf('\n');
    fprintf('hidden nodes=%g    SD train sgl=    %g\n',hiddenno(1,i),std(sigltrain(i,:)));    fprintf('\n\n');
end
fprintf(' min test sim=    %g    ',min(simtest));    fprintf('\n');
fprintf(' mean test sim=    %g    ',mean(simtest));    fprintf('\n');
fprintf('    SD test sim=    %g    ',std(simtest));    fprintf('\n\n');
fprintf(' min train sim=    %g    ',min(simtrain));    fprintf('\n');
fprintf(' mean train sim=    %g    ',mean(simtrain));    fprintf('\n');
fprintf('    SD train sim=    %g    ',std(simtrain));    fprintf('\n\n');
fprintf(' min test AIC =    %g    ',min(entest));    fprintf('\n');
fprintf('mean test AIC =    %g    ',mean(entest));    fprintf('\n');
fprintf('    SD test AIC =    %g    ',std(entest));    fprintf('\n\n');
fprintf('min train AIC =    %g    ',min(entrain));    fprintf('\n');
fprintf('mean train AIC =    %g    ',mean(entrain));    fprintf('\n');
fprintf('    SD train AIC =    %g    ',std(entrain));    fprintf('\n\n');
% end of function

```