

https://dspace.org/

The use of genetic algorithms and neural networks to approximate missing data in database

Abdella, Mussa Ismael

2006-01-16T07:52:17Z
http://hdl.handle.net/10539/105

Downloaded from a DSpace repository. https://dspace.org/

THE USE OF GENETIC ALGORITHMS AND NEURAL NETWORKS TO APPROXIMATE MISSING DATA IN DATABASE

MUSSA ISMAEL ABDELLA

A RESEARCH REPORT SUBMITTED TO THE FACULTY OF ENGINEERING AND BUILT ENVIRONMENT, UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ENGINEERING BY COURSEWORK AND RESEARCH REPORT

JOHANNESBURG, JANUARY 2005

Declaration

I declare that this research report is my own, unaided work. It is being submitted for the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

_____

Mussa Ismael Abdella

This _____ day of _____ 2005

Abstract

Missing data creates various problems in analysing and processing of data in databases. Due to this reason missing data has been an area of research in various disciplines for a quite long time. This report introduces a new method aimed at approximating missing data in a database using a combination of genetic algorithms and neural networks. The proposed method uses genetic algorithm to minimise an error function derived from an auto-associative neural network. The error function is expressed as the square of the difference between the actual observations and predicted values from an auto-associative neural network. In the event of missing data, all the values of the actual observations are not known hence, the error function is decomposed to depend on the known and unknown (missing) values. Multi Layer Perceptron (MLP), and Radial Basis Function (RBF) neural networks are employed to train the neural networks. The research focus also lies on the investigation of using the proposed method in approximating missing data with great accuracy as the number of missing cases within a single record increases. The research also investigates the impact of using different neural network architecture in training the neural network and the approximation

found to the missing values. It is observed that approximations of missing data obtained using the proposed model to be highly accurate with 95% correlation coefficient between the actual missing values and corresponding approximated values using the proposed model. It is found that results obtained using RBF are better than MLP. Results found using the combination of both MLP and RBF are found to be better than those obtained using either MLP or RBF. It is also observed that there is no significant reduction in accuracy of results as the number of missing cases in a single record increases. Approximations found for missing data are also found to depend on the particular neural network architecture employed in training the data set.

Acknowledgements

First, I would like to thank my supervisor Prof.Tshilidzi Marwala for his constant guidance, assistance, support, and advice throughout the course of this research. Without his constant support and guidance this report would not have been in its present structure. Thank you for your immense devotion as a supervisor, which has been instrumental to the success of this research. The original idea presented in this research is Marwala's idea and the investigation carried out here is one of his suggestions.

I would also like to thank Dr.Ravindran Ramalingam for his assistance in proof reading the document. Many thanks also to all my other friends for providing moral support and wishing me good luck.

I wish to thank my parents for their endless understanding, patience, encouragement, and support.

Last but not least, thanks to the ALMIGHTY for amply supplying me with strength, peace, and joy in my life.

Contents

Declaration **i**

Abstract **ii**

Acknowledgements **iv**

List of Figures **x**

List of Tables **xi**

List of Abbreviations **xii**

1 Introduction **1**

1.1 Statement of the problem 1

1.2 Missing data 2

1.3 Importance of the research 3

1.4 Overview of approach 4

1.5 Overview of results 5

1.6 Structure of report 6

2 Artificial Neural Networks **8**

2.1 Simple neuron 10

2.2 Learning rules 12
2.2.1 Supervised learning 12
2.2.2 Unsupervised learning 13
2.3 Architecture of neural networks 14
2.3.1 Single layer perceptrons 15
2.3.2 Multi layer perceptrons 17
2.3.3 Radial basis function 21
2.4 Chapter summary 23

3 Genetic Algorithms **24**
3.1 Why genetic algorithms? 25
3.2 Structure of genetic algorithm 26
3.3 Optimization of a simple function 29
3.3.1 Representation of the problem 29
3.3.2 Evaluation 30
3.3.3 Crossover 32
3.3.4 Mutation 34
3.3.5 Decoding 36
3.4 Chapter summary 37

4 Missing Data **38**
4.1 Missing data mechanisms 39
4.1.1 Missing completely at random 39
4.1.2 Missing at random 40
4.1.3 Non ignorable case 41
4.2 Patterns of missing data 41

4.3 Missing data imputation algorithms 43
4.3.1 Listwise or casewise data deletion 43
4.3.2 Pairwise data deletion 44
4.3.3 Mean substitution 44
4.3.4 Hot deck imputation 45
4.3.5 Regression methods 46
4.3.6 Expectation maximization 47
4.3.7 Raw maximum likelihood methods 49
4.3.8 Multiple imputation 51
4.4 Chapter summary 54

5 Research Method **55**
5.1 Introduction 55
5.2 Research question and statement of hypothesis 56
5.3 Hypotheses testing 57
5.4 Proposed method 59
5.5 Selecting best neural network architecture 61
5.5.1 MLP architecture used in the research 62
5.5.2 RBF architecture used in the research 62
5.6 Experimental data 63
5.7 Genetic algorithm implementation 64
5.8 Approach followed 65
5.9 Data analysis 66
5.10 Chapter summary 67

6 Results and Discussion **69**

6.1 Introduction 69
6.2 Training of neural network 70
6.3 Performance of genetic algorithm 72
6.4 Missing data analysis 73
6.5 Results and hypotheses testing 80
6.6 Comparisons of results with other methods 81
6.7 Chapter summary 82

7 Conclusions and Further Research 83

7.1 Conclusions 83
7.1.1 Summary of research 83
7.1.2 Summary of results and conclusions 84
7.2 Future work 85
7.2.1 Using different machine learning techniques 85
7.2.2 Using different optimization methods 86
7.2.3 Comparison with other models under the same data set 86
7.2.4 Forecasting and risk analysis 87

References 92

Appendix 1 xciii

List of Figures

2.1 Single input neuron 11

2.2 Multiple input neuron 12

2.3 Architecture of single layer perceptron 16

2.4 MLP architecture 18

2.5 RBF architecture [Haykin 1999] 22

3.1 Genetic algorithm approach [Michalewicz 1996] 28

3.2 Genetic algorithm overview [Forrest 1996] 37

5.1 Schematic representation of proposed model 61

5.2 MLP architecture used in the research 63

5.3 RBF architecture used in the research 64

6.1 Data vs trained using MLP 71

6.2 Data vs trained using RBF 71

6.3 Performance of the genetic algorithm during the run 72

6.4 Correlation coefficient MLP, RBF, and MLP+RBF 75

6.5 Standard error MLP, RBF, and MLP+RBF 75

6.6 One missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF 76

6.7 Two missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF 76

6.8 Three missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF 77

6.9 Four missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF 77

6.10 Five missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF 78

List of Tables

1.1 Table with missing values 2

4.1 Table with missing entries 40

4.2 Arbitrary missing data pattern 42

4.3 Monotone missing data pattern 42

5.1 Statistical summary of input data 65

6.1 Correlation coefficient 74

6.2 Standard error 74

6.3 Actual and approximated values using MLP 78

6.4 Actual and approximated values using RBF 79

6.5 Actual and approximated values using MLP+RBF 79

7.1 Proposed model on forecasting and risk analysis applications 87

List of Abbreviations

ANN	Artificial Neural Networks
EM	Expectation Maximization
FIML	Full Information Maximum Likelihood
GAs	Genetic Algorithms
GLMs	Gneralised Linear Models
MAR	Missing At Random (MAR)
MCAR	Missing Completely At Random
MLP	Multi Layer Perceptron
MI	Multiple Imputation
r	Correlation coefficient
RBF	Radial Basis Function
SCG	Scaled Conjugate Gradient
SAB	South African Breweries
Se	Standard error
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Statement of the problem

Inferences made from available data for a certain application depends on the completeness and quality of the data being used in the analysis. Thus, inferences made from a complete data are most likely to be more accurate than those made from incomplete data.

Missing data in a database may arise due to various reasons. To mention some it can arise due to data entry errors, respondents non-response to some items on data collection process, etc. In Table 1.1 we have a database consisting of five variables namely, age, gender, race, income, and educational level, where the values for some variables are missing. Assume Table 1.1 consists of many records of the five variables. But some of the variables in some records are not available. For instance, in Table 1.1 the income and race for the second and fourth records respectively

Table 1.1: Table with missing values

Age	**Gender**	**Race**	**Income**	**Educational level**
25	Male	Black	5000	B.Sc
33	Female	White	?	M.Sc
45	Female	Black	1500	Ph.D
15	Male	?	3000	Diploma

are not available. The question that arises in this case is how do we know the income for the second record? Similarly how do we know the race for the fourth entry? Are there any mechanisms to predict or approximate the missing data depending on the interrelationships that exist between the variables in the database?

The aim of this research is to approximate missing data in databases using a technique which employs combination of genetic algorithm and neural networks. The proposed method uses the interrelationships between the variables in the database to approximate the missing values.

1.2 Missing data

Missing data creates various problems in many applications which depend on good access to accurate data. Hence, methods to handle missing data have been an area of research in statistics, mathematics, and other various disciplines [Yuan 2000; Allison 2000; Rubin 1978]. The reasonable way to handel missing data depends upon how data points become missing. According to Little and Rubin [1987] there are three types of missing

data mechanisms. They are Missing Completely at Random (MCAR), Missing at Random (MAR) and non-ignorable. MCAR situation arises if the probability of missing value for variable X is unrelated to the value X itself or to any other variable in the data set. This refers to data where the absence of data does not depend on the variable of interest or any other variable in the data set [Rubin 1978]. MAR arises if the probability of missing data on a particular variable X depends on other variables, but not on X itself and the non-ignorable case arises if the probability of missing data X is related to the value of X itself even if we control the other variables in the analysis [Allison 2000]. Depending on the mechanism of missing data, currently various methods are being used to treat missing data. The various methods used in treatment of missing data are presented in Chapter 4.

1.3 Importance of the research

As pointed out previously the presence of missing data in databases creates various problems hence, one of the obvious uses of the research is that it directly provides a new and efficient method for alleviating the problem by approximating the missing values in the database. Other important contributions of the research are:

- Introduces a new research direction into the literature of missing data analysis through neural networks and evolutionary computing

angle.

- As complete and accurate data are the basis for good decisions, the research will have crucial contribution to the process of decision making process.
- Highlights the application of the proposed algorithm in this research to other problems.
- Apart from applications in databases, there are time critical applications which require us to estimate or approximate the values of some missing variables that have to be supplied in relation to the values of other corresponding variables in some systems. Such situations may arise in a system which uses a number of instruments and in some cases one or more of the sensors used to monitor the instruments fail. In such situation the value of the sensor have to be estimated within short time and with great precision, and by taking into account the values of other sensors in the system. The proposed method in this research can be used to approximate the values of the missing sensors in the system.

1.4 Overview of approach

The approach used in this research employs a genetic algorithm to minimise an error function derived from an auto-associative neural network. The error function is expressed as the square of the difference between

the actual observations and predicted values from an auto-associative neural network. In the case of missing data, all the values of the actual observations are not known, hence, the error function is decomposed to depend on the known and unknown (missing) values. Multi-Layer Perceptron (MLP), and Radial Basis Function (RBF) neural networks are employed to train the neural networks.

To evaluate the accuracy of the approximated missing values using the model a complete database was used in the experiment. Entries from the database were removed and approximated using the model. A neural network toolbox by Nabney [2001] and genetic algorithm toolbox by Houck *et al.* [1995] were used to implement the proposed model. Correlation coefficients and standard error of approximated values were used to evaluate the accuracy and validity of approximated values using the model.

1.5 Overview of results

The results of the research has revealed a high correlation coefficient between approximated values and actual missing data removed from the database. This means that the proposed method's approximation to missing values to be highly accurate. It is found that the specific architecture used in training the data set have a significant impact on the approximations. There was no significant reduction in accuracy observed

as the number of missing cases in single record gets bigger. Results found using both MLP and RBF combined are better than either MLP or RBF individually. Results found using RBF are relatively better than MLP.

1.6 Structure of report

In the following chapter, background information regarding artificial neural networks is presented. Definition of neural networks and different neural network architecture are described. Chapter 3 begins by introducing genetic algorithms and their applications in finding solutions to difficult real world optimization problems. The significance of the literature in relation to the research is also highlighted.

Chapter 4 describes missing data, different missing data mechanisms and various missing data imputation algorithms. Chapter 5 begins by giving the research question that the research is attempting to answer in this research. This leads to the formal hypothesis detailing the results expected in the research regarding the approximated values in missing databases. Detailed methodology used in the research process is outlined at the end of the chapter.

Chapter 5 presents results obtained from the experiments and corresponding discussion on the results. Hypotheses testing is done based on the results observed. Finally, Chapter 6 presents conclusion, and possible future research. Applications of the proposed algorithm in finding

new solutions to other problems is also given at the end of the chapter.

Chapter 2

Artificial Neural Networks

A neural network is an information processing paradigm that is inspired by the way biological nervous systems, like the brain process information [Yoon and Peterson 1990; Haykin 1999]. It is a machine that is designed to model the way in which the brain performs a particular task or function of interest [Haykin 1999].

A neural network consists of four main parts [Haykin 1999]. These are the processing units u_j, where each u_j has a certain activation level $a_j(t)$ at any point in time, weighted interconnections between the various processing units which determine how the activation of one unit leads to input for another unit, an activation rule which acts on the set of input signals at a unit to produce a new output signal, and a learning rule that specifies how to adjust the weights for a given input/output pair. One of the main important features of a neural networks is its ability to adapt to new environment. Hence, learning algorithms are critical to neural

networks.

Due to their ability to derive meaning from complicated data, neural networks are used to extract patterns and detect trends that are too complex to be noticed by many other computer techniques [Hassoun 1995]. A trained neural network can be considered as an expert in the category of information it has been given to analyse [Yoon and Peterson 1990]. This expert can then be used to provide predictions given new situations. Because of their ability to adapt to a non-linear data neural networks are also being used to model various non-linear applications [Haykin 1999; Hassoun 1995].

Neural networks have many advantages as machine learning technique and some of them are outlined as follows.

- **Adaptive learning:** An ability to learn how to do tasks based on the data given for training or initial experience [Haykin 1999].
- **Nonlinearity:** An artificial neuron can be linear or nonlinear. A neural network made up of an interconnection of nonlinear neurons is nonlinear. Haykin [1999] points that, nonlinearity is a highly important property, especially if the underlying physical mechanism responsible for generation of the input signal is inherently nonlinear.
- **Adaptivity:** Neural Networks have a built in capability to adapt their synaptic weights to changes in the surrounding environment. In particular, a neural network trained to operate in a specific envi-

ronment can easily be retrained to deal with minor changes in the operating environmental conditions [Haykin 1999].

- **Fault Tolerance:** Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage [Haykin 1999].

2.1 Simple neuron

A neuron is "an information-processing unit that is fundamental to the operation of a neural network"[Haykin 1999, p.10]. In neural networks we can have a single or multiple input neurons as illustrated in Figure 2.1 and 2.2.

Figure 2.1 shows a single input neuron. The scalar input p is multiplied by the scalar weight w to form wp, one of the terms that is sent to the summer. The other input 1 is multiplied by a bias b and then passed to the summer. The summer output n often referred to as the net input, goes into an activation function f which produces the scalar neuron output a.

Figure 2.2 shows a multiple input neuron with R inputs. The individual inputs are each weighted by corresponding elements $w_{11}, w_{12}, ...w_{1R}$, of the weight matrix W. The neuron has a bias b, which is summed with

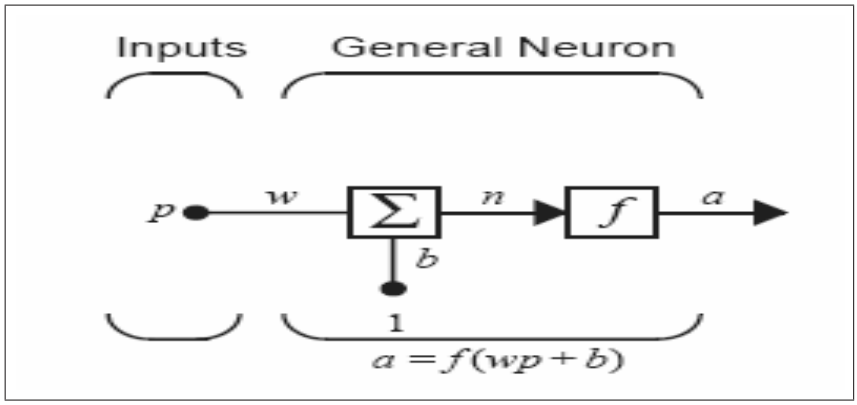


Figure 2.1: Single input neuron

the weighted inputs to form the net input n:

$$n = w_{11}p_1 + w_{12}p_2 + ... + w_{1R}p_R + b \tag{2.1}$$

Equation (2.1) can be written in matrix form as:

$$n = WP + b \tag{2.2}$$

Where the matrix W for the single neuron case has only one row. Now the neuron output can be written as

$$a = f(WP + b) \tag{2.3}$$

The bias b is an external parameter of the neuron. The activation function f in Figure 2.1 and 2.2 defines the output of a neuron in terms of the result from n. According to Haykin [1999] there are three basic types of activation functions. These are the threshold functions, piecewise linear function and sigmoid function.

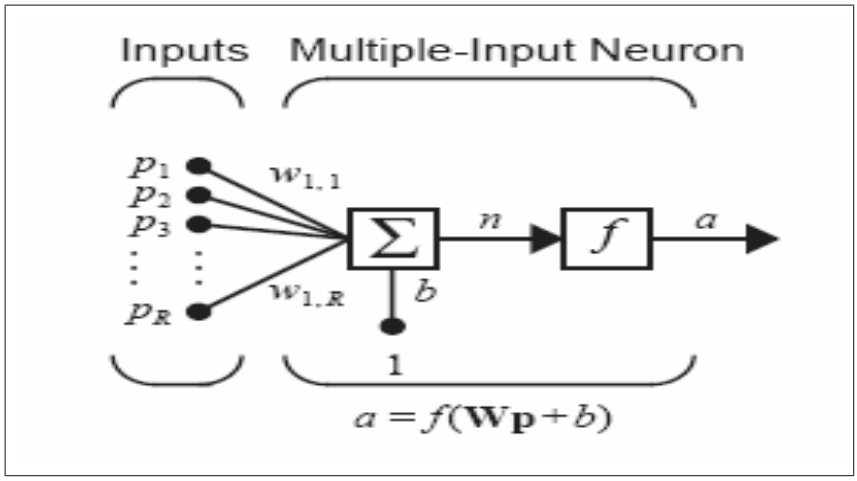


Figure 2.2: Multiple input neuron

2.2 Learning rules

One of the most significant attributes of a neural network is its ability to learn by interacting with its environment or with an information source [Hassoun 1995; Haykin 1999]. Therefore learning algorithm are central to neural networks. Depending on the existence or absence of a target value in the training process, learning rules in neural networks are mainly classified into supervised and unsupervised learning.

2.2.1 Supervised learning

Supervised learning which is also called learning with a teacher is learning which involves a target values for the network outputs [Haykin 1999; Bishop 1995]. For each input pattern the value of the desired output is specified.

Supervised learning is based on the system trying to predict outcomes

for known examples and is a commonly used training method [Freeman and Skapura 1991]. It compares its predictions to the target answer and adjusts the weights accordingly. The data starts as an input to the input layer neurons. The neurons pass the inputs along to the next nodes.

In a supervised learning system, the predicted output is compared to the actual or target output. If the predicted output is equal to the actual output, no change is made to the weights in the network. However, if the predicted output is higher or lower than the actual or target value, the error is propagated back through the system and the weights are adjusted accordingly.

The process of feeding errors backwards through the network is called back propagation [Nabney 2001]. Both the Multi-Layer Perceptron and the Radial Basis Function are supervised learning techniques. The Multi-Layer Perceptron uses the back-propagation while the Radial Basis Function uses a feed-forward approach which trains on a single pass [Nabney 2001].

2.2.2 Unsupervised learning

The other form of learning is unsupervised learning or learning without teacher as it is called. Unsupervised learning does not involve the use of target data. Instead of learning an input-output mapping, the aim is to discover or model the probability of input data [Bishop 1995]. Neural

networks which use unsupervised learning are most effective for clustering or detection of similarities on unlabeled patterns of a given training [Hassoun 1995; Kolarik and Rudorfer 1994]. In unsupervised learning the idea is to optimize some criterion or performance function defined in terms of the output activity of the units in the network. The main use of unsupervised neural networks are in cluster analysis where the goal is to group or classify in to groups. The advantage of the neural network for this type of analysis is that it requires no initial assumptions about what constitutes a group or how many groups there are [Hassoun 1995].

2.3 Architecture of neural networks

The arrangement of neural processing units and their interconnections in neural network can have a profound impact on the processing capabilities of a network [Haykin 1999]. Hence, there are many different connections of how the data flows between the input, hidden and output layers. Neural networks have some set of processing units that receive inputs from the outside, known as the input units. Most neural networks also have one or more layers of hidden layer that receive inputs only from other layers. A layer of processing unit receives a vector of data or the outputs of a previous layer [Freeman and Skapura 1991]. The set of processing units that represent the final result of the neural networks are known as output units. The following sections present single layer perceptron and proceed to the two neural network architecture (MLP and RBF) which

are used in the research.

2.3.1 Single layer perceptrons

A single layer perceptron network consists of a single layer of output nodes. The inputs are fed directly to the outputs via a series of weights. In this way it can be considered as the simplest kind of feed-forward network. Single layer networks implement the well known statistical techniques of regression and Generalised Linear Models (GLMs) [Nabney 2001].

Figure 2.3 depicts the architecture of a single layer perceptron. There is a direct connection between the inputs and outputs. If we denote the input values to the network by x_i where $i = 1, ..., d$. The output a_j associated with each output unit can be represented [Nabney 2001]:

$$a_j = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + b_j^{(1)} \qquad j = 1, ..., c \tag{2.4}$$

Where $w_{ji}^{(1)}$ represents the elements of the weight matrix, $b_j^{(1)}$ are the bias parameters, and c the number of outputs. The variables a_j are then transformed by the activation functions of the output layer to give the output values y_j. The three different activation functions commonly used are the linear, logistic sigmoidal and the softmax activation functions [Nabney 2001]. The linear activation function is appropriate in regression

problems. It is given by

$$y_j = a_j \tag{2.5}$$

The sigmoidal activation function which is commonly used for classification problems is given by

$$y_j = \frac{1}{1 + exp(-a_j)} \tag{2.6}$$

And finally the softmax is

$$y_j = \frac{exp(a_j)}{\sum_{j'} exp(a_{j'})} \tag{2.7}$$

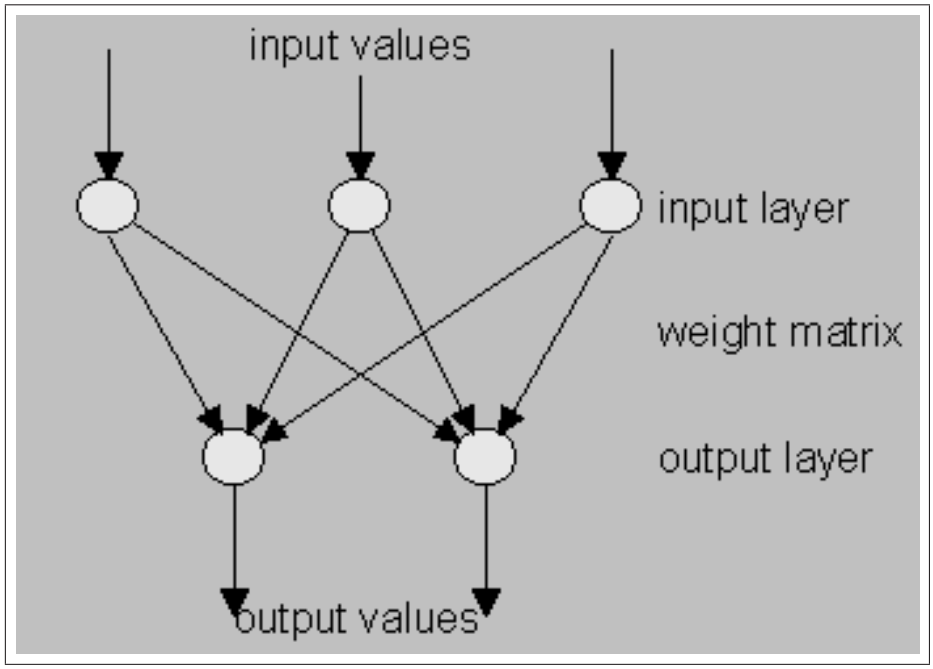


Figure 2.3: Architecture of single layer perceptron

2.3.2 Multi layer perceptrons

Multi Layer Perceptrons (MLP) neural networks consist of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer is directly connected to the neurons of the subsequent layer. In most cases the network consist of two layers of adaptive weights with full connectivity between inputs and hidden units, and between hidden units and outputs [Nabney 2001]. In many applications the units of these networks apply a sigmoid function as an activation function [Bishop 1995].

Figure 2.4 shows the architecture of an MLP network, with three input units, a hidden layer with two neurons and three output units. Analogous to single layer perceptron if x_i denotes input values to the network, where $i = 1, ..., d$. The first layer of the network forms M linear combinations of these inputs to give a set of intermediate activation variables $a_j^{(1)}$ with one variable $a_j^{(1)}$ associated with each hidden unit.

$$a_j = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + b_j^{(1)} \qquad j = 1, ..., M \tag{2.8}$$

In equation (2.8) $w_{ji}^{(1)}$ represents the elements of the first-layer weight matrix and $b_j^{(1)}$ are the bias parameters associated with the hidden units.

The variables of $a_j^{(1)}$ are then transformed by the non-linear activation functions of the hidden layer. If hyperbolic tangent function (tanh) is taken as an activation function, the outputs of the hidden units are then

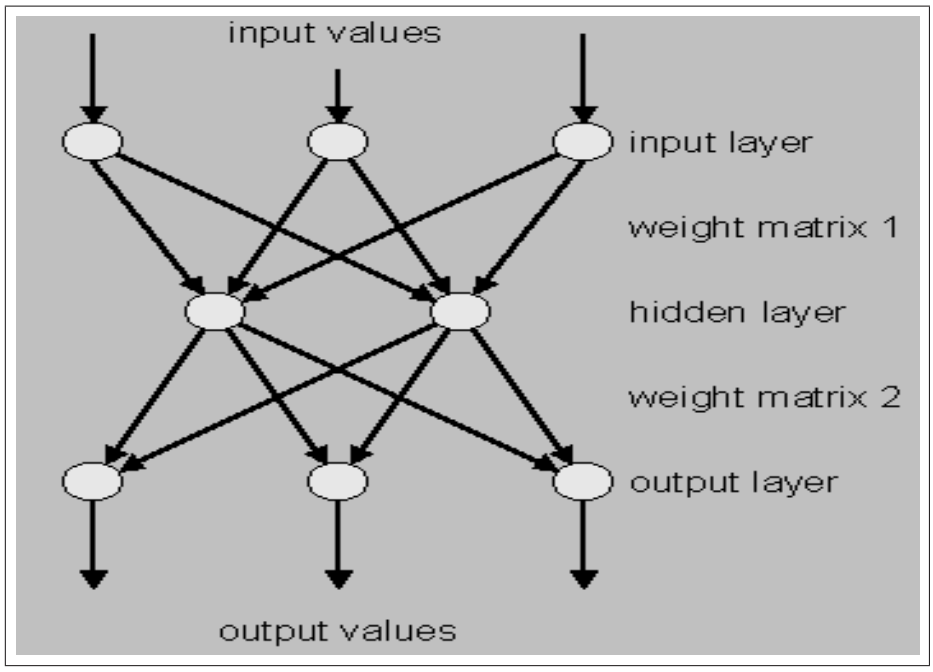


Figure 2.4: MLP architecture

given by

$$z_j^{(1)} = tanh(a_j^{(1)}) \qquad j = 1, ..., M \tag{2.9}$$

The z_i are then transformed by the second layer of weights and biases to give second-layer activation values $a_k^{(2)}$

$$a_k^{(2)} = \sum_{i=1}^{M} w_{kj}^{(2)} z_j + b_k^{(2)} \qquad k = 1, ..., c \tag{2.10}$$

c in the above equation denotes the number of outputs. At the final stage the values are passed through the output unit activation function to give the output values y_k where $k = 1, ..., c$. There are three activation functions which are commonly used [Nabney 2001]. For regression problems

the linear activation function is used

$$y_k = a_k^{(2)} \tag{2.11}$$

For classification problems involving multiple independent attributes the logistic sigmoidal activation function is applied

$$y_k = \frac{1}{1 + exp(-a_k^{(2)})} \tag{2.12}$$

Finally for more usual kind of classification problems in which there are a set of c mutually exclusive classes, the softmax activation function given below is applied

$$y_k = \frac{exp(a_k^{(2)})}{\sum_{k'} exp(a_{k'}^{(2)})} \tag{2.13}$$

Back-propagation: MLP networks apply different learning techniques, the most popular being back-propagation [Bishop 1995]. In back-propagation the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques the error is then fed-back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error-function by a small amount. After repeating this process for a sufficiently large number of training cycles the network converges

to some state where the error of the calculations is small. In this state the network is said to have learned a certain target function [Bishop 1995; Haykin 1999]. This algorithm provides a computationally efficient method for the training of multi-layer perceptron [Haykin 1999]. Given

$$y_j(k) = \phi_j(v_j) \tag{2.14}$$

$$d_j(k) = \sum_{i=1}^{n} w_{ij} y_i \tag{2.15}$$

The error at the output of neuron j is given as:

$$e_j(k) = y_j(k) - d_j(k) \tag{2.16}$$

Where d_j is the desired output, y_j is the neuron output, and k indicates the k^{th} output. The sum of the squared output errors is given by:

$$\varepsilon(k) = \frac{1}{2} \sum_{j=1}^{l} e_j^2(k) \tag{2.17}$$

l is the number of neurons of the output layer. The average squared error energy is obtained by summing equation (2.17) over all n and then normalising with respect to the set size N:

$$\varepsilon_{av} = \frac{1}{N} \sum_{k=1}^{N} \varepsilon(k) \tag{2.18}$$

The objective at this stage is to adjust the parameters (synaptic weights and bias levels) of the network to minimize equation (2.18) using an

optimization technique [Haykin 1999].

2.3.3 Radial basis function

Supervised neural network can be designed in a variety of ways [Haykin 1999]. The above section described one of the ways a supervised neural network can be designed. The back-propagation method used in MLP networks is considered as the application of recursive technique or stochastic approximation [Haykin 1999]. Radial Basis Function (RBF) networks employ a different design approach which is considered as curve-fitting [Haykin 1999].

RBF networks are feed-forward networks trained using a supervised training algorithm. They are typically configured with a single hidden layer of units whose activation function is selected from a class of functions called basis functions [Haykin 1999].

While similar to back propagation in many aspects, radial basis function networks have several advantages. They usually train much faster than back propagation networks [Nabney 2001]. They are also less prone to problems with non-stationary inputs due to the behavior of the radial basis function [Hassoun 1995]. Figure 2.5 depicts the architecture of an RBF network. In back-propagation networks, all weights in all of the layers are adjusted at the same time. On the other hand in radial basis function networks, the weights into the hidden layer basis units are

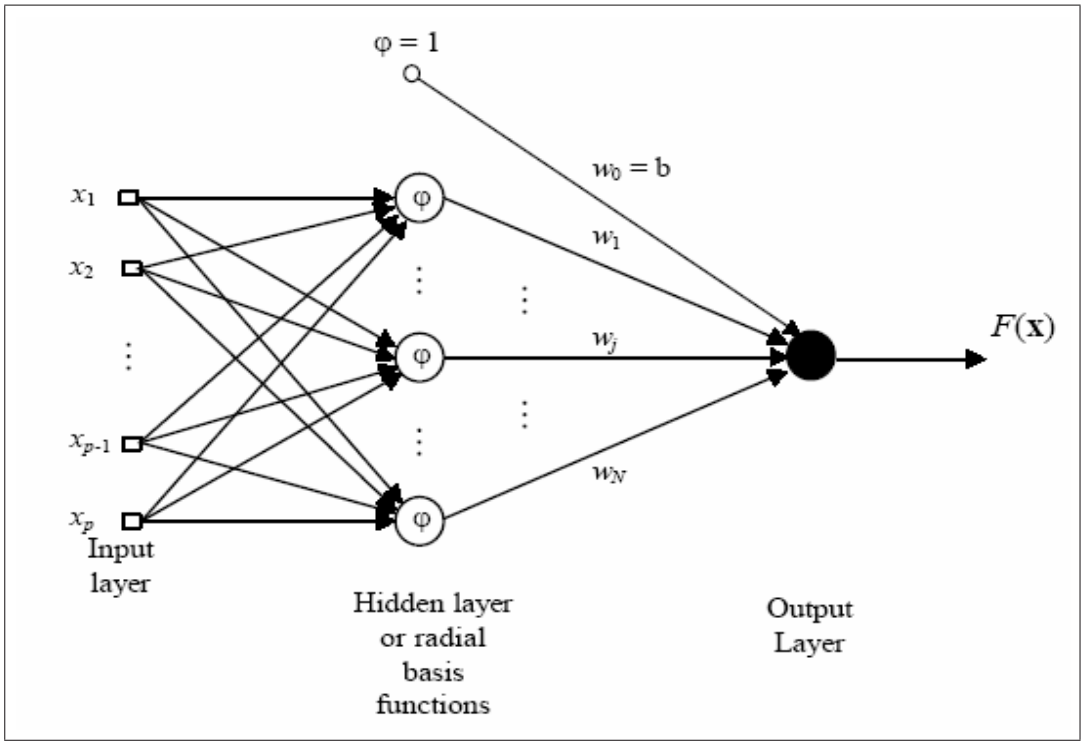


Figure 2.5: RBF architecture [Haykin 1999]

usually set before the second layer of weights are adjusted. Radial basis function mapping can be represented as [Nabney 2001]

$$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(x) + w_{ko} \tag{2.19}$$

Where ϕ_j are the basis functions, and w_{kj} are the output layer weights. The three commonly used basis functions in RBF are [Haykin 1999]

1. Gaussian functions:

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad for\ some\ \sigma > 0\ and\ r \in R \tag{2.20}$$

2. Multiquadrics:

$$\phi(r) = \left(r^2 + c^2\right)^{\frac{1}{2}} \qquad for\ some\ c > 0\ and\ r \in R \tag{2.21}$$

3. Inverse multiquadrics:

$$\phi(r) = \frac{1}{(r^2 + c^2)^{\frac{1}{2}}} \qquad for\ some\ c > 0\ and\ r \in R \tag{2.22}$$

2.4 Chapter summary

This chapter presented background material related to artificial neural networks. A neural network is an information processing paradigm that is inspired by the way biological nervous systems, like the brain process information. It is designed to model the way in which the brain performs a particular task or function of interest. The interconnection of units in neural networks affect the performance of neural networks, hence, there are various ways the units can be connected in neural network and based on the connections between the units they are classified into various architectures. MLP and RBF network architectures are used in this research and more specific details related to the MLP and RBF architecture used in this research are given in Chapter 5.

Chapter 3

Genetic Algorithms

Genetic algorithms (GAs) are defined as algorithms that are used to find approximate solutions to difficult problems through application of the principles of evolutionary biology [Michalewicz 1996]. Genetic algorithms use biologically derived techniques such as inheritance, mutation, natural selection, and recombination [Banzhaf *et al.* 1998].

The idea behind genetic algorithms is to do what nature does and they use vocabulary borrowed from natural genetics [Michalewicz 1996]. GAs have been proved to be successful in optimization problems like wire routing, scheduling, adaptive control, game playing, cognitive modeling, transportation problems, traveling salesman problems, optimal control problems and database query optimization problems [Michalewicz 1996].

Genetic algorithms view learning as a competition among a population of evolving candidate problem solutions [Alfonseca 1991; Back *et al.* 1992]. A fitness function evaluates each solution to decide whether it will con-

tribute to the next generation of solutions. Through operations analogous to gene transfer in sexual reproduction, the algorithm creates a new population of candidate solutions [Banzhaf *et al.* 1998].

3.1 Why genetic algorithms?

There are so many ways that makes a genetic algorithm superior to other optimization techniques. GAs differ from traditional optimization and search methods in several aspects [Jones and Konstam 1999; Pendharkar and Rodger 1999]. Rather than focusing on a single candidate solution genetic algorithms operate on populations of candidate solutions, and the search process favors the reproduction of individuals with better fitness values than those of previous generations [Goldberg 1989; Michalewicz 1996]. Whereas calculus-based and other traditional optimization methods of solution are local in the scope of their search and depend on well-defined gradients in the search space.

Thus, GAs do not only differ in approach from traditional optimization methods but also offer an alternative method for cases in which traditional methods are inappropriate. Genetic algorithm as a discrete optimization process is distinct from more conventional optimization techniques in four ways [Goldberg 1989; Michalewicz 1996; Forrest 1996]:

- GAs encode designs in a string, and it is this encoding which the GA works with. Each individual in a population is an encoding

of a possible solution to the discrete optimization problem being analyzed.

- GAs work simultaneously with a population of designs, not a single design or candidate solution.
- GAs use only an objective function to evaluate candidate solutions, not derivatives or other auxiliary information.
- GAs use random change in their search, not solely on deterministic rules.

Taking the above mentioned merits of genetic algorithm into consideration, the choice of genetic algorithm as an optimization method in this research was imminent from the fact that genetic algorithms provide more feasible and optimum solution than other ordinary optimization methods.

3.2 Structure of genetic algorithm

The procedure of a genetic algorithm is given as follows [Goldberg 1989; Michalewicz 1996]

1. Generate randomly a population of solutions.
2. Calculate the fitness for each of the population elements.
3. Create offsprings by three genetic operators (reproduction, crossover, and mutation).

4. Evaluate the new solution and calculate the fitness of each solution
5. If optimum solution is achieved, stop and return, otherwise go to step 3.

Thus, the most common steps involved in genetic algorithm for solving a particular problem involves [Michalewicz 1996, p.17] the following:

- Representation of the problem.
- Generation of an initial population of potential solutions.
- Developing fitness or evaluation function.
- Determining genetic operators that alter the composition of children.
- Determining various parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators, etc.).

The following pseudo-code from Michalewicz [1996] illustrates the high level description of how genetic algorithms work. $P(t)$ represents the population at generation t.

```
procedure genetic algorithm
  begin
    t ← 0
    initialise P(t)
    evaluate P(t)
    while (not termination condition) do
    begin
      t ← 0
      select P(t) from P(t − 1)
      alter P(t)
      evaluate P(t)
    end
  end
```

Michalewicz [1996] states that, unlike evaluation programs which leave the original problem unchanged, a genetic algorithm transforms the problem into appropriate form. This process is depicted in Figure 3.1 where the problem will be modified first in some form before the genetic algorithm is applied.

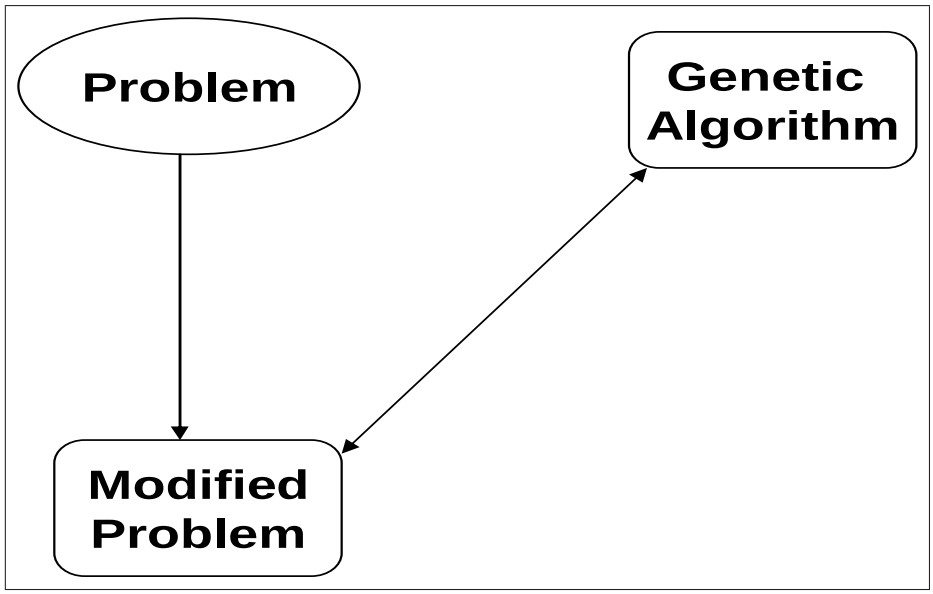


Figure 3.1: Genetic algorithm approach [Michalewicz 1996]

The major steps involved in genetic algorithms are Encoding, Evaluation, Crossover, Mutation and Decoding. To make the discussion more easier a basic explanation is given to understand how the different steps of genetic algorithms work in solving a particular optimization problem.

The following problem will be used throughout this section to make the discussion on genetic algorithm concepts more easier. The example will be to maximize the function $f = -x^2 + 10x + 2$ over the integer interval of [0, 10].

$$f(x) = -x^2 + 10x + 2 \tag{3.1}$$

3.3 Optimization of a simple function

Applying elementary mathematical techniques, it can be easily observed that equation (3.1) is maximized when x=5. Now the following discussion will lead us on how genetic algorithm can be used to maximize this function to find the approximate maximum solution.

3.3.1 Representation of the problem

The first and most difficult step in genetic algorithm is encoding the problem in a suitable manner. It is often the most difficult aspect of solving a problem using genetic algorithms. When applying them to a specific problem it is often hard to find an appropriate representation of the solution that will be easy to use in the crossover process.

It should be possible to encode many possible solutions to create the initial population. The traditional way to represent a solution is with a string of zeros and ones. However genetic algorithms are not restricted to this encoding, they can also be represented in real or some other way [Michalewicz 1996]. For this example a binary string representation will be used.

Considering the problem defined above. The possible solutions are obviously just numbers, so the representation is simply the binary form of each number. For instance, the binary representations of 7 and 8 are 0111 and 1000, respectively. Note that zero is added to the beginning of the string 0111 even though it has no real meaning. It is added so that all the numbers in the set have the same length. These strings are called chromosomes and each element (or bit) of the string is called a gene. After representing the problem, the next step is generating randomly many chromosomes and they constitute the initial population.

3.3.2 Evaluation

The evaluation function plays an important role in genetic algorithms. The evaluation function is used to decide how good a chromosome is. The evaluation function usually comes straight from the problem. In this case the evaluation function would simply be the function $f = -x^2 + 10x + 5$, and the larger the value for f, the better, as the aim is to maximize the function. So, in this case, the evaluated values for 7 and 8 are

$$f(7) = 26$$

$$f(8) = 21$$

From the above evaluation function it can be observed that 7 is a better solution than 8, because 7 has a higher fitness than 8. This fitness is then used to decide the probability that a particular chromosome would be chosen to contribute to the next generation. The scores are normalized and then used to create a cumulative probability distribution. The cumulative probability distribution is used in the crossover process.

The stopping criteria is used in the evaluation process to determine whether or not the current generation and the best solution found so far are close to the global optimum. Various stopping criteria can be used, and usually more than one is employed to account for different possibilities during the running of the program. The decision depends on whether an optimal solution is found, optimal solution is not found, a local optimum is found, and etc.

The standard stopping criteria that is used to stop the procedure is after a given number of iterations which is called the number of generations in genetic algorithm. This is so that if we do not find a local optimum or a global optimum and do not converge to any one point, the procedure will still stop at some given time.

Another stopping criteria is to stop after the best solution has not changed over a specified number of iterations. This will usually happen when an

optimum solution which can be either local, global or a point near the optimum solution is found. Another stopping criteria is when the average fitness of the generation is the same or close to the fitness of the best solution.

3.3.3 Crossover

Crossover can be a fairly straightforward procedure. In this example, which uses the simplest case of crossover, two chromosomes are randomly chosen to crossover. Randomly pick a crossover point, and then switch all genes after that point. For example, using the chromosomes

$$v_1 = 0111$$
$$v_2 = 1000$$

It can be randomly chosen the crossover point to be after the second gene

$$v_1 = 01 \mid 11$$
$$v_2 = 10 \mid 00$$

Switching the genes after the crossover point would give

$$v_1' = 0100 = 4$$
$$v_2' = 1011 = 11$$

Now the two new chromosomes will be moved into the next population called the next generation. Not every chromosome is used in crossover.

The evaluation function gives each chromosome a score which is used to decide that chromosomes probability of crossover. The chromosomes are chosen to crossover randomly and the chromosomes with the highest scores are more likely to be chosen.

The cumulative distribution created in the evaluation stage is used to choose the chromosomes. A random number between zero and one is generated to choose which chromosome corresponds to the distribution. This is done again to get a pair, then the crossover is performed and both the new chromosomes are moved into the new generation.

This will hopefully mean that the next generation will be better than the last because only the best chromosomes from the previous generation were used to create this generation. Crossover continues until the new generation is full.

It is possible to check each new chromosome to make sure it does not already exist in the new generation. This means that a variety of possible solutions in each generation will be obtained, but also that once the optimal solution in one chromosome is found, the other chromosomes will not probably be optimal.

This means that the average fitness of the generation can never be as good as the fitness of the optimal chromosome, which could be used as a decision criteria on when to stop. It is also possible to move the best solution from the previous generation directly into the new generation.

This means that the best solution can never get any worse since even if on average the generation is worse, it will still include the best solution so far. There can also be two point crossover. In this case we randomly choose two crossover points and switch the genes between the two points. In our problem we could pick the points after the first gene and after the third gene.

$$v_1' = 0 \mid 11 \mid 1$$
$$v_2' = 1 \mid 00 \mid 0$$

to get

$$v_1'' = 0001 = 1$$
$$v_2'' = 1110 = 14$$

There are many different crossover methods which can be used in different manners, accordingly. It is important to choose the crossover method so that it will not be possible to reach unacceptable chromosome (an infeasible solution).

3.3.4 Mutation

The next step that follows after cross over is mutation. Mutation is used so that the solution will not be confined to a local optimum, which makes genetic algorithm peculiar from other optimization techniques like

hill climbing and simulated annealing. Due to the randomness of the process occasionally the chromosomes can be near a local optimum solution rather than near the global optimum solution. Therefore the chromosomes near the local optimum will be chosen to crossover because they will have the better fitness and there will be very little chance of finding the global optimum solution.

Mutation is therefore a completely random way of getting to possible solutions that would otherwise not be found. Mutation is performed after crossover by randomly choosing a chromosome in the new generation to mutate. A random point is chosen to mutate by switching that point. For instance, in the example we had $v_1 = 0111$. If we chose the mutation point to be gene three, v_1 would become $v_1' = 0101$. We simply changed the 1 in position three to a 0. If there had been a 0 in position three then we would have changed it to a 1. This is extremely easy in our example but we do not always use a string of zeros and ones as our chromosome.

Like crossover, mutation is designed specifically for the problem that it is being used on. Inversion is a different form of mutation. It is sometimes used in appropriate cases. Here the inversion operator on our basic example will be explained. The inversion operator consists of randomly choosing two inversion points in the string and then inverting the bits between the two points. For instance $v_2 = 1000$, the two points are chosen after gene one and after gene three ($v_2 = 1 \mid 00 \mid 0$).

Now, since there are only two genes between the inversion points, they

are switched to give $v_2' = 1000$. If there was a larger chromosome, say $v_3 = 110100101001111$ the inversion points can be chosen after the third point and after the eleventh point. $v_3 = 110 \mid 00101001 \mid 1111$. Now, we start at the ends of the cut region and switch the genes at either end moving in to get $v_3' = 110001010011111$. Essentially we are just reversing (or inverting) the order of the genes in between the two chosen points.

3.3.5 Decoding

After repeating this step up to a given number of generations, the final stage is to convert the final strings of zeros and ones to its corresponding real value. This value found is supposed to be the optimum solution obtained by the genetic algorithm.

In this example if the final chromosome was 0100, its corresponding real value is 4 and the value of the function evaluated at this point is

$$f(4) = -x^2 + 10x + 5 = (-4)^2 + 10(4) + 5 = 29.$$

Figure 3.2 shows the graphic representation of the above genetic operations that are performed by genetic algorithm. Initially at generation T_n we have a population consisting of four chromosomes. After applying mutation and crossover we get a new population at generation T_{n+1}, which is more fit (evaluated in terms of a fitness function) than the initial population (chromosomes).

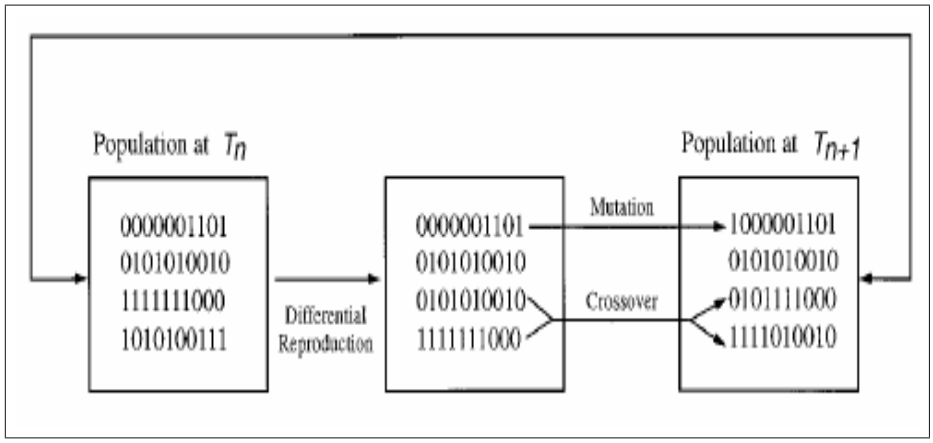


Figure 3.2: Genetic algorithm overview [Forrest 1996]

Other important steps in genetic algorithms are determining the parameters like population size, probability of crossover, and probability of mutation.

3.4 Chapter summary

Genetic algorithms are algorithms that are used to find approximate solutions to difficult problems through application of the principles of evolutionary biology. It uses the principles of natural selection to find the global optimum solution to problems that other optimization techniques are not sufficiently capable. Genetic algorithms have been proved to be successful in many classical optimization problems.

Chapter 4

Missing Data

Missing data refers to the case that some of the components of the data vectors are not available for all data items in the database, or may not even be applicable or defined [Rubin 1978]. This creates various problems in many applications which depend on good access to complete data. Consequently methods to handle missing data in database have been an area of research for long time in various disciplines [Yuan 2000; Allison 2000; Rubin 1978].

There are many ways that a missing data may occur in database. It may occur due to respondents non-response to questions in the data collection process, data entry process, and other various cases. There are also other situations in which missing data may occur due to failure of instruments in recording process. The following sections present mechanisms and pattern of missing data and the various methods used to handle missing data in databases.

4.1 Missing data mechanisms

The reasonable way to handle missing data depends upon how data points become missing. Little and Rubin [1987] define three missing data mechanisms. These are Missing Completely at Random, Missing at Random, and non-ignorable case.

4.1.1 Missing completely at random

Missing Completely At Random (MCAR) situation arises if the probability of missing value for variable X is unrelated to the value X itself or to any other variable in the data set. This means that the missing data does not depend on the variable of interest or any other variable in the data set [Rubin 1978]. In other words missing data values are simple random samples of all data values in the database. The missing data for a variable age for example is said to be MCAR if the missing value is unrelated to the variable age itself or to the values of any other variable in the database, whether missing or observed [Allison 2002].

Another example of MCAR is that if people who do not report their income are the same as people who do report income, income in this case is considered as MCAR. In this situation, cases with complete data are indistinguishable from cases with incomplete data.

In Table 4.1 the missingness of the missing value in x_3 is said to be MCAR

if the missingness does not depend on x_1, x_2, x_4, and x_5 and the variable x_3 itself.

Table 4.1: Table with missing entries

Observation	x_1	x_2	x_3	x_4	x_5
1	25	3.5	?	5000	-3.5
2	?	6.9	5.6	?	0.5
3	45	3.6	9.5	1500	46.5
4	27	9.7	?	3000	?

4.1.2 Missing at random

Missing at Random (MAR) arises if the probability of missing data on a particular variable X depends on other variables, but not on X itself. If the probability of missing income depends on marital status but within each category of marital status, the probability of missing income is unrelated to the value of income, income in this case is considered as MAR [Little and Rubin 1987].

Cases with incomplete data differ from cases with complete data but the pattern of missingness is traceable or predictable from other variables in the database rather than being due to the specific variable on which the data are missing. MAR means the value for the variable is missing, but conditional on some other X variable observed in the data set, although not on the Y variable of interest [Scheffer 2000]. Thus, the probability of missing data on any variable is not related to its particular value.

In Table 4.1 the missingness of the missing value in x_3 is said to be MAR

if the missingness depends on x_1, x_2, x_4, and x_5 but not in the variable x_3 itself.

4.1.3 Non ignorable case

The third type of missing data mechanism is non ignorable. The non ignorable case arises if the probability of missing data X is related to the value of X itself even if we control the other variables in the analysis [Allison 2000]. This means missing data are not random and depend on the values that are missing.

If high income households are less likely to report their income even after adjusting for other variables, then the probability of missing income is said to be non ignorable [Little and Rubin 1987]. In this case the pattern of data missingness is non-random and it is not predictable from other variables in the database. Non ignorable missing data is the most difficult to approximate and model than the other two missing mechanisms [Rubin 1978].

In Table 4.1 the missingness of the missing value in x_3 is said to be non ignorable if the missing value in x_3 depends on the variable x_3 itself.

4.2 Patterns of missing data

Little and Rubin [1987] defined two ways of missing data patterns. These

are arbitrary and monotone missing pattern. In arbitrary missing data, missing observation may occur anywhere and ordering of variables is unimportant [Rubin 1978]. In monotone missing pattern the ordering of variables is important. In monotone a data set with variables x_{1+1}, x_{1+2}, $x_{1+3}...x_{1+n}$ in the order is said to have a monotone missing pattern, if a variable x_j is observed for a particular individual it implies that all previous variables x_k, where $k < j$, are also observed for that individual.

Table 4.2 shows an arbitrary missing data and Table 4.3 shows a monotone missing data pattern. In Table 4.2 the missing values are random and can happen in any place, whereas in Table 4.3 it can be observed that missing values have some common order. That is if a value for a variable x_j is missing so are the values for other variable i, where $i > j$.

Table 4.2: Arbitrary missing data pattern

Observation	x_1	x_2	x_3	x_4	x_5
1	25	3.5	?	5000	-3.5
2	?	6.9	5.6	?	0.5
3	45	3.6	9.5	1500	46.5
4	27	9.7	?	3000	?

Table 4.3: Monotone missing data pattern

Observation	x_1	x_2	x_3	x_4	x_5
1	25	3.5	15	?	?
2	2.0	6.9	?	?	?
3	45	3.6	9.5	1500	46.5
4	27	?	?	?	?

4.3 Missing data imputation algorithms

Depending on the situation and data missing mechanisms, currently there are various data imputation methods used in statistical packages [Yansaneh *et al.* 1998]. These include simple methods like listwise or casewise data deletion to methods employing sophisticated statistical and artificial intelligence techniques. The following discussion presents some of the most commonly used missing data handling mechanisms. The discussion starts with the simple methods and proceeds to introduce the most complicated and efficient ones.

4.3.1 Listwise or casewise data deletion

Many statistical procedures will eliminate an entire observation or case if there are any missing data in the defined variables. This is known as listwise or casewise data deletion and occurs when a record has missing data for one or more identified variables.

The listwise or casewise data method is the simplest and easy way of treating data. But it is also the worst choice of treating missing data. In this method a record containing missing data for any variable is omitted.

Note that treating missing data using this method is plausible only if the missing data are very small compared to the available complete data in the database [Little and Rubin 1987]. Otherwise, using this method when

the missing data is relatively larger than the complete data, it may lead to a biased estimates to be obtained from the database. Taking Table 4.2 as an example casewise deletion method will remove records number 1, 2, and 4 and proceed the analysis based on the remaining records.

4.3.2 Pairwise data deletion

The pairwise data deletion method works by making the required analysis from available pairwise data. This means a record with missing data on one variable will be used only in calculations that do not involve that variable. In this manner, the sample size is often larger than when using complete case analysis. Allison [2002] points that unless the data are MCAR, pairwise deletion produces biased estimates and is not recommended.

Taking Table 4.2 as an example pairwise deletion method means record number 1 will be used, whenever there is any analysis that do not involve variable x_3.

4.3.3 Mean substitution

In this method, the variable's mean value is calculated from the available cases and is used as the imputed value for the missing cases. As with the pairwise deletion method, mean substitution has a high likelihood of producing biased estimates and is not recommended [Allison 2002] .

Employing mean substitution method to Table 4.2 the values for all missing values in variable x_3 will be substituted by averaging the values of the available values in that variable. In this case the value will be

$$\frac{\sum_{i=1}^{2} x_i}{n} = \frac{5.6 + 9.5}{2} = 7.55 \tag{4.1}$$

4.3.4 Hot deck imputation

In hot deck imputation method, we identify the most similar case to the case with a missing value and substitute the most similar case's x value for the missing case's x value [Allison 2002; Scheffer 2000].

Considering Table 4.2, using this method the value for the missing value in the first record will be substituted by finding the most similar record to record number one and substituting the most similar record's x_3 value to record one x_3 variable.

Case two and four have a missing data cell. Hot deck imputation examines the cases with complete records (case three in this example) and substitutes the value of record number four into one. Thus, the missing value in record one will be substituted by 9.5. In this case since there is only one complete record, there is no way where we can make a similarity comparison. But, in case there are more than one complete record in the database the most similar case for the missing data point have to be substituted.

Once the hot deck imputation determines which case among the observations with complete data is the most similar to the record with incomplete data, it substitutes the most similar complete case's value for the missing variable into the data matrix.

Some of hot deck's advantages are its conceptual simplicity, its maintenance of the proper measurement level of variables thus a categorical variable remains categorical and continuous variable remains continuous, and the availability of a complete data at the end of the imputation process that can be analyzed like any complete data [Roth 1994; Rubin 1978; Hu *et al.* 1998].

One of hot deck's disadvantages is the difficulty in defining similarity. There may be any number of ways to define what similarity is in this context. Thus, the hot deck procedure is not an out of the box approach to handling incomplete data [Hu *et al.* 1998]. More sophisticated hot deck algorithms identify more than one similar record and then randomly select one of those available donor records to impute the missing value or use an average value if it is appropriate [Hu *et al.* 1998; Scheffer 2000; Little and Rubin 1987].

4.3.5 Regression methods

In regression method we develop a regression equation based on complete case data for a given variable, treating the missing variable as dependent

variable and using all other relevant variables in the database as predictors. For the records where the value is missing we predict or approximate its value by the regression equation developed in terms of other variables [Little and Rubin 1987].

Note that, in this method a regression model is fitted for each variable with missing values, with the other variables as dependents. The process is repeated sequentially for variables with missing values, which means that for a variable x_j with missing values, a model is fitted using observations with observed values for the other variables.

Using regression method in Table 4.2 to approximate the missing value in record 1, a regression equation will be fitted in terms of variables x_1, x_2, x_4, and x_5. The equation can be formulated as

$$x_3 = b_1x_1 + b_2x_2 + b_4x_4 + b_5x_5 + \epsilon \tag{4.2}$$

The fitted model includes the regression parameter estimates b_i and an error ϵ. Equation 4.2 can then be used to estimate the missing value by plugging the values of x_1, x_2, x_4, and x_5 into equation (4.2).

4.3.6 Expectation maximization

The Expectation maximization (EM) approach is an iterative procedure that proceeds in two steps [Little and Rubin 1987]. The first step called the expectation (E) step computes the expected value of the complete

data log likelihood based upon the complete data cases and the algorithm's best guess as to what the sufficient statistical functions are for the missing data based upon the model specified and the existing data points.

In the second step called the maximization (M) step, it substitutes the expected values for the missing data obtained from the E step and then maximizes the likelihood function as if no data were missing to obtain new parameter estimates. The new parameter estimates are substituted back into the E step and a new M step is performed. The procedure iterates through these two steps until convergence is obtained. Convergence occurs when the change of the parameter estimates from iteration to iteration becomes negligible.

Thus, the main steps involved in EM approach are [Little and Rubin 1987]:

- Replace missing values by estimated values.
- Estimate parameters.
- Reestimate the missing values assuming the new parameter estimates are correct.
- Reestimate parameters, iterating until convergence.

The advantage of the expectation maximization approach is that it has well known statistical properties and it generally performs better than

methods such as listwise, pairwise data deletion, and mean substitution because it assumes incomplete cases have data missing at random rather than missing completely at random [Allison 2002; Rubin 1978].

The main disadvantage of the EM approach is that it adds no uncertainty component to the estimated data. Practically speaking, this means that while parameter estimates based upon the EM approach are reliable, standard errors and associated test statistics are not [Allison 2002; Rubin 1978]. This weakness led to the development of two newer likelihood based methods for handling missing data, the raw maximum likelihood approach (full information maximum likelihood) and multiple imputation.

4.3.7 Raw maximum likelihood methods

Raw maximum likelihood, which is also known as Full Information Maximum Likelihood (FIML), methods use all available data points in a database to construct the best possible first and second order moment estimates under the MAR assumption. In simple terms, if the missing at random (MAR) assumption can be met, maximum likelihood-based methods can generate a vector of means and a covariance matrix among the variables in a database that is superior to the vector of means and covariance matrix produced by commonly used missing data handling methods such as listwise deletion, pairwise deletion, and mean substitution [Little and Rubin 1987].

It uses all available data to calculate a vector of means and covariance matrix (i.e., maximum likelihood-based sufficient statistics) in a way that is superior to other methods. Under an unrestricted mean and covariance structure, raw maximum likelihood and EM return identical parameter estimate values. Unlike EM, however, raw maximum likelihood can be employed in the context of fitting user specified linear models, such as structural equation models, regression models, etc.

Raw maximum likelihood also produces standard errors and parameter estimates under the assumption that the fitted model is not false, so parameter estimates and standard errors are model-dependent. That is, their values will depend upon the model chosen and fitted by the investigator.

Raw maximum likelihood has the advantage of convenience or ease of use and well known statistical properties. Unlike EM, it also allows for the direct computation of appropriate standard errors and test statistics. Disadvantages include an assumption of joint multivariate normality of the variables used in the analysis and the lack of a raw data matrix produced by the analysis [Rubin 1978; Little and Rubin 1987; Scheffer 2000].

Raw maximum likelihood methods are also model based. That is, they are implemented as part of a fitted statistical model. The investigator may want to include relevant variables that will improve the accuracy of parameter estimates, but not include these variables in the statistical

model as predictors or outcomes. While it is possible to do this, it is not always easy or convenient, particularly in large or complex models.

Finally, raw maximum likelihood assumes the incomplete data cells are missing at random [Little and Rubin 1987]. Raw maximum likelihood can offer superior performance to listwise and pairwise deletion methods even in the non ignorable data situation [Hu *et al.* 1998].

4.3.8 Multiple imputation

Multiple imputation (MI) is similar to raw maximum likelihood but it creates five to ten data sets in which raw data are generated that can be used to fill in the missing data [Scheffer 2000]. The data from the imputed data set are then pooled and parameters are estimated.

Multiple imputation combines the well known statistical advantages of EM and raw maximum likelihood with the ability of hot deck imputation to provide a raw data matrix to analyze [Allison 2000; Scheffer 2000]. Multiple imputation works by generating a maximum likelihood based covariance matrix and vector of means, like EM. Multiple imputation takes the process one step further by introducing statistical uncertainty into the model and using that uncertainty to emulate the natural variability among cases one encounters in a complete database. Multiple imputation then imputes actual data values to fill in the incomplete data points in the data matrix, just as hot deck imputation does [Little and Rubin

1987].

The primary difference between multiple imputation and hot deck imputation from a practical or procedural standpoint is that multiple imputation requires that the data analyst generate five to ten databases with imputed values. The data analyst then analyzes each database, collects the results from the analyses, and summarizes them into one summary set of findings. For instance, suppose a researcher wishes to perform a multiple regression analysis on a database with incomplete data. The researcher would run multiple imputation, generate ten imputed databases, and run the multiple regression analysis on each of the ten databases. The researcher then combines the results from the ten regression analyses together into one summary.

Multiple imputation has several advantages. It is fairly well understood and robust to violations of non-normality of the variables used in the analysis. Like hot deck imputation, it outputs complete raw data matrices. It is clearly superior to listwise, pairwise, and mean substitution methods of handling missing data in most cases. Disadvantages include the time intensiveness in imputing five to ten databases, testing models for each database separately, and recombining the model results into one summary.

Multiple imputation (MI) appears to be one of the most applicable methods for general purpose handling of missing data in multivariate analysis. The basic idea of multiple imputation as presented in Little and Rubin

[1987] are:

1. Impute missing values using an appropriate model that incorporates random variation.

2. Repeat this M times (usually 3-5 times), producing M complete data sets.

3. Perform the desired analysis on each data set using standard complete data methods.

4. Average the values of the parameter estimates across the M samples to produce a single point estimate.

5. Calculate the standard errors by (a) averaging the squared standard errors of the M estimates (b) calculating the variance of the M parameter estimates across samples, and (c) combining the two quantities using a simple formula.

Currently, raw maximum likelihood and MI methods appear to be the methods of choice for handling missing data. Other methods such as expectation maximization, regression, and hot deck imputation do not have any notable advantages over raw maximum likelihood or MI. MI methods are particularly flexible for a wide variety of linear and nonlinear models. Even when missing data are non ignorable it has been observed that raw maximum likelihood outperforms pairwise deletion and complete case analysis methods [Scheffer 2000; Allison 2000; Hu *et al.* 1998].

4.4 Chapter summary

Missing data refers to the case that some of the components of the data vectors are not available for all data items in the database, or may not even be applicable or defined. This creates various problems in many applications which depend on good access to complete data. Currently there are various methods of dealing with missing data. Each of the methods have their pros and cons, and they are used in different situations based on the mechanism of the missing data in the database. Since multiple imputation and raw maximum likelihood perform better even when the missing data is non ignorable, they appear to be the methods of choice for handling missing data in most cases.

Chapter 5

Research Method

5.1 Introduction

In discussing the methodology used in the research this chapter first states, in the form of the research question, the specific research questions examined in the research. Formal statement of the hypotheses derived from the research question details expected results in the research. Detailed description of the proposed model used in the research also is presented. Approach followed and analysis taken to answer the research question are detailed thereafter. Finally, procedures on how the approach followed in the research was used to accept or reject the hypotheses is highlighted.

5.2 Research question and statement of hypothesis

It should be clear at this stage that the study concentrated on how it would be possible to efficiently approximate missing values in a database. In looking at this, the research attempted to answer four main questions:

- *Is it possible to approximate or predict missing values in a database efficiently using a model employing an artificial neural networks and genetic algorithms?*
- *Is there any relationship between the accuracy of the approximated or predicted values and the number of missing cases/variables in a single record?*
- *Does approximated values found using the proposed model depend on the particular neural network architecture employed in training the neural network?*
- *Does using the combination of both architectures (MLP+RBF) enhance the accuracy of approximated values?*

The expected answers to these questions, in the light of prior research done and related work, are detailed in the hypotheses of this research.

Hypothesis One

- *It is possible to approximate or predict the missing values in a database*

efficiently using a model that employs an artificial neural networks and genetic algorithms.

Hypothesis Two

- *It is expected that as the number of missing cases within a single record gets larger the approximation tends to be less reliable.*

Hypothesis Three

- *Approximated values found using the proposed model depends on the particular neural network architecture employed in training the neural network.*

Hypothesis Four

- *It is expected that using a combination of both architecture (MLP+RBF) enhances the accuracy of approximated values.*

5.3 Hypotheses testing

To accept or reject the above hypotheses, they were tested individually as follows:

In order to test hypothesis one, approximated values from the proposed model and the actual values corresponding to each missing value were

compared. To examine the degree of relationship between the actual and approximated values using the model, correlation coefficient was calculated between the actual and approximated values. The correlation coefficient was used to decide whether to accept or reject the hypothesis.

Hypothesis two was tested using approximated values obtained from the model as the number of missing cases within a single record gets larger and actual corresponding missing values. This was done by calculating coefficient of correlation and standard error between the actual and approximated values and observing the value of the correlation coefficient, and standard error as the number of missing cases get larger. Decision to accept or reject the hypothesis was done depending on the values of correlation coefficient and standard error.

Hypothesis three was tested using approximated values obtained from the proposed model using different neural network architectures and actual corresponding missing values. This was done by calculating coefficient of correlation and standard error between actual and approximated values and observing the value of the correlation coefficient and standard error obtained by training the neural networks using different neural network architectures.

Hypothesis four was tested using approximated values obtained from the proposed model using MLP, RBF, and MLP+RBF, and actual corresponding missing values. This was done by calculating coefficient of correlation and standard error between actual and approximated values

and observing the value of the correlation coefficient and standard error obtained by MLP, RBF individually and the combined (MLP+RBF).

5.4 Proposed method

The neural network was trained to recall to itself or predict its input vector (auto-associative neural network). Mathematically the neural network can be represented as

$$\vec{Y} = f(\vec{X}, \vec{W}) \tag{5.1}$$

where $\vec{Y}$ is the output vector, $\vec{X}$ the input vector and $\vec{W}$ the vector of weights. Since the network is trained to predict its own input vector, the input vector $\vec{X}$ will be approximately equal to output vector $\vec{Y}$ ($\vec{X} \approx \vec{Y}$).

In reality the input vector $\vec{X}$ and output vector $\vec{Y}$ will not always be perfectly the same hence, we will have an error function expressed as the difference between the input and output vectors. Thus, the error can be formulated as

$$e = \vec{X} - \vec{Y} \tag{5.2}$$

Substituting the value of $\vec{Y}$ from equation (5.1) into equation (5.2) we get

$$e = \vec{X} - f(\vec{X}, \vec{W}) \tag{5.3}$$

We want the error to be minimum and non-negative. Hence, the error

function can be rewritten as the square of equation (5.3)

$$e = (\vec{X} - f(\vec{X}, \vec{W}))^2 \tag{5.4}$$

In the case of missing data, some of the values for the input vector $\vec{X}$ are not available. Hence, we can categorize the input vector ($\vec{X}$) elements into $\vec{X}$ known represented by ($\vec{X}_k$) and $\vec{X}$ unknown represented by ($\vec{X}_u$). Rewriting equation (5.4) in terms of $\vec{X}_k$ and $\vec{X}_u$ we have

$$e = \left(\left\{ \begin{array}{c} \vec{X}_k \\ \vec{X}_u \end{array} \right\} - f \left(\left\{ \begin{array}{c} \vec{X}_k \\ \vec{X}_u \end{array} \right\}, \vec{W} \right) \right)^2 \tag{5.5}$$

To approximate the missing input values, equation (5.5) is minimized using genetic algorithm. Genetic algorithm was chosen because it finds the global optimum solution [Goldberg 1989]. Since a genetic algorithm always finds the maximum value, the negative of equation (5.5) was supplied to the GA as a fitness function. Thus, the final error function minimized using the genetic algorithm is

$$e = - \left(\left\{ \begin{array}{c} \vec{X}_k \\ \vec{X}_u \end{array} \right\} - f \left(\left\{ \begin{array}{c} \vec{X}_k \\ \vec{X}_u \end{array} \right\}, \vec{W} \right) \right)^2 \tag{5.6}$$

Figure 5.1 depicts the graphical representation of proposed model. The error function is derived from the input and output vector (obtained from the trained auto-associative neural network). The error function is then minimized using genetic algorithm to approximate the missing variables

in the error function.

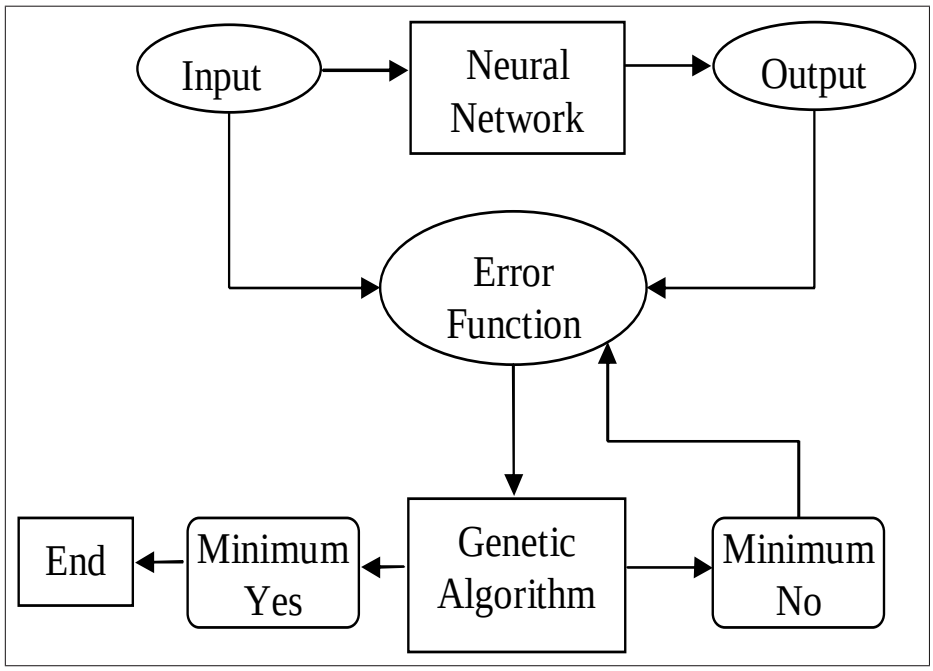


Figure 5.1: Schematic representation of proposed model

5.5 Selecting best neural network architecture

As discussed in chapter two, the result of a neural network depends on the particular network architecture and parameters (number of hidden layers, hidden units, activation and optimization functions) used in training the neural network. To select the best architecture that gives best results, different network architectures and parameters were selected and the network trained using the parameters. The architecture and parameters that gave best results were sleeted to be used in this research. Below are the MLP and RBF architectures and parameters used in this research.

5.5.1 MLP architecture used in the research

A fully connected two layered MLP architecture was used in the experiment. Each neuron in one layer is directly connected to the neurons of the subsequent layer. A NETLAB toolbox that runs in MATLAB discussed in Nabney [2001] was used to implement the MLP neural network. A two-layered MLP architecture was used because it resulted in better results and due to the universal approximation theorem, which states that a two layered architecture is adequate for MLP [Nabney 2001].

Figure 5.2 depicts the architecture of the MLP used in the research. The MLP network contains 14 inputs, 2 hidden layers with 10 neurons and 14 output units. A linear activation function (equation 2.11) discussed in Chapter 2 was selected. The optimisation technique used for training this architecture was the Scaled Conjugate Gradient (SCG) method. SCG method was used because it gave better results and has been found to solve the optimization problems encountered when training an MLP network more efficiently than the gradient descent and conjugate gradient methods [Bishop 1995].

5.5.2 RBF architecture used in the research

A fully connected two layered RBF architecture was used in the experiment. Each neuron in one layer is directly connected to the neurons of the subsequent layer. Like the MLP a NETLAB toolbox that runs in

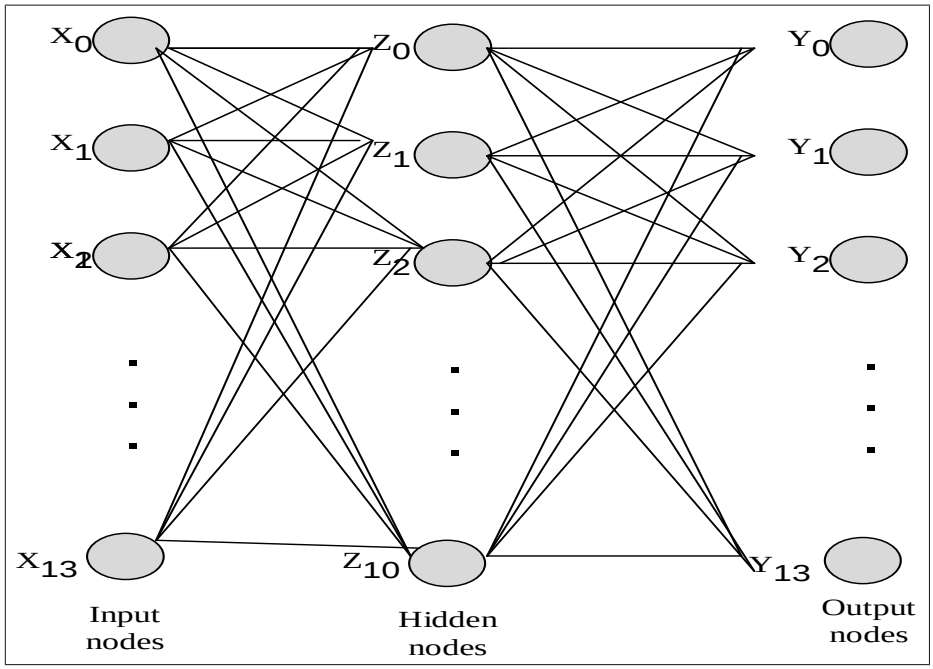


Figure 5.2: MLP architecture used in the research

MATLAB discussed in Nabney [2001] was used to implement the RBF architecture. Like the MLP, the network has 14 inputs, 10 neurons and 14 output units. The thin plate spline function was used as hidden unit activation function and the SCG was used as network optimization method. The RBF network used in this research is depicted in Figure 5.3.

5.6 Experimental data

The input data (real database) used in the experiment was obtained from the South African Breweries (SAB). The database used has 14 variables. To examine the distribution of the database and its representability in the investigation, key statistical summary of the database are given in Table 5.1. Where N represents the number of observations in the variable.

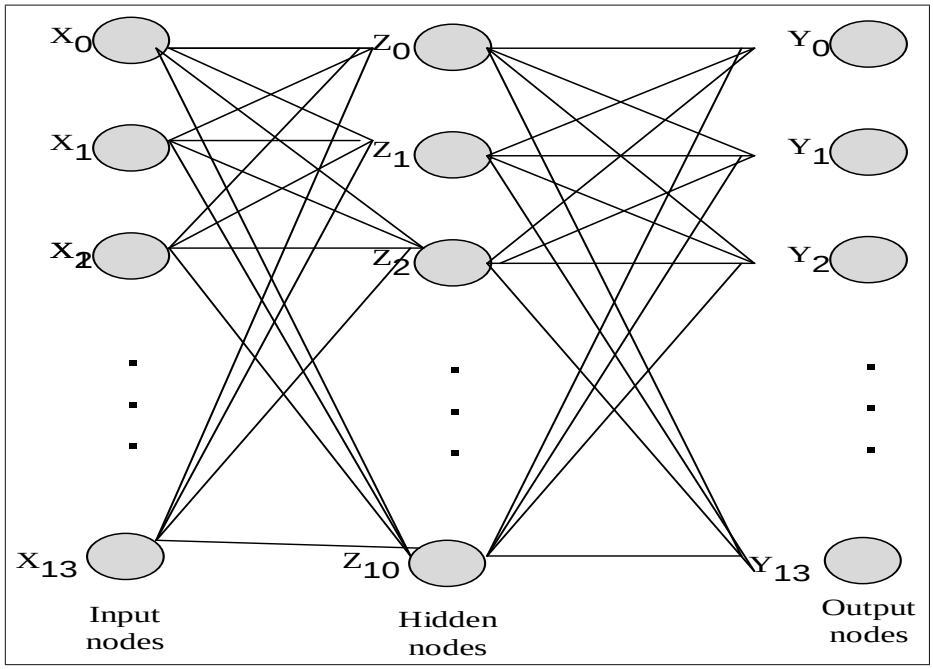


Figure 5.3: RBF architecture used in the research

Looking at the distribution of statistical summaries of the database, it can be observed that the database has ideal measures of central tendency and variation.

5.7 Genetic algorithm implementation

Genetic algorithm toolbox implemented in MATLAB by Houck *et al.* [1995] has been used in this research. After running (executing) the toolbox with different genetic operators, operators that gave better results were selected to be used in the experiment. Detailed operators and implementation of the genetic algorithm used in the research is presented in the next chapter.

Table 5.1: Statistical summary of input data

Variables	N	Mean	Min	Max	Standard deviation
x_1	198	4.248	3.900	4.540	0.117
x_2	198	6.925	5.600	8.800	0.549
x_3	198	9.442	0.000	39.800	7.355
x_4	198	21.230	12.900	25.000	1.720
x_5	198	63.652	34.000	98.000	13.968
x_6	198	0.042	0.010	0.180	0.022
x_7	198	161.419	72.000	286.000	82.501
x_8	198	2.090	1.000	3.200	0.239
x_9	198	0.342	0.100	0.800	0.148
x_{10}	198	0.160	0.000	0.300	0.057
x_{11}	198	35.384	13.000	64.000	8.005
x_{12}	198	5.824	1.700	13.600	2.130
x_{13}	198	20.325	8.000	38.000	4.714
x_{14}	198	1.849	1.000	4.300	0.620

5.8 Approach followed

The task of accepting or rejecting the hypotheses was approached by using data obtained from the experiment and analysis done on it. The steps followed in conducting the experiment are training of the MLP and RBF auto-associative neural networks and implementing Equation (5.6) as the fitness function in the genetic algorithm. Each value of the database was removed (considered as missing value) and then approximated using the model.

For the individual cases of MLP and RBF Equation (5.6) was used as the fitness function, where f in the equation refers to MLP or RBF. The fitness function used in the genetic algorithm for the combined case was a simple linear combination of the MLP and RBF networks. The error

function used in this case was the negative of Equation (5.7). In Equation (5.7) the first f refers to the MLP function and the second f refers to the RBF function.

$$\left[\left(\left\{\begin{array}{c}\vec{X}_k\\ \vec{X}_u\end{array}\right\} - f\left(\left\{\begin{array}{c}\vec{X}_k\\ \vec{X}_u\end{array}\right\}, \vec{W}\right)\right)^2 + \left(\left\{\begin{array}{c}\vec{X}_k\\ \vec{X}_u\end{array}\right\} - f\left(\left\{\begin{array}{c}\vec{X}_k\\ \vec{X}_u\end{array}\right\}, \vec{W}\right)\right)^2\right] \quad (5.7)$$

5.9 Data analysis

The following estimates are used to asses the accuracy and similarity of approximated values with the actual missing observations.

The estimates used to measure the modeling quality are:

- **Correlation coefficient (r)**: the correlation coefficient measures the linear relationship between two variables. The absolute value of "r" provides an indication of the strength of the relationship. The value of "r" varies between negative and positive 1, with -1 or 1 indicating a perfect linear relationship, and r = 0 indicating no relationship. The sign of the correlation coefficient indicates whether the two variables are positively or negatively related [Draper and Smith 1998]. For a given data $x_1, x_2, ..., x_n$ and corresponding approximated values $\hat{x}_1, \hat{x}_2, ..., \hat{x}_n$ the correlation coefficient is computed as

$$\mathrm{r} = \frac{\sum_{i=1}^{n}(x_i - \overline{x}_i)\left(\hat{x}_i - \overline{\hat{x}}_i\right)}{\left[\sum_{i=1}^{n}(x_i - \overline{x}_i)^2 \sum_{i=1}^{n}\left(\hat{x}_i - \overline{\hat{x}}_i\right)^2\right]^{1/2}} \quad (5.8)$$

Correlation coefficient (r) in this context measures the degree of relationship between the actual missing data and corresponding approximated values using the model. A positive value indicates a direct relationship between the actual missing data and its approximated value using the model.

- **Standard error (Se)**: the standard error represents average deviation between actual and predicted observations [Draper and Smith 1998]. For a given data $x_1, x_2, ..., x_n$ and corresponding approximated values $\hat{x}_1, \hat{x}_2, ..., \hat{x}_n$ the Standard error (Se) is computed as

$$\text{Se} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}{n}} \tag{5.9}$$

The higher the value of the standard error, the less the accuracy and vice versa.

5.10 Chapter summary

This chapter introduced the research methodology followed in conducting the research. The research question and hypotheses clearly stated the problem that the research tries to address. Correlation coefficient and standard error were used to asses accuracy of the model and approximated values. A fully connected two layered MLP and RBF architectures were used in the experiment. Each neuron in one layer is

directly connected to the neurons of the subsequent layer. A NETLAB toolbox that runs in MATLAB described in Nabney [2001] was used to implement both the MLP and RBF neural networks. A two-layered architecture was used because it gave better results. Genetic algorithm toolbox implemented in MATLAB discussed in Houck *et al.* [1995] was used to implement the genetic algorithm and genetic operators that gave better results were selected to run the genetic algorithm.

Chapter 6

Results and Discussion

6.1 Introduction

In this chapter experimental results and related discussion are presented. The first section presents the experiment done to evaluate the training of the neural network and measure the performance of the genetic algorithm in the experiment. Subsequent sections present methods investigated to approximate missing data in the database using MLP, RBF and their combination.

To test the hypotheses proposed in the previous chapter, correlation coefficient and standard error between the approximated and actual missing values are used. It is found that the approach approximates missing data with accuracy of 95% correlation coefficient between the actual missing data and corresponding approximated values.

6.2 Training of neural network

An MLP and RBF neural networks with 10 hidden neurons, 14 inputs and 14 outputs were trained on the data obtained from South African Breweries (SAB). A total of 198 training inputs were provided for each network architecture. Each element of the database was removed and approximated using the model. Cases of 1, 2, 3, 4, and 5 missing values in a single record were examined to investigate the accuracy of the approximated values as the number of missing cases within a single record increases. To asses the accuracy of the values approximated using the model the standard error and correlation coefficient explained in the previous chapter were calculated for each missing case.

Before going into the main experiment (approximation of missing values), first the trained network was examined, to asses how the trained network actually fits to the input data. Different input records were randomly selected from the 198 training records used in training the network. Since all randomly selected records can not be presented graphically, only one representative record is presented for each network (MLP and RBF).

Figures 6.1 and 6.2 illustrate the data and trained network of one record for both MLP and RBF, respectively. It can be observed that trained values using the network are similar to the actual data for both MLP and RBF networks.

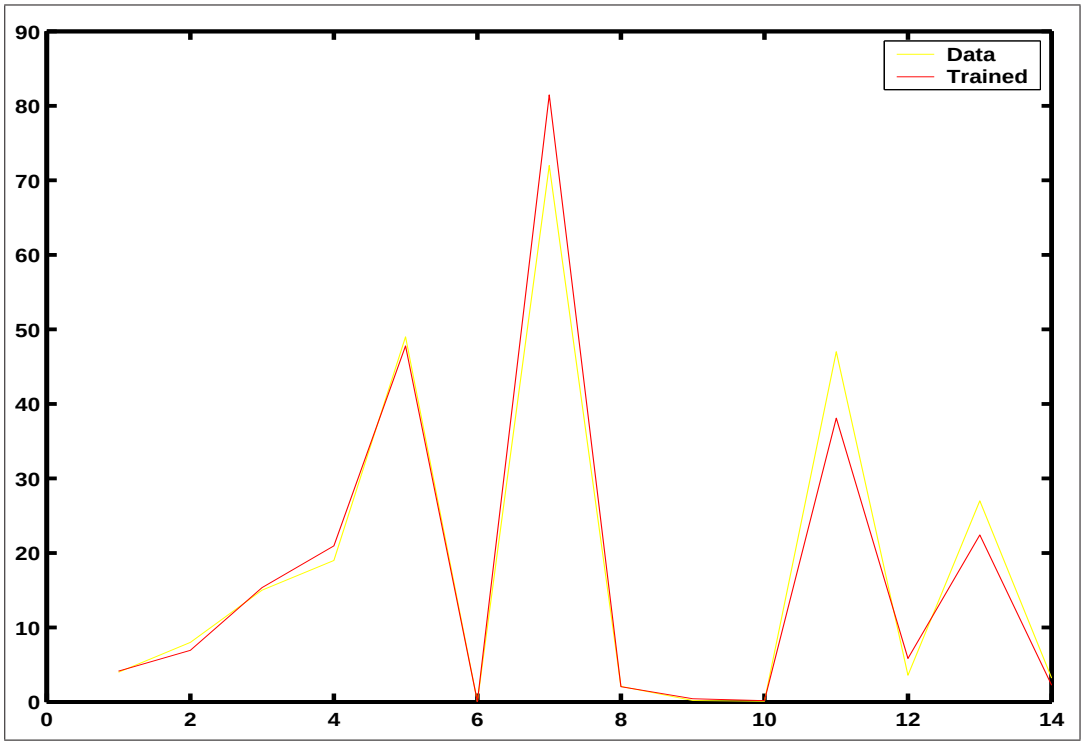


Figure 6.1: Data vs trained using MLP

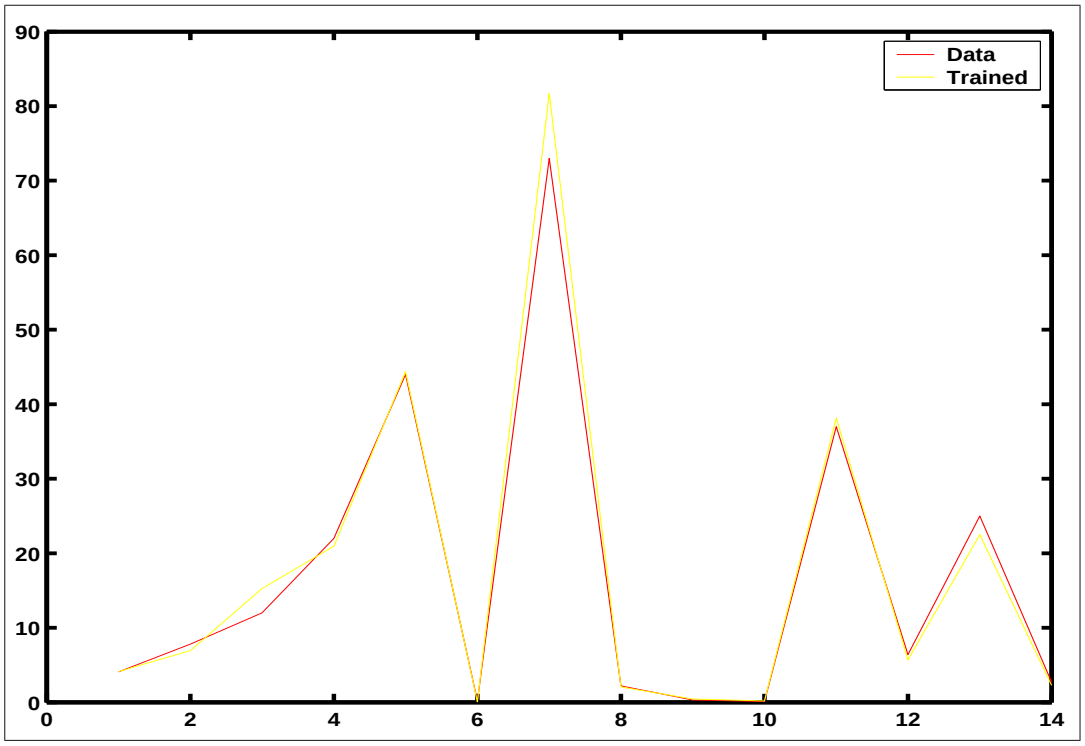


Figure 6.2: Data vs trained using RBF

6.3 Performance of genetic algorithm

The MATLAB implementation of genetic algorithm described in Houck *et al.* [1995] has been used to implement the genetic algorithm. The genetic algorithm was evaluated by altering genetic operator values (number of generation, mutation, crossover, etc.). The final genetic algorithm chosen in the experiment was the one that gave the best fitness illustrated in Figure 6.3. As depicted in Figure 6.3 the number of generation was selected to be 500. The number of generation was selected to be 500, because there was no any improvement in fitness observed after 500 generations for all the experiments done. Other operators were selected in a similar manner as the number of generation.

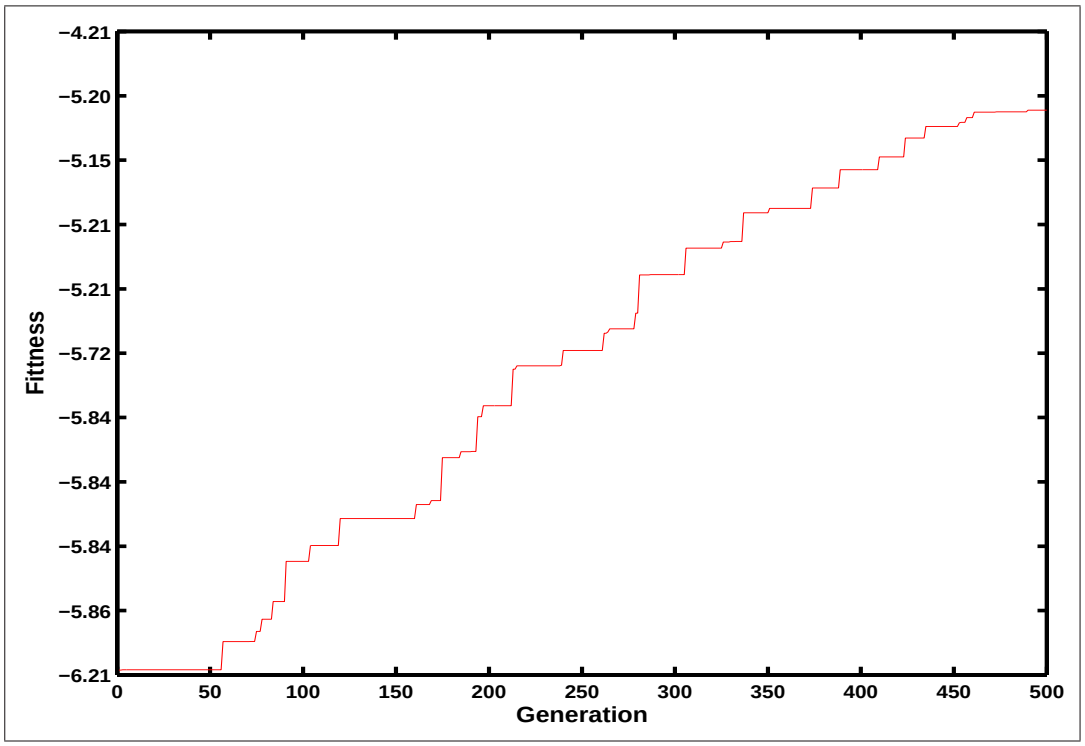


Figure 6.3: Performance of the genetic algorithm during the run

6.4 Missing data analysis

The result of the correlation and standard error measures obtained from the experiment are given in Table 6.1 and 6.2 respectively. The results are also depicted in Figure 6.4 and 6.5 for an easy comparison between the results found by MLP, RBF and MLP+RBF[1]. The results show that the models approximation to the missing data to be highly accurate. There seems to be less significant difference among the approximations obtained for the different number of missing cases within a single record.

The standard error (Se) estimates the capability of the model to predict the known data set, and the correlation coefficient (r) measures the degree of relationship between the actual missing data and corresponding approximated values using the model. It always ranges between -1 and 1. A positive value indicates a direct relationship between the actual missing data and its approximated value using the model. The more the value is closer to one the more the similarity between the two values. Approximations obtained using the combined model (MLP+RBF) in all the missing cases are better than the corresponding values found for MLP and RBF. Results found for RBF are significantly better than the ones found using MLP for all the missing cases.

A sample of the actual missing data and its approximated values using the model for the 14 variables used in the model are presented in Table 6.3 and 6.4, and Figure 6.4 and 6.5. The results show that the models

[1]Is the evaluation function which uses the linear combination of MLP and RBF

approximation of the missing data to be similar to the actual missing values. It can also be observed that the estimates found for 1, 2, 3, 4, and 5 missing cases are not significantly different from within each other.

Table 6.1: Correlation coefficient

	Number of Missing Value				
	1	2	3	4	5
MLP	0.94	0.939	0.939	0.933	0.938
RBF	0.968	0.969	0.970	0.970	0.968
MLP+RBF	0.96	0.97	0.97	0.97	0.96

Table 6.2: Standard error

	Number of Missing Value				
	1	2	3	4	5
MLP	16.62	16.77	16.8	16.31	16.4
RBF	11.89	11.92	11.80	11.92	12.02
MLP+RBF	12.02	11.78	11.36	11.23	12.60

Figure 6.6, 6.7, 6.8, 6.9, and 6.10 illustrate the actual missing values and the corresponding approximated values using the model for 1,2,3,4, and 5 missing cases respectively. The figures show that the approximated values in all the missing cases[2] are more accurate. It can also be observed that there is no significant difference in the approximated values for the 1,2,3,4, and 5 missing cases. It can also be observed that approximations found using the combined[3] is better than MLP and RBF. Approximations

[2]all missing cases refers to 1,2,3,4, and 5 missing cases examined in the research
[3]MLP+RBF

found using RBF are relatively higher than MLP. For further illustration the values depicted in Figure 6.6, 6.7, 6.8, 6.9 are also presented in Table 6.3, 6.4, and 6.5.

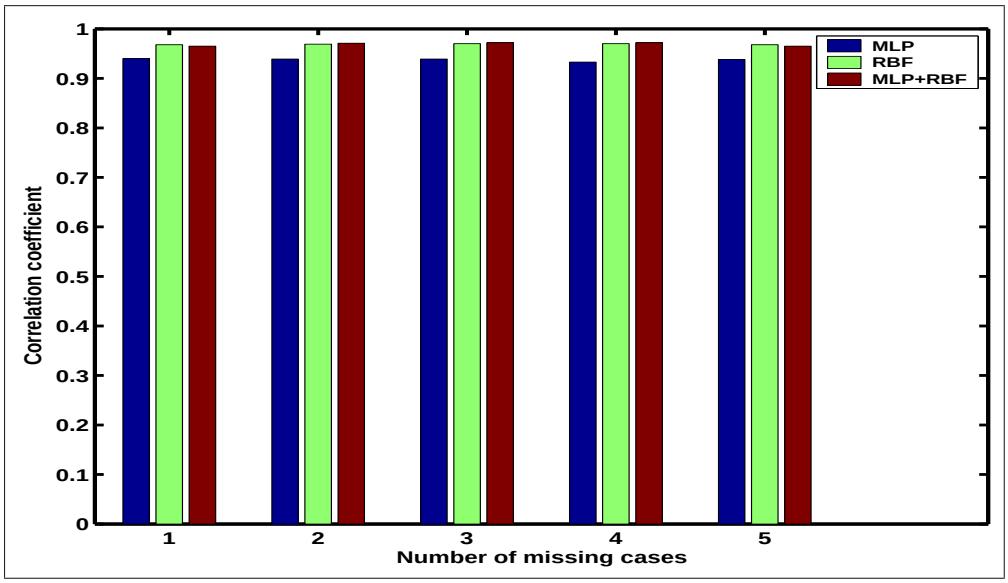


Figure 6.4: Correlation coefficient MLP, RBF, and MLP+RBF

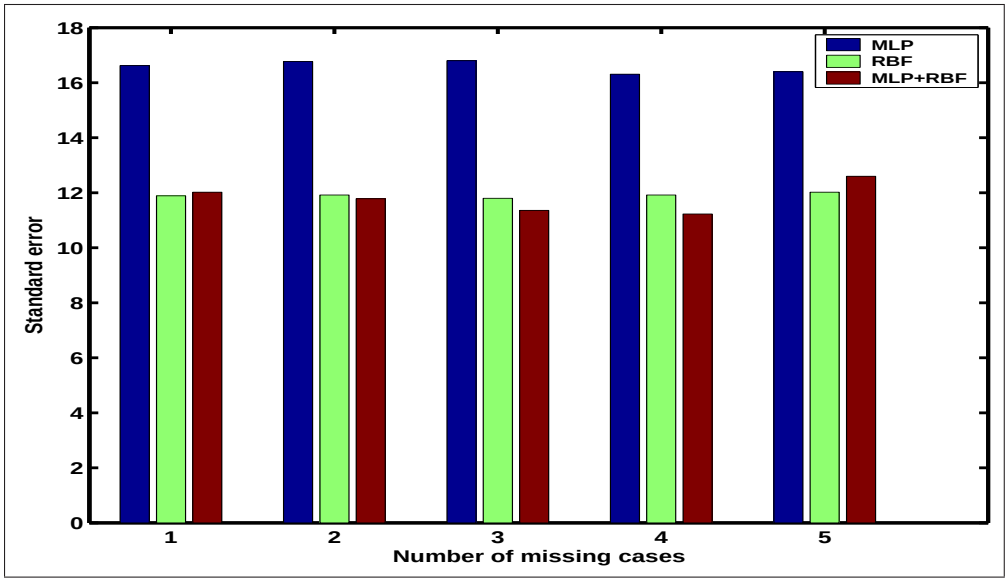


Figure 6.5: Standard error MLP, RBF, and MLP+RBF

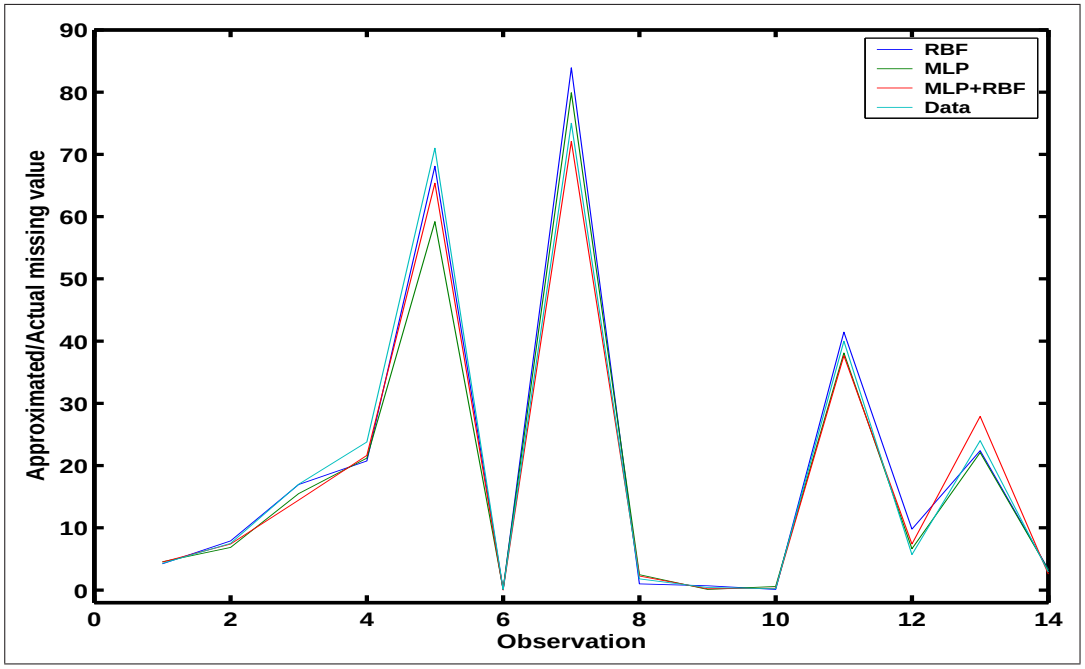


Figure 6.6: One missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF

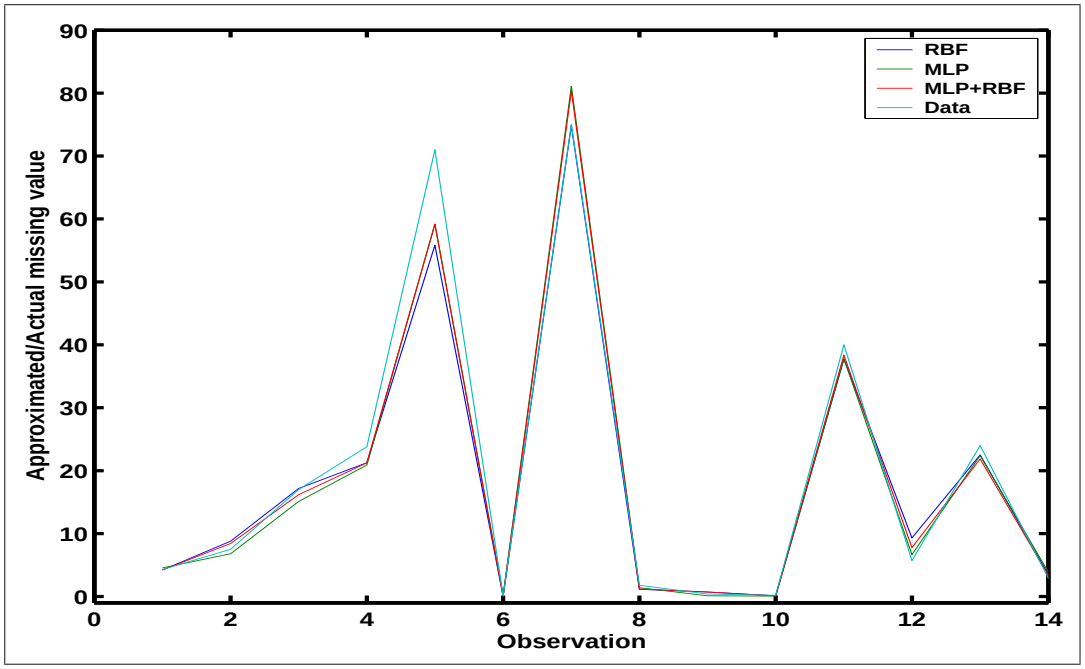


Figure 6.7: Two missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF

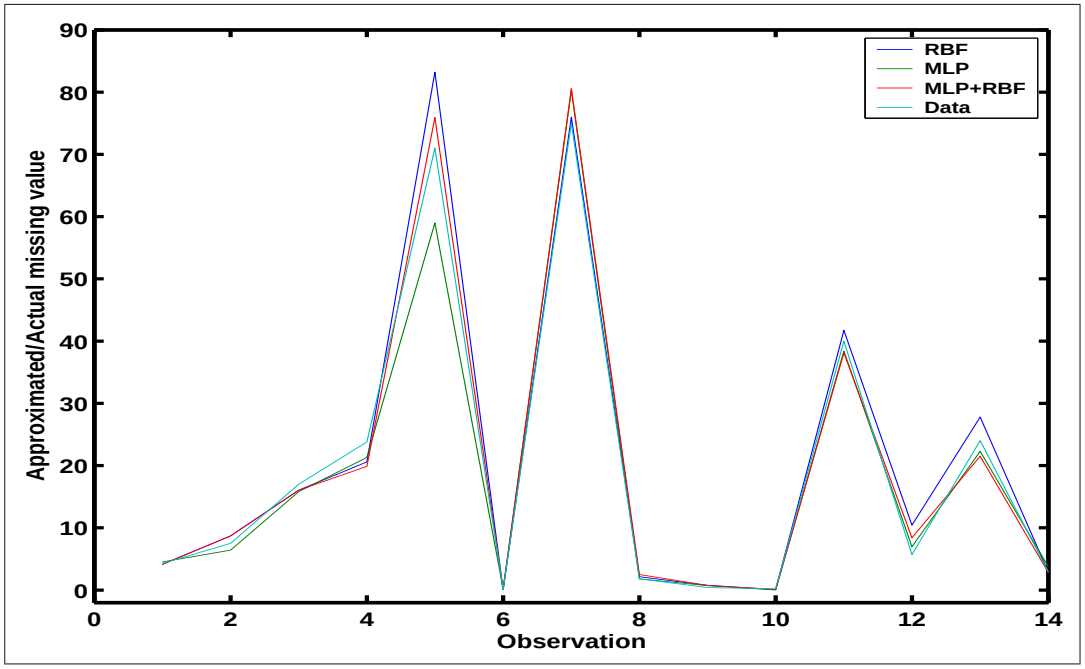


Figure 6.8: Three missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF

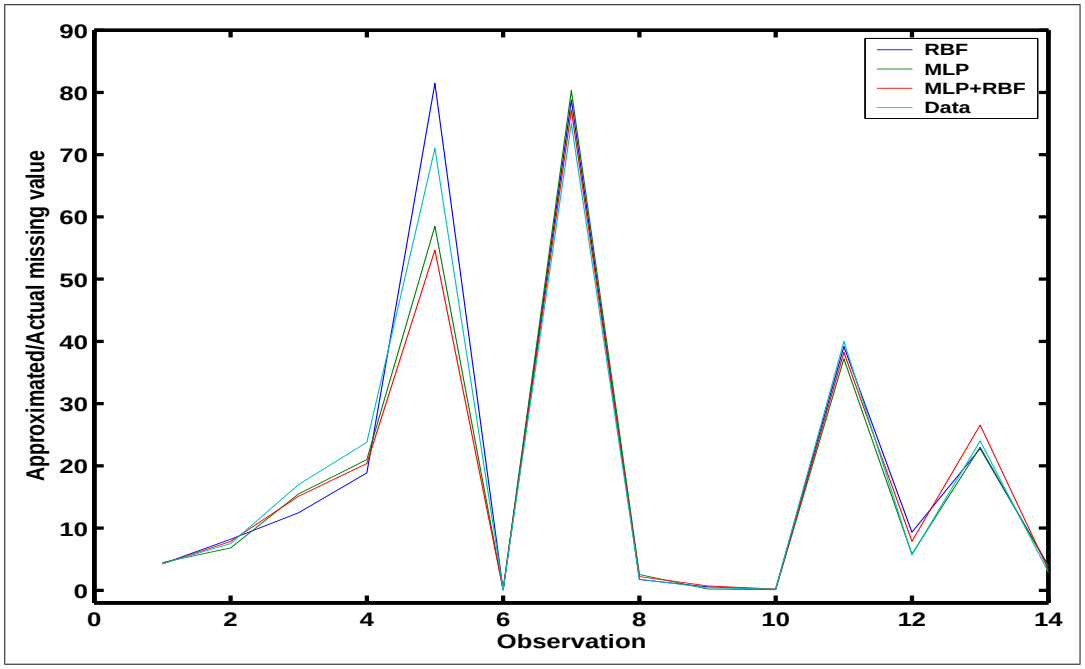


Figure 6.9: Four missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF

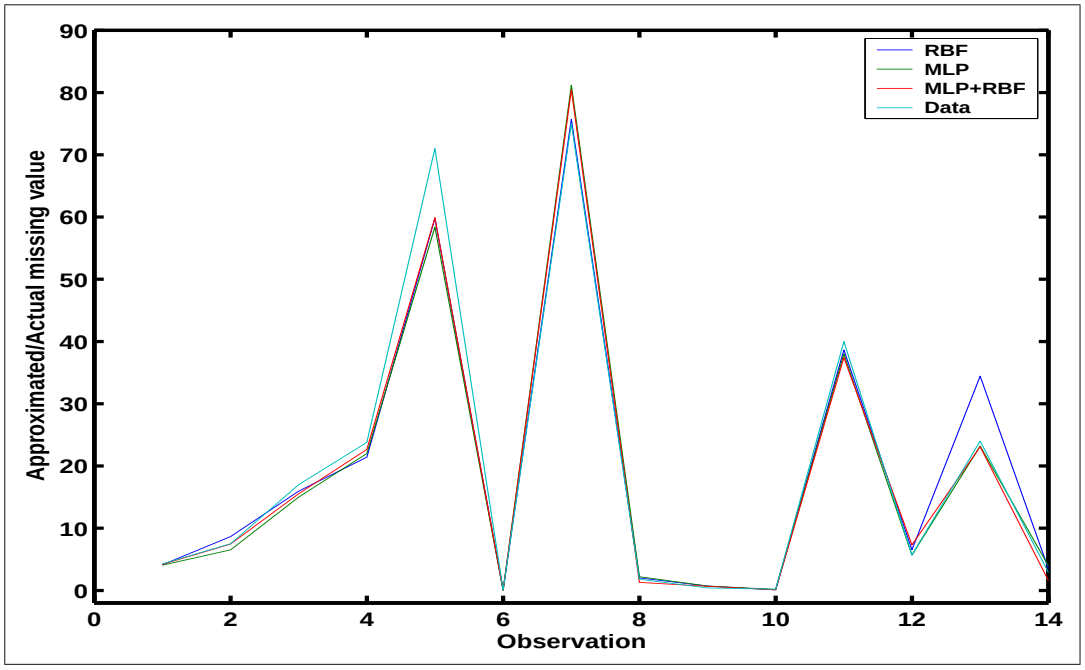


Figure 6.10: Five missing case: actual vs. approximated values using MLP, RBF, and MLP+RBF

Table 6.3: Actual and approximated values using MLP

	Number of missing cases in a record				
Data	1	2	3	4	5
4.28	4.54	4.54	4.53	4.47	4.07
7.5	6.86	6.79	6.41	6.80	6.52
17	15.50	15.10	15.8	15.5	15.0
23.8	21.20	20.90	21.3	21.0	22.0
71	59.20	59.20	59.0	58.5	58.4
0.1	0.18	0.17	0.17	0.05	0.02
75	79.90	81.1	80.3	80.3	81.2
1.8	2.48	2.41	1.81	2.54	2.21
0.4	0.10	0.104	0.72	0.22	0.72
0.2	0.58	0.06	0.02	0.11	0.159
40	38.10	37.8	38.4	37.2	38.0
5.7	6.64	6.66	6.96	5.82	5.67
24	22.10	22.4	22.3	23.0	23.2
2.9	3.23	3.86	3.74	3.83	3.97

Table 6.4: Actual and approximated values using RBF

	Number of missing cases in a record				
Data	1	2	3	4	5
4.28	4.21	4.20	4.12	4.25	4.13
7.5	7.89	8.79	8.71	8.21	8.65
17	16.96	17.16	16.04	12.48	15.95
23.8	20.74	21.25	20.60	18.88	21.43
71	68.11	55.83	83.21	81.46	59.78
0.1	0.06	0.04	0.05	0.05	0.08
75	83.92	74.84	75.96	78.79	75.70
1.8	1.00	1.14	2.15	1.73	2.01
0.4	0.70	0.71	0.76	0.55	0.71
0.2	0.10	0.10	0.09	0.16	0.11
40	56.45	57.73	61.73	62.16	62.65
5.7	9.79	9.30	10.43	9.33	6.54
24	22.40	22.52	27.81	36.79	34.45
2.9	3.31	3.48	2.87	3.98	3.50

Table 6.5: Actual and approximated values using MLP+RBF

	Number of missing cases in a record				
Data	1	2	3	4	5
4.28	4.50	4.20	4.12	4.24	4.20
7.5	7.40	8.42	8.71	7.85	7.47
17	14.46	16.17	16.04	15.13	15.53
23.8	21.63	21.24	19.88	20.38	22.67
71	65.38	59.16	75.92	54.66	59.89
0.1	0.01	0.04	0.06	0.07	0.03
75	72.09	80.37	80.56	77.16	80.44
1.8	2.26	1.20	2.50	2.21	1.30
0.4	0.20	0.71	0.75	0.72	0.67
0.2	0.30	0.11	0.11	0.21	0.10
40	37.66	38.36	38.13	38.23	37.43
5.7	7.40	7.78	8.40	7.87	7.34
24	27.91	21.87	21.51	26.55	23.09
2.9	2.56	3.35	2.81	3.33	1.61

6.5 Results and hypotheses testing

The results and analysis presented above are used to assess the hypotheses presented in the previous chapter.

The strong correlation coefficient and low standard error obtained for the missing values using the two networks and combined (MLP+RBF) leads to accepting the first hypothesis put forward in the previous chapter. This leads to the conclusion that the model proposed in this research can be used to approximate missing values in a database efficiently.

Since results obtained for the different missing cases were not significantly different to each other, as it is observed from the correlation coefficient and standard error as well as from the figures depicted to compare the approximated and actual missing values, this leads us to rejecting the second hypothesis proposed in the previous chapter. This means that there is no significant difference on accuracy and similarity observed as the number of missing cases within a single record increases. This result is contradictory to other results found using hot deck imputation, case wise analysis, etc in different researches and needs a complete further investigation.

Results found for different network architecture used in the research are significantly different. As it can be observed in Figure 6.4 and 6.5, correlation coefficients found using RBF are stronger than MLP and standard errors found for RBF are smaller than MLP. Correlation coefficients

found using MLP+RBF are stronger than MLP and slightly better than RBF. Standard error found for MLP+RBF is also smaller than MLP and slightly better than in the case of 1 and 5 missing cases in RBF. This shows that the specific neural network employed in the training of data set has a significant impact on the approximated values. This result can also be used to accept hypotheses four, which leads to a conclusion that using a combination of both architectures in the error function leads to better results.

6.6 Comparisons of results with other methods

Since there was no other research found employing the methodologies used in this research, it is really difficult to compare the results of this research with other results found using currently used missing data imputation algorithms. This is primarily due to the evaluation methods used in the research. Though this is the case, based on the correlation coefficient found in the research, it can be concluded with great confidence that the proposed method outperforms the old fashioned data imputation methods like listwise or casewise data deletion, pairwise data deletion, mean substitution, hot deck imputation, and regression methods. It is recommended that future works compare the advantages and disadvantages of the proposed method in relation with other missing data imputation algorithms. Further research is needed to exactly examine the effectiveness of each algorithm on the same database with similar

experimental methodology.

6.7 Chapter summary

This chapter attempted to give analysis and discussion to prove the hypotheses presented in the previous chapter. Results from the experiments revealed a high correlation coefficient between actual missing data and approximated values. This leads to the acceptance of hypothesis one. It is found that the specific architecture used in training the data set has a significant impact on the approximations. There was no significant reduction in accuracy observed as the number of missing cases in single record gets bigger.

Chapter 7

Conclusions and Further Research

This chapter begins by summarizing the research. It then provides suggestions for areas of further investigation. After examining the contributions of this research, final conclusions are given.

7.1 Conclusions

7.1.1 Summary of research

This research evaluates the effectiveness of using neural network and genetic algorithm to approximate missing data in database. Thus the research attempted to answer four basic questions:

- *To approximate or predict missing values in a database efficiently using a model employing an artificial neural networks and genetic algorithms.*

- *To investigate the relationship between the accuracy of approximated or predicted values using the proposed model and the number of missing cases/variables in single record.*
- *Asses the impact of using different neural network architecture employed in training the neural network on the approximated values.*
- *Asses accuracy of results found using individual networks as compared to combined approach.*

7.1.2 Summary of results and conclusions

Neural networks and genetic algorithms are proposed to predict missing data in a database. An auto-associative neural network is trained to predict its own input. An error function is derived as the square of the difference of the output vector from the trained neural network and the input vector. Since some of the input vectors are missing, the error function is expressed in terms of the known and unknown components of the input vector. Genetic algorithm is used to approximate the missing values in the input vector that best minimise the error function.

RBF and MLP neural networks are used to train the neural network. Moreover the combination of both RBF and MLP trained networks was employed to investigate if it could lead to a better results.

It is found that approximated values using the proposed model are highly accurate with over 95% correlation coefficient between the actual miss-

ing values and corresponding approximated values. It was observed that, though there is a slight decrease in correlation coefficient, there was no significant reduction in accuracy of results observed as the number of missing cases within a single record gets larger. It is also observed that results found using the combination of both RBF and MLP trained networks are superior than the one found using either RBF or MLP. Results found using RBF are found to be far better than MLP.

7.2 Future work

While the results presented in this report are quite promising and reliable, a number of avenues for future work exist that may greatly improve the effectiveness of this approach or which can utilize the approach proposed in this research to tackle other problems/applications.

7.2.1 Using different machine learning techniques

The research employed neural network as a learning method to train the data set. Though the choice for neural network was imminent from its merits compared with other machine learning techniques and has revealed good solutions, it is worthwhile trying to investigate/construct the model using other machine learning approaches. One of those approaches which could be used to tackle the same problem is to train the model using Support Vector Machine (SVM) instead of neural networks. Other than

SVM, other learning methods can also be used to train the data set.

7.2.2 Using different optimization methods

The research employed genetic algorithm to optimize the error function and approximate the missing values in the error function. Though the choice of genetic algorithm was due to its superiority as an optimization method compared to other optimization methods, it is worthwhile investigating the model by applying other optimization approaches. Some of the optimization approaches which could be used to tackle the same problem are simulated annealing, particle swarm optimization, and hill climbing.

7.2.3 Comparison with other models under the same data set

As it is presented in Chapter 4, currently there are various missing data imputation algorithms used under different situation. Each algorithms has its own advantages and disadvantages. Despite the promising results obtained using the proposed model in this research, one of the crucial investigations that have to be done is to compare the results found in this research with the other missing data imputation algorithms under the same data set with different database categories. The database categories could be

- **Qualitative:** where the database only consists of qualitative values.

- **Quantitative:** where the database only consists of quantitative values
- **Combination of both:** where the database consists of both qualitative and quantitative values.

The investigations could also be conducted on different data missing mechanism assumptions (MCAR, MAR, and non-ignorable case).

7.2.4 Forecasting and risk analysis

The research attempted to use the proposed model to approximate missing data in databases. Apart from this, the proposed model can be applied to forecasting and risk analysis applications. Table 7.1 shows a database ideal for using the proposed model on forecasting and risk analysis. In Table 7.1 the data for time 1, 2, and 3 are complete, but some of the values for time 4 are missing. If we consider time 1, 2, and 3 as past complete historical data and the known values for time 4 as known or expected values for time 4, we can utilize the proposed model in this research to forecast the values of x_3 and x_5 for time 4. Similarly, if we

Table 7.1: Proposed model on forecasting and risk analysis applications

Time	x_1	x_2	x_3	x_4	x_5
1	25	3.5	236	5000	-3.5
2	45	6.9	5.6	3600	0.5
3	45	3.6	9.5	1500	46.5
4	27	9.7	?	3000	?

consider that we have only the complete data for time 1,2, and 3, and would like to use risk analysis on the future values of variable x_1, x_2, x_3, x_4, and x_5 at time 4. We can fix the expected values of some variables (which could be done on applications like stock market prediction, exchange rate determination, and scientific stochastic processes) and asses the likelihood of other variables.

References

[Alfonseca 1991] Manuel Alfonseca. Genetic algorithms. In *Proceedings of the international conference on APL*, pages 1–6. ACM Press, 1991.

[Allison 2000] P.D. Allison. Multiple imputation for missing data: A cautionary tale. In *Sociological Methods and Research*, volume 28, pages 301–309, 2000.

[Allison 2002] P. Allison. Missing data, 2002. Thousand Oaks, CA: Sage.

[Back *et al.* 1992] T Back, F. Hoffmeister, and H. Schwefel. Applications of evolutinary algorithms, 1992.

[Banzhaf *et al.* 1998] W. Banzhaf, P. Nordin, R. Keller, and F. Francone. *Genetic Programming-an introduction: on the automatic evolution of computer programs and its applications.* Morgan Kaufmann Publishers, California, fifth edition, 1998.

[Bishop 1995] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[Draper and Smith 1998] Norman Draper and Harry Smith. *Applied regression analysis*. J. Wiley, New York, third edition, 1998.

[Forrest 1996] Stephanie Forrest. Genetic algorithms. *ACM Comput. Surv.*, 28(1):77–80, 1996.

[Freeman and Skapura 1991] James Freeman and David Skapura. *Neural Networks: Algorithms, Applications and Programming Techniques*. Addison-Wesley, 1991.

[Goldberg 1989] David Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[Hassoun 1995] Mohamad H. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, Cambridge, Massachusetts, 1995.

[Haykin 1999] Simon Haykin. *Neural Networks*. Prentice-Hall, New Jersey, second edition, 1999.

[Houck *et al.* 1995] Christopher R. Houck, Jeffery A. Joines, and Michael G. Kay. A genetic algorithm for function optimisation: a matlab implementation. Technical Report NCSU-IE TR 95-09, Carolina State University, 1995.

[Hu *et al.* 1998] M. Hu, S. Savucci, and M. Choen. Evaluation of some popular imputation algorithms. In *Proceedings of the Survey Research Methods Section of the American Statistical Association*, pages 308–313, 1998.

[Jones and Konstam 1999] Michael Jones and Aaron Konstam. The use of genetic algorithms and neural networks to investigate the baldwin effect. In *Proceedings of the 1999 ACM symposium on Applied computing*, pages 275–279. ACM Press, 1999.

[Kolarik and Rudorfer 1994] Thomas Kolarik and Gottfried Rudorfer. Time series forecasting using neural networks. In *Proceedings of the international conference on APL : the language and its applications*, pages 86–94. ACM Press, 1994.

[Little and Rubin 1987] R. Little and D. Rubin. *Statistical analysis with missing data*. John Wiley and Sons, New York, first edition, 1987.

[Michalewicz 1996] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berling Heidelberg, New York, third edition, 1996.

[Nabney 2001] Ian T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer-Verlag, United Kingdom, 2001.

[Pendharkar and Rodger 1999] Parag C. Pendharkar and James A. Rodger. An empirical study of non-binary genetic algorithm-based neural approaches for classification. In *Proceeding of the 20th international conference on Information Systems*, pages 155–165. Association for Information Systems, 1999.

[Roth 1994] P. Roth. Missing data:a conceptual overview for applied

psychologists. In *Personnel Psychology*, volume 47, pages 537–560, 1994.

[Rubin 1978] D. B. Rubin. Multiple imputations in sample surveys - a phenomenological bayesian approach to nonresponse. In *The Proceedings of the Survey Research Methods Section of the American Statistical Association*, pages 20–34, 1978.

[Scheffer 2000] Judi Scheffer. Dealing with missing data, 2000. I.I.M.S Quad A, Massey University, Aukland. http://www.massey.ac.nz/ wwiims/research/letters.

[Yansaneh *et al.* 1998] I. S. Yansaneh, L. s. Wallace, and D. A. Marker. Imputation methods for large complex datasets: An application to the nehis. In *Proceedings of the Survey Research Methods Section, American Statistical Association*, pages 314–319, 1998.

[Yoon and Peterson 1990] Youngohc Yoon and Lynn L. Peterson. Artificial neural networks: an emerging new technique. In *Proceedings of the 1990 ACM SIGBDP conference on Trends and directions in expert systems*, pages 417–422. ACM Press, 1990.

[Yuan 2000] Yang Yuan. Multiple imputation for missing data: Concepts and new development. In *SUGI Paper 267-25*, 2000.

Appendix 1

Conference Paper

During the project, selected topics of the work were used to write conference papers. In particular the new proposed model and its reliability in approximating missing data.

The papers have been accepted to be published in the IEEE 3^{rd} International Conference on Computational Cybernetics, April 13-16, 2005, Mauritius, and International Joint Conference on Neural Networks, July 31-August 4, 2005, Montreal, Canada. See below for the full references.

Mussa Abdella and Tshilidzi Marwala. **The Use of Genetic Algorithms and Neural Networks to Approximate Missing Data in Database**. *IEEE 3^{rd} International Conference on Computational Cybernetics*, Mauritius, 2005.

Mussa Abdella and Tshilidzi Marwala. **Treatment of Missing Data Using Neural Networks and Genetic Algorithms**. *International Joint Conference on Neural Networks*, Montreal, Canada, 2005.