The Use of Genetic Algorithms and Neural Networks to Approximate Missing Data in Database

Mussa Abdella
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg, South Africa
m.abdella@ee.wits.ac.za

Tshilidzi Marwala
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg, South Africa
t.marwala@ee.wits.ac.za

Abstract—**Missing data creates various problems in analysing and processing data in databases. In this paper we introduce 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. Multi-Layer Perceptron (MLP) and Radial Basis Function (RBF) networks are employed to train the neural networks. Our focus also lies on the investigation of using the proposed method in accurately predicting missing data as the number of missing cases within a single record increases. It is observed that there is no significant reduction in accuracy of results as the number of missing cases in a single record increases. It is also found that results obtained using RBF are superior to MLP.**

I. Introduction

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. Moreover 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. Such situations may arise in a system which uses a number of instruments and in some cases one or more of the sensors used in the system fail. In such situation the value of the sensor have to be estimated within short time and with great precision and by taking in to account the values of the other sensors in the system. Approximation of the missing values in such situations require us to estimate the missing value taking into account the interrelationships that exists between the values of other corresponding variables.

Missing data in a database may arise due to various reasons. It can arise due to data entry errors, respondents non response to some items on data collection process, failure of instruments and other various reasons. In Table I we have a database consisting five variables namely x_1, x_2, x_3, x_4, and x_5 where the values for some variables are missing. Assume we have a database consisting of various records of the five variables. But some of the observations for some variables in various records are not available. How do we know the values for the missing entries? Are there ways to approximate the missing data depending on the interrelationships that exist between the variables in the database? Thus, the aim of this paper is to use neural networks and genetic algorithms to approximate the missing data in such situations.

TABLE I
Table with missing values

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

II. Background

A. 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 [1][2][3]. The reasonable way to handel missing data depends upon how data points become missing. According to [4] there are three types of missing data mechanisms. These 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 [3]. 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 [2]. Depending on the mechanism of missing data, currently various methods are being used to treat missing data. For a detailed discussion on the various methods used to handle missing data refer to [3][2][4] and [5]. The method proposed in this paper is applicable to situations where the missing data mechanism is either MCAR, MAR or non-ignorable.

B. Neural Networks

A neural network is an information processing paradigm that is inspired by the way biological nervous systems, like the brain process information [6]. It is a machine that is designed to model the way in which the brain performs a particular task or function of interest [7].

A neural network consists of four main parts [7]. 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.

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. A trained neural network can be considered as an expert in the category of information it has been given to analyse [6]. 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 [7][8].

The arrangement of neural processing units and their interconnections can have a profound impact on the processing capabilities of a neural network [7]. Hence, there are many different connection of how the data flows between the input, hidden and output layers. The following section details the architecture of the two neural networks employed in this paper.

1) Multi-Layer Perceptrons (MLP): MLP neural networks consist of multiple layers of computational units, usually interconnected in a feed-forward way [7][8]. Each neuron in one layer is directly connected to the neurons of the subsequent layer. A fully connected two layered MLP architecture was used in the experiment. A NETLAB toolbox that runs in MATLAB discussed in [9] has been used to implement the MLP neural network. A two-layered MLP architecture was used because of the universal approximation theorem, which states that a two layered architecture is adequate for MLP [9]. Fig. 1 depicts the architecture of the MLP used in this paper. We have 14 inputs, 2 hidden layers with 10 neurons and 14 output units.

MLP networks apply different learning techniques, the most popular being back-propagation [7]. In back-propagation the output values are compared with the correct answer to compute the value of some predefined error-function. 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 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 [7].

2) Radial-Basis Function (RBF): RBF networks are feed-forward networks trained using a supervised training algorithm [7]. They are typically configured with a single hidden layer

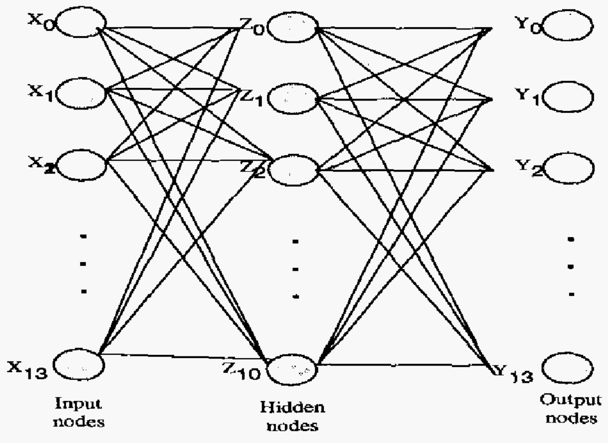

Fig. 1. MLP architecture

of units whose activation function is selected from a class of functions called basis functions. While similar to back propagation in many aspects, radial basis function networks have several advantages. They usually train much faster than back propagation networks and less prone to problems with non-stationary inputs due to the behavior of the radial basis function [10]. Like the MLP a NETLAB toolbox that runs in MATLAB discussed in [9] was used to implement the RBF architecture. The network has 14 inputs, 10 neurons and 14 output units. Fig. 2 depicts the architecture of the RBF network used in this paper. Z_i's in Fig. 2 represent the non-linear activation functions.

C. Genetic Algorithms

Genetic Algorithms (GAs) are algorithms used to find approximate solutions to difficult problems through application of the principles of evolutionary biology to computer science [11][12]. They use biologically derived techniques such as inheritance, mutation, natural selection, and recombination to approximate an optimal solution to difficult problems [13][14].

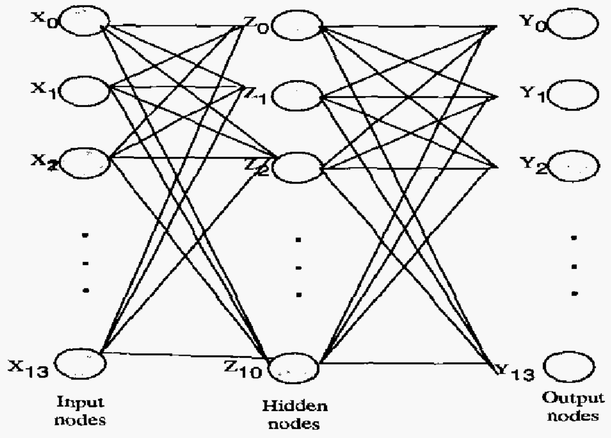

Fig. 2. RBF architecture

Genetic algorithms view learning as a competition among a population of evolving candidate problem solutions. A fitness function evaluates each solution to decide whether it will contribute to the next generation of solutions. Through operations analogous to gene transfer in sexual reproduction, the algorithm creates a new population of candidate solutions [14].

The three most important aspects of using genetic algorithms are [11][15]:

- Definition of the objective function.
- Definition and implementation of the genetic representation, and
- Definition and implementation of the genetic operators.

GAs have been proved to be successful in optimization problems such as wire routing, scheduling, adaptive control, game playing, cognitive modeling, transportation problems, traveling salesman problems, optimal control problems, and database query optimization [11].

The following pseudo-code from [11] illustrates the high level description of the genetic algorithm employed in the experiment. $P(t)$ represents the population at generation t.

```
procedure genetic algorithm
  begin
    $t \leftarrow 0$
    initialise $P(t)$
    evaluate $P(t)$
    while(not termination condition) do
    begin
      $t \leftarrow 0$
      select $P(t)$ from $P(t-1)$
      alter $P(t)$
      evaluate $P(t)$
    end
  end
```

Algorithm 1: Structure of genetic algorithm [11]

The MATLAB implementation of genetic algorithm described in [15] has been used to implement the genetic algorithm. After executing the program with different genetic operators, optimal operators that gave the best results were selected to be used in conducting the experiment.

III. Method

The neural network was trained to recall to itself (predict its input vector). Mathematically the neural network can be written as

$$\vec{Y} = f(\vec{X}, \vec{W}) \tag{1}$$

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 vector. Thus, the error can be formulated as

$$e = \vec{X} - \vec{Y} \tag{2}$$

Substituting the value of $\vec{Y}$ from (1) into (2) we get

$$e = \vec{X} - f(\vec{X}, \vec{W}) \tag{3}$$

We want the error to be minimum and non-negative. Hence, the error function can be rewritten as the square of equation (3)

$$e = (\vec{X} - f(\vec{X}, \vec{W}))^2 \tag{4}$$

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 in to $\vec{X}$ known represented by ($\vec{X}_k$) and $\vec{X}$ unknown represented by ($\vec{X}_u$). Rewriting equation (4) in terms of $\vec{X}_k$ and $\vec{X}_u$ we have

$$e = \left(\left\{ \begin{array}{c} \vec{X}_k \\ \vec{X}_u \end{array} \right\} - f\left(\left\{ \begin{array}{c} \vec{X}_k \\ \vec{X}_u \end{array} \right\}, \vec{W} \right) \right)^2 \tag{5}$$

To approximate the missing input values, equation (5) is minimized using genetic algorithm. Genetic algorithm was chosen because it finds the global optimum solution. Since a genetic algorithm always finds the maximum value, the negative of equation (5) was supplied to the GA as a fitness function. Thus, the final error function minimized using the genetic algorithm is

$$e = -\left(\left\{ \begin{array}{c} \vec{X}_k \\ \vec{X}_u \end{array} \right\} - f\left(\left\{ \begin{array}{c} \vec{X}_k \\ \vec{X}_u \end{array} \right\}, \vec{W} \right) \right)^2 \tag{6}$$

Fig. 3 depicts the graphical representation of proposed model. The error function is derived from the input and output vector obtained from the trained neural network. The error function is then minimized using genetic algorithm to approximate the missing variables in the error function.

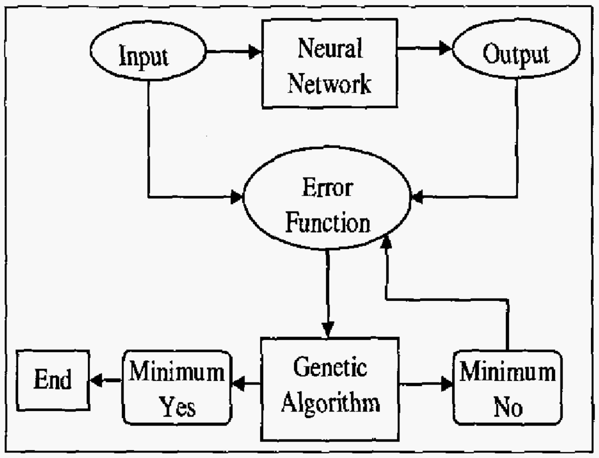

Fig. 3. Schematic representation of proposed model

IV. Result and Discussion

An MLP and RBF with 10 neurons, 14 inputs and 14 outputs was 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 were calculated for each missing case.

We have used the following terms to measure the modeling quality: (i) Standard error (Se) and (ii) Correlation coefficient (r). 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

$$\text{Se} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}{n}} \tag{7}$$

and the correlation coefficient (r) is computed as

$$\text{r} = \frac{\sum_{i=1}^{n} (x_i - \bar{x}_i)(\hat{x}_i - \bar{\hat{x}}_i)}{\left[\sum_{i=1}^{n} (x_i - \bar{x}_i)^2 \sum_{i=1}^{n} (\hat{x}_i - \bar{\hat{x}}_i)^2\right]^{1/2}} \tag{8}$$

The 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 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 result of the correlation and standard error measures obtained from the experiment are given in Table II and III respectively. The results are also depicted in Fig. 4 and 5 for easy comparison between the results found by MLP and RBF. 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.

Approximations obtained using RBF in all the missing cases are better than the corresponding values found using MLP. 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 IV and V, and Fig. 6 and 7. The results show that the models approximated value of the missing data to be similar to the actual 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 II
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

TABLE III
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

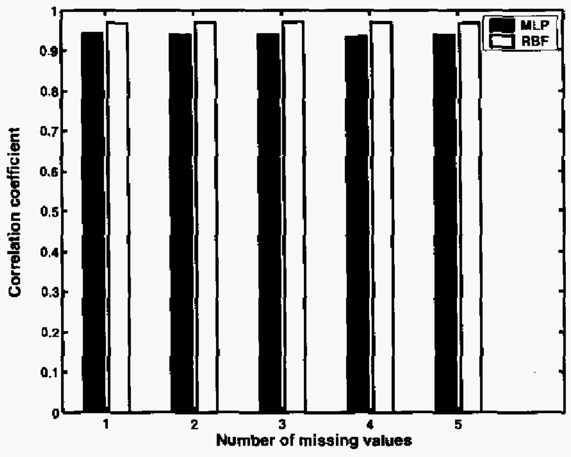

Fig. 4. Correlation coefficient MLP vs RBF

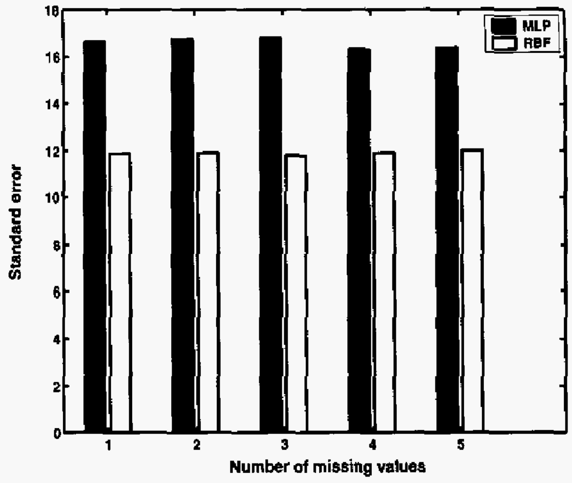

Fig. 5. Standard error MLP vs RBF

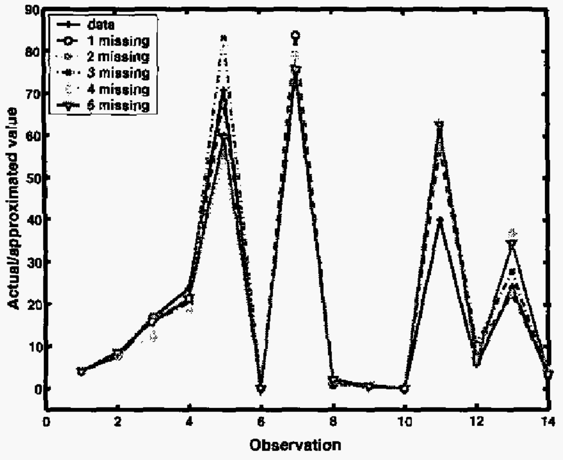

Fig. 6. Actual vs. approximated values using RBF

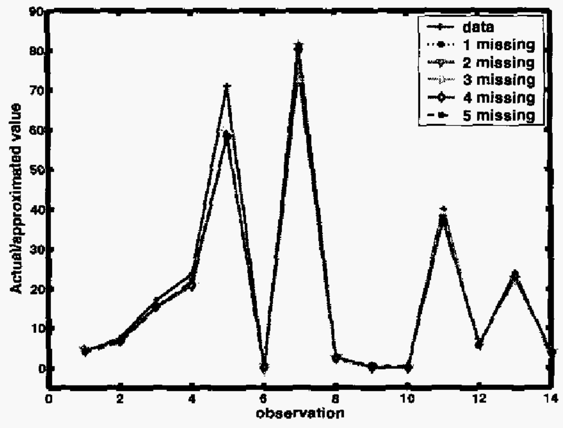

Fig. 7. Actual vs. approximated values using MLP

TABLE IV
ACTUAL AND APPROXIMATED VALUES USING MLP

Data	Number of missing values in a record				
	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 V
ACTUAL AND APPROXIMATED VALUES USING RBF

Data	Number of missing values in a record				
	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

V. CONCLUSION

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 was 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. It is found that the model approximates the missing values with higher accuracy and there was no significant reduction in accuracy as the number of missing data within a single record increases. It is also observed that results found using RBF are better than MLP.

REFERENCES

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

[2] P. Allison. "Multiple imputation for missing data: A cautionary tale." In *Sociological Methods and Research*, vol. 28, pp. 301–309. 2000.

[3] 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*, pp. 20–34. 1978.

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

[5] 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*, pp. 308–313. 1998.

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

[7] S. Haykin. *Neural Networks*. New Jersey: Prentice-Hall, second ed., 1999.

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

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

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

[11] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Berling Heidelberg, NY: Springer-Verlag, third ed., 1996.

[12] S. Forrest. "Genetic algorithms." *ACM Comput. Surv.*, vol. 28, no. 1, pp. 77–80, 1996.

[13] P. C. Pendharkar and J. 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*, pp. 155–165. Association for Information Systems, 1999.

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

[15] C. R. Houck, J. A. Joines, and M. G. Kay. "A genetic algorithm for function optimisation: a matlab implementation.", 1995. Carolina State University. http://www.ie.ncsu.edu/mirage/GAToolBox /gaot/.