Neural Network Based Techniques for Estimating Missing Data in Databases

Shakir Mohamed, Tshilidzi Marwala

School of Electrical and Information Engineering
University of the Witwatersrand, Johannesburg
shakir@metroweb.co.za, t.marwala@ee.wits.ac.za

Abstract

Three techniques based on the application of neural networks to the estimation of missing data in a medical database are introduced. The approximation unit is introduced as a basic missing data estimation mechanism and is extended to an agent–based system, which provides improved accuracy. A view of the estimation of missing data from a classification perspective is also presented. The neural networks designed implement a multi–layer perceptron architecture with weight decay. The optimisation of the regularisation coefficient using the evidence framework is shown to improve accuracy in the estimation. Analysis of the results show that all methods proposed are comparable providing the same level of accuracy while maintaining the statistical properties of the original data set. The agent-based system is found to provide a better accuracy on average. Results also suggest that the method of approximation units can be used to classify the missing data into the categories of MAR or MNAR.

Keywords: Auto–Associative, Data Estimation, Genetic Algorithms, Missing Data, Neural Networks.

1. Introduction

Databases such as those which store measurement or medical data, may become subject to missing values either in the data acquisition or data-storage process. A problem in a sensor, a break in the data transmission line or non-response to questions posed in a questionnaire are prime examples. This missing data poses a problem to the analysis and decision making processes which depend on this data, requiring methods of estimation which are accurate and efficient. Various techniques exist as a solution to this problem, but in many cases are not applicable because they introduce biases and noise into the data or make assumptions about the data which may not be true.

This paper compares three techniques of estimating missing data in a database using artificial neural networks. Neural networks are ideal for this type of problem because they are able to represent complex decision spaces and model complex relationships between data. The *approximation unit* is introduced as a basic missing data estimation mechanism and consists of a neural network and a genetic algorithm. An extension of this system to a committee of agents is implemented as the second technique. The third technique views the missing data problem from the perspective of pattern classification, where known values are used in a classification system whose output is a missing value in the database.

The design methodology employed will be described providing a justification for the methods and architectures which have been applied to each of the estimation techniques. The application of each of these techniques is then shown using data from the South African Department of Health. Comparisons of accuracy, change in statistical properties and the effect of various optimisation techniques for each of these methods is made and conclusions drawn therefrom.

2. Missing Data and Imputation Methods

Three classes of missing data have been identified, according to Little and Rubin [1]. These three categories can be understood by considering figure 1, which shows a data pattern with variables **X** = $\{X_1, \ldots, X_p\}$, and **Y** which has some missing data.

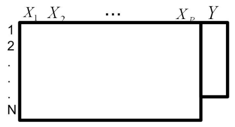


Figure 1: Pattern of non-response in a rectangular data set. Rows correspond to records in the database and the columns correspond to variables or fields of the data set [2].

X and **Y** are columns of a table in the database, with the column identified by **Y** having missing data. Considering **X** and **Y** as random variables, the three categories of missing data are:

- **Missing at Random (MAR).** Also known as the ignorable non-response. The probability of data **Y**, being missing is dependant only on **X**, the existing values in the database and not on any missing data.
- **Missing Not at Random (MNAR).** Also known as the non-ignorable case. The probability that **Y** is missing is dependant on the missing data.
- **Missing Completely at Random (MCAR)**. The probability of data **Y**, being missing is not dependant on either **X** or **Y**, i.e. is not dependant on either missing or complete values in the same record or any other record in the database.

The most prominent and useful techniques for the MAR case are multiple imputation (MI) and maximum likelihood (ML) [2]. Methods applicable to MCAR and MNAR cases have also been developed and have been reviewed by Schafer and Graham [2]. The estimation mechanisms presented in this paper assume that the data is MAR. Under this assumption, it is valid to use the neural network models of values in a database for data estimation, since missing data will be related to the known values in the database.

3. Background

3.1. Neural Networks and Genetic Algorithms

The simplest of the neural network architectures available are the Generalised Linear Models (GLM) [3]. These are single layer networks which consist of an input and an output layer and implement well known statistical techniques such as linear regression. The more common neural network architecture is the multilayer perceptron (MLP) [4]. The multi-layer perceptron network is a two–layer feed-forward network and in this application is trained using a supervised learning algorithm.

Genetic algorithms (GA) find approximate solutions to problems by applying the principles of evolutionary biology, such as crossover, mutation, reproduction and natural selection [5]. The GA search process consists of the following steps: 1) Generating a pool of candidate solutions (chromosomes) and encoding all values in a binary or floating point representation. 2) Evaluation of the fitness for each chromosome in the gene pool. The fitness is determined via a fitness function defined for the problem being solved, and chromosomes with the lowest fitness are discarded and make way for a new set of chromosomes. Replacement sets of chromosomes are created by the genetic operations of crossover and mutation on the most fit individuals. These genetic operations add an element of randomness to the search process allowing a wider range of the solution space to be explored. 3) Steps 1 and 2 are repeated for a given number of generations until a specified fitness level is attained or the maximum number of generations is exceeded [5].

3.2. Existing Techniques

The approximation unit presented in this paper takes the form of the system described by Abdella [6]. This work describes the estimation of missing data using a neural network and a genetic algorithm. The work explores the use of both MLP and RBF neural networks but the major focus lies in using this approach to investigate the estimation ability of the system as the number of missing cases in a single record increases.

Qiao *et. al.* also proposes a missing data estimator based on an auto-encoder neural network coupled with a particle swarm optimisation [7] and is used in tracking the dynamics of a plant in the presence of missing sensor measurements. Both techniques mentioned show success in estimating missing data using the same fundamental approximation structure. These structures are described, extended and contrasted in this paper showing alternative approaches to the estimation of missing data.

3.3. Data Analysis and Preprocessing

Data from the South African Department of Health is used in this investigation and consists of seven columns of data for fields such as HIV status, age, age group and gravidity (number of pregnancies). Using the maximum and minimum values of each column, the available data is normalised to the range of [0,1]. This is necessary to ensure that the values do not have wide ranges and aid in the network training by allowing better generalisation performance and in the estimation of missing values when using the genetic algorithm due to reduced searching range.

The values in the database do not have a large range and thus a linear normalisation is used. The formula used for normalisation for the i*th* column in the database with values given by *x* is:

$$x_{i,norm} = \frac{x_i - x_{i,min}}{x_{i,max} - x_{i,min}} \tag{1}$$

The data set available consists of 5776 records and is split into three parts with training and validation data sets consisting of 2500 entries each and the test set consisting of 776 entries. It must be noted that the network model is dependant on quality of the information used in the training set. Since patients tend to give false information under varying circumstances, the network model might not be completely accurate. This property of the data must be considered when analysing the effectiveness of the estimation techniques.

3.4. Performance Analysis

The effectiveness of the estimation system is evaluated using three performance measures: the mean square error, the correlation coefficient and the relative accuracy. Apart from these measures, the mean and the variance of the estimated data compared to that of the original data is used to evaluate how the estimation alters the statistical properties of the data. The mean square error is defined as

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i})^2 \tag{2}$$

where *N* is the number of estimates for which the error is being evaluated, x_i is the actual value and $\hat{x_i}$ is the predicted value produced by the estimation system for the i*th* sample. This measure must be interpreted in comparison with other values to determine the error performance. Smaller errors are preferred at all times over larger errors, since they indicate that the predicted and actual values are close.

The correlation coefficient *r*, indicates the strength of the relationship between the predicted and actual values x_i and $\hat{x_i}$, and measures the degree to which the two variables move together. If the magnitude of *r* is greater than 0.5, then there is a reasonable correlation between x_i and $\hat{x_i}$. If the value is less than 0.5 then the correlation value is less reliable and indicates poor estimation performance.

The final measure is the accuracy of the estimation as a percentage within a given tolerance. The relative accuracy is given by

$$A = \frac{n_\tau}{N} \times 100\% \tag{3}$$

where n_τ is the number of predictions in the sample set of size *N* that lie within the tolerance region τ. In this paper a tolerance of $\tau = 10\%$ is used.

4. Approximation Unit Design

4.1. System Overview

The approximation unit is a complete missing data estimation system, and is based on a trained auto-associative neural network and a genetic algorithm. The genetic algorithm drives the search process by generating a candidate solution for a missing data value in the database. The fitness of this solution is evaluated using a fitness function and a new solution is determined by the genetic algorithm according to the process discussed in section 3.1. This process continues until a globally optimal estimate for the missing data is found. The form of this fitness function is discussed in section 4.4.

The fitness function used is the difference between the inputs and outputs of the neural network. This network is trained to capture the relationships between fields and records in the database and the correlations between them. The fittest solution is one in which the difference between the networks inputs and outputs is minimum, indicating the the value estimated is consistent with the data used to train the network. A schematic representation of the system is shown in figure 2. As shown, the system is automatic, being driven by the GA, which generates an initial set of estimates and operates directly on the database with missing values.

4.2. Network Architecture

As stated, the neural network is an auto–associative or auto–encoder network. These networks consist of the same number

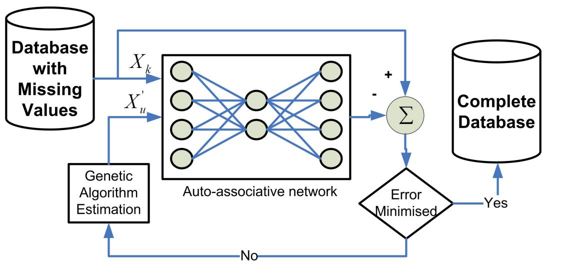


Figure 2: An approximation unit showing the place of the genetic algorithm and the neural network in the system. X_u' is an estimate for the missing values in the database produced by the GA.

of inputs and outputs and training involves training the network to replicate its inputs at the outputs. The number of hidden layer nodes in these networks are of a dimension less than that of the input layer [8], creating a "bottleneck" in the feed forward path of the network, allowing the network to capture the correlations between the data at the input nodes [7]. The error between the input and output of the network is low for normal data — data similar to that of the training data, but varies considerably for data that is un-similar to the training set. This property of the auto-associative neural network, is what is used to ensure that an optimal estimate for the missing data is found.

Two–layer MLP networks are used with linear activation functions for the output layer. The activation function for the hidden layer in the network used is the hyperbolic tangent function, $tanh(x)$. The optimal number of nodes has been chosen by training networks with the number of hidden layer nodes varying from 2 to 6. The networks were trained with the same training data set, and the network with the lowest error on a separate validation data set was chosen as the optimal network. The optimal network structure determined consists of 6 nodes in the hidden layer using 600 training epochs. This exhaustive search approach to determining the ideal number of hidden units is feasible in this instance since the range of the number of possible hidden layer units is small.

4.3. Network Training

The neural network is trained using a Bayesian learning approach and is applied in two stages. The Bayesian approach has the feature that a validation set is not needed in the training phase to ensure good generalisation abilities, unlike methods which use maximum likelihood for training. The validation and training data sets that were created from the original data are combined to form a single data set which is used in training this network.

The scaled conjugate gradient (SCG) algorithm introduced by Møller is used for network training and has been shown to be significantly faster when training networks since it avoids line searches in the optimisation and includes second order gradient information in searching the weight space[9]. This training algorithm is combined with weight regularisation to ensure that over-training does not occur. Over-trained networks are characterised by network weights with large values, resulting in sharp decision boundaries. [4]. Weight regularisation results in smoother decision spaces by adding a penalty term to the error function, given by

$$\overline{E} = E + \frac{\alpha}{2}\sum_i w_i^2 \tag{4}$$

where E is the error function, which is the sum–of–squares error [4] and α, the regularisation coefficient which affects the degree of influence that the penalty has on the error. The networks are initially trained with a regularisation coefficient of 0.01.

The second stage of the training process involves optimising α, and retraining the networks with the optimal alpha. This optimisation is achieved by using the evidence framework presented by Mackay [10]. For a detailed discussion regarding this process and its implementation, the reader is referred to MacKay [10], Bishop [4] and Nabney [3].

The disadvantage of this technique lies in the determination of the Hessian matrix, which is computationally demanding. This is balanced by the improved generalisation accuracy of the network, and that using this technique does not require a separate validation set to test the generalisation of the network. The effect of the Bayesian learning on the networks with regularisation is shown in figure 3. The graph shows the evolution of network weights during the training process and that the weight values are restricted to the range of operation of the hidden layer activation function as a result of the α optimisation.

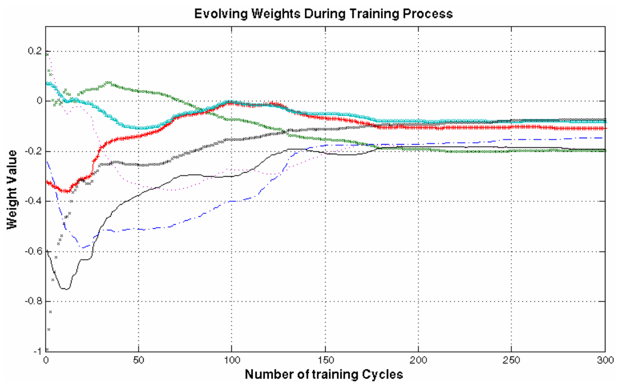


Figure 3: Evolving weights for the first hidden layer node showing weights converging to final values.

4.4. Estimation by Genetic Algorithms

A floating point representation is used for the GA implementation since this encoding allows the search to be faster, providing higher precision and produces results which are more consistent from run to run than other encodings [5]. Using this representation, the arithmetic crossover and non-uniform mutation operators have been used. The arithmetic crossover provides better stability in generating solutions, with lower standard deviation of the best solutions as compared to other types of crossover such as simple crossover. The non-uniform mutation also aids in the search of an optimal solution by allowing faster convergence and greater accuracy [5]. The GA system using these genetic operators is implemented with an initial population of 20 chromosomes, with 25 generations.

The fitness function used is a function of the neural network input and output. Mathematically, for a vector of inputs to a neural network $\mathbf{x}$, the output of the network is:

$$\mathbf{T} = f(\mathbf{W}, \mathbf{x}) \tag{5}$$

where, $\mathbf{W}$ is a vector of network weights, f is the transformation learned by the network during training and $\mathbf{T}$ is the output vector.

For the auto–associative network, the output will not be exactly that of the input, $\mathbf{T} \approx \mathbf{x}$; and a difference between the two exists which is defined as the error of the network.

$$\begin{aligned}\varepsilon &= \mathbf{x} - \mathbf{T} \\ \varepsilon &= \mathbf{x} - f(\mathbf{W}, \mathbf{x}) \end{aligned} \tag{6}$$

By minimising this error for values which are substituted for missing data into the database, an accurate estimate for the missing data which is based on similar data used to train the network, can be obtained. This error function must be non-negative and thus the square error is taken. The genetic algorithm will maximise the evaluation function, since we wish to minimise this error function, this is equivalent to maximising the inverse equation [5]. The equation is thus:

$$\varepsilon = -\left(\mathbf{x} - f(\mathbf{W}, \mathbf{x})\right)^2 \tag{7}$$

Since the input vector consists of both known and unknown entries, this error function can be re-written [6] as

$$\varepsilon = -\left(\left\{\begin{array}{c}\mathbf{x}_k \\ \mathbf{x}_u\end{array}\right\} - f\left(\mathbf{W}, \left\{\begin{array}{c}\mathbf{x}_k \\ \mathbf{x}_u\end{array}\right\}\right)\right)^2 \tag{8}$$

where $\mathbf{x}_u$ are the unknown and $\mathbf{x}_k$ are the known elements of the input vector $\mathbf{x}$. This is the fundamental equation required to relate the problem to the GA and to allow for successful data estimation.

4.5. An Agent-Based Approach

In this system a committee of agents, each agent being an approximation unit is used to estimate values for the missing data. The output of the committee is then the average of the output of the individual units, which is known as the equal weighting method. This system is shown diagrammatically in figure 4.

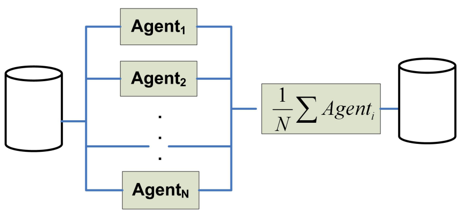


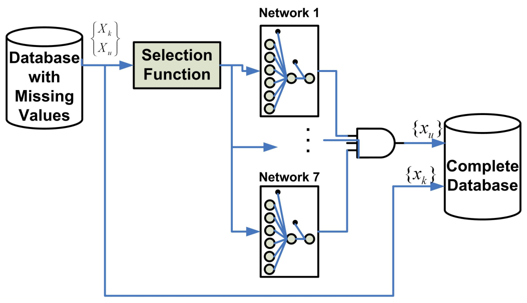
Figure 4: A multi-agent system where each agent is an approximation unit. The results of each unit are combined using an equal weighting approach.

The major design criterion is the number of agents which should form the committee. Large numbers of agents in the committee will increase the time taken to produce an estimate and is computationally demanding. For this application a committee consisting of six approximation units is used to test the benefit that is obtained when using this ensemble method.

5. Estimation by Data Classification

5.1. System Overview

This technique views the estimation of missing data a classification problem. Consider the database of values that is used, which consists of seven fields. If we assume that only one column can have missing data in any given record, then the available six fields are used as inputs to a neural network, whose output is the missing columns value. This system is shown diagrammatically in figure 5. Due to the nature of this system, each column in the database requires a neural network which is capable of estimating the missing columns values from the



Database with Missing Values
$\left\{\begin{array}{c}X_k \\ X_u\end{array}\right\}$
Selection Function
Network 1
Network 7
$\{x_u\}$
$\{x_k\}$
Complete Database

Figure 5: Estimation system consisting of neural network classifiers for each column of missing data. The selection function decides which of the networks is to be used.

other remaining inputs. The system also requires a selection function, which determines which column has missing data and selects the appropriate classifier to be used.

5.2. Network Architecture and Training

Neural network classifiers for every column have a simple architecture with 6 input units and a single output unit. because of this simple structure, two differing network architectures will be used and compared in the design. The first is an implementation of the classifiers as a set of generalised linear models (GLM). The Iterated Re-weighted Least Squares (IRLS) training method [3] is used and the network is trained with the training data set for 20 training cycles.

The second technique uses an MLP system as the implementation of the classifiers. The networks are trained for 500 training cycles using the training data set. For this network the optimal number of nodes was chosen as 20, as larger numbers of nodes in the hidden layer offered no significant advantage in the error. Both techniques are trained using weight decay with a regularisation coefficient of 0.01.

6. Results and Discussion

6.1. Experimental Setup

The estimation system was developed using the open source NETLAB [1] toolbox and the Genetic Algorithm Optimisation Toolbox (GAOT) [2] for the MATLAB computing environment.

The simulation procedure involves applying the estimation systems to the medical database described previously. The experiment involves systematically removing entries from each of the columns and obtaining an estimate for each missing value. The test set used, consisted of an extract of 100 entries which are estimated at a time. The estimates produced are compared to the actual value that is removed and is evaluated using the performance measures described in section 3.4.

6.2. Approximation Unit Results

The approximation unit estimated values for the test set in an average of 57.72 seconds[3], taken over 10 simulation runs. Table 1, shows the results for the correlation coefficient, mean square error and the accuracy of the estimation for simulations using both optimised and an unoptimised regularisation coefficient.

[1]NETLAB toolbox, I. T.Nabney. www.ncrg.aston.ac.uk/netlab

[2]GAOT toolbox, C. R. Houck *et. al.*. www.ie.ncsu.edu/mirage/GAToolBox/gaot

[3]Concluded using a Pentium IV, 2.66GHZ processor with 512MB RAM

Table 1: Simulation Results for estimations produced with i) $\alpha = 0.01$ and ii) with $\alpha = 11$

	$\alpha = 0.01$			$\alpha = 11$		
Col	A	MSE	r	A	MSE	r
2	31	10.25	0.25	53	6.81	0.3856
3	37	0.98	0.74	88	0.23	0.94
4	43	0.86	0.76	91	0.16	0.95
5	100	0.0	1	100	0.17	0.99
6	100	0.0	1	100	0	1
Mean A	62.2%			85%		

From the results shown, it can be seen that the approach with α optimised using the evidence framework has higher accuracy than the unoptimised network. It must be noted that the accuracy quoted here is the accuracy with a tolerance of 10%. Therefore, we can say on average that 85% of the missing values estimated in the database are accurate to within 10% of the exact value. This is demonstrated by the graph of figure 6, which shows how the estimated results fit within the shaded confidence region for columns three and five of the testing data.

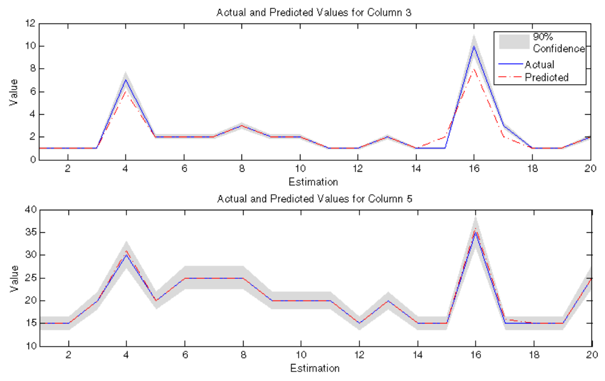


Figure 6: Graphs showing the actual values and the 90% confidence regions. The estimated values are superimposed and are seen in most cases to be within the shaded confidence regions which have been set.

A comparison of the statistical properties of the estimated and actual data sets is shown in table 2, where μ and σ are the mean and variance of the actual data; $\hat{\mu}$ and $\hat{\sigma}$ are the mean and variance of the estimated data; and ϵ_μ and ϵ_σ are the differences between the actual and estimated means and variances respectively.

Table 2: Comparison of Statistical properties for the actual and estimated data sets for each column.

Col	μ	$\hat{\mu}$	ϵ_μ	σ	$\hat{\sigma}$	ϵ_σ
2	8.67	9.56	0.89	11.96	4.33	7.63
3	2.28	2.24	0.04	2.65	2.08	0.56
4	1.24	1.22	0.02	2.61	2.03	0.58
5	22.5	22.49	0.01	34.59	34.21	0.38
6	2.5	2.5	0	1.38	1.38	0

The statistical properties indicate that the estimates are very close to the original properties, as indicated by the low values of ϵ_μ and ϵ_σ. This indicates that the estimation does not add biases to the data.

The agent based system, consisting of 6 approximation units improved on the results produced by a single approximation unit. The results of simulations is shown in table 3. These results show that the combination of the individual estimation on average improves the overall estimation accuracy for the entire database.

Table 3: Estimation results using the agent based approach. In this data $\alpha = 0.01$

Column	Accuracy	MSE	r
2	42	11.51	0.2959
3	97	0.92	0.81
4	96	1.02	0.80
5	100	0	1
6	100	0	1
Mean A	87%		

6.3. Estimation by Classification results

Simulations for this method of estimation involved training seven networks, one for each column of missing data for each of the two network architectures under investigation. The test set of 100 elements was then used to estimate each column of data which was removed and assumed to be missing. The results are shown in table 4.

Table 4: Simulation Results for estimations from missing data estimators using GLM and MLP networks.

	GLM			MLP		
Col	A	MSE	r	A	MSE	r
2	51	9.87	0.49	51	9.96	0.45
3	85	0.19	0.96	86	0.18	0.97
4	85	0.19	0.96	84	0.22	0.96
5	100	0.0	1	100	0	1
6	100	0.0	1	100	0	1
Mean A	84.2%			84.2%		

These results also show good performance on average for all columns estimated with 84% of the estimated values within 10% of the actual value. The methods produce almost the same estimation accuracies for all columns. From the above results, the GLM networks are preferred since they train much faster and require fewer training cycles. The fact that a GLM network produces this level of accuracy implies that there is a linear relationship between certain columns of the database, but not for all. As a results of this, the GLM is applicable to this problem, but may not be generally applicable to other databases. This property of the GLM networks suggest that they can be applied to a database in the initial stages of analysis to determine if there are any linear relationships between the data elements before more advanced estimators involving MLP networks are designed.

6.4. Discussion

The methods explored: missing data estimation using the approximation unit with the optimised alpha, the use of an agent based approximation system and the estimation by classification produce results which are comparable to each other with similar average accuracies. The approximation unit has the disadvantage that it consists of two optimisation stages: an optimisation of the neural network and a second optimisation for the genetic algorithm. An error in the first stage propagates to the next stage making the estimation poor. This is demonstrated by the fact that the approximation unit with alpha unoptimised produced poorer results than the system with the optimised alpha. A further disadvantage is that the computation time required to estimate a large number of values in lengthy.

This problem of multiple optimisations is only enhanced in the use of the multi-agent system, since multiple dual optimisations exist for every agent used in the committee. Given this though, the multiple optimisations allowed better results to be produced on average, as can be seen in the graph of figure 7.

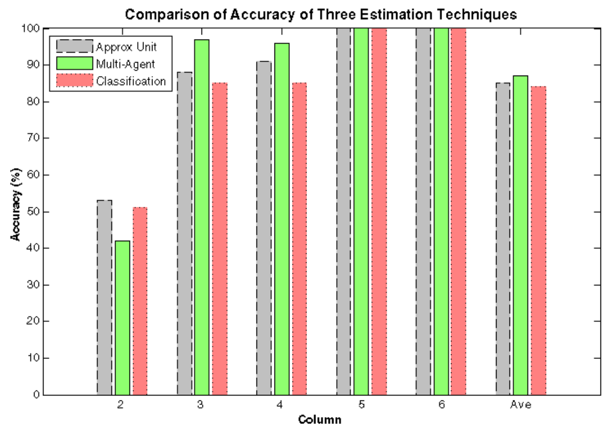


Figure 7: Comparison of the Accuracy of the three techniques described. The agent based system on average produces a better approximation to the missing data.

Viewing the problem from a data classification perspective also does not overcome the problem of multiple optimisations, since a neural network classifier is needed for each of the columns in the database. Another problem with this technique, is that if this system were to be extended to estimate multiple missing values in any given record of the database, the number of network classifiers required increases dramatically. In general for *n* columns in the database with up to *M* missing values, the number of network classifiers needed is given by:

$$N_{classifiers} = \sum_{r=0}^{M} \binom{n}{r} = \sum_{r=0}^{M} \frac{n!}{r!(n-r)!} \qquad (9)$$

This obviously is a large number of classifiers for anything but a trivial number of missing values in any given record; and the number of networks to be trained and optimised will prove to be time consuming and error prone.

It must be noticed that none of the methods were able to approximate values in column 2 with a high accuracy and with good coefficient of correlation. This suggests a breakdown in the initial assumption that data missing in this column is MAR. We must therefore conclude that the data which may be missing in these columns is either MCAR or MNAR, requiring more specialised methods of estimation such as pattern mixture models or selection models [2].

7. Conclusions and Further Work

Methods for the estimation of missing data have been presented which include the method of approximation units, an agent based system consisting of a committee of approximation units as well as network classifiers which perform estimation. These methods each produced results which are similar, each implementing certain architectures which optimise the estimation of the data. It is suggested that the method using a classification system be applied to estimate missing values in databases with few columns and few missing data values in any record. For databases with a large number of columns, the method of approximation units either singly or in committee, is better suited as they do not require large numbers of networks to be trained. The added benefit given by the committee based system must be evaluated in respect of the type of data being estimated and the range of values which are being estimated.

Further work must include a comparison of the accuracy of the results produced using the techniques presented to the results produced by techniques such as multiple imputation and maximum likelihood. An investigation into whether this type of a system can be used in the classification of missing data as either MAR, MCAR or MNAR should also be undertaken. The results presented show that this method identifies which columns of data in a database can be estimated given the other values; indicating that data missing from these columns, if missing can be classified as MAR. The classification of missing data according to the categories identified by Little and Rubin is difficult for large data sets, and thus this exploration could prove promising.

8. References

[1] R. Little and D. Rubin. *Statistical Analysis with Missing Data*. New York: John Wiley and Sons, first ed., 1987.

[2] J. L. Schafer and J. W. Graham. “Missing Data: Our View of the State of the Art.” *Psychological Methods*, vol. 7, no. 2, pp. 147–177, 2002.

[3] I. T. Nabney. *Netlab: Algorithms for Pattern Recognition*. London: Springer, 2003.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition*. New York: Oxford University Press, 2003.

[5] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer, third, revised and extended ed., 1999.

[6] M. I. Abdella. *The Use of Genetic Algorithms and Neural Networks to Approximate Missing Data in Database*. Master’s thesis, School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, January 2005.

[7] W. Qiao, Z. Gao, and R. G. Harley. “Continuous On-line Identification of Nonlinear Plants in Power Systems with Missing Sensor Measurements.” In *Proc. of Intnl. Joint Conf. on Neural Networks*, pp. 1729–1734. IEEE, July, 31 – August, 4 2005.

[8] B. B. Thompson, R. J. Marks II, J. J. Choi, M. El-Sharkawi, M. Huang, and C. Bunje. “Implicit Learning in Auto-encoder Novelty Assessment.” In *Proceedings of Intnl. Joint Conf. on Neural Networks IJCN’02*, pp. 2878–2883. IEEE, May 2002.

[9] M. F. Møller. “A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning.” *Neural Networks*, , no. 6, pp. 525–533, 1993.

[10] J. C. MacKay. *Bayesian Modelling and Neural Networks*. Ph.D. thesis, Department of Computation and Neural Systems, CalTech, 1992.