Information Sciences xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A dynamic programming approach to missing data estimation using neural networks

Fulufhelo V. Nelwamondo [a,b,*], Dan Golding [a], Tshilidzi Marwala [b]

[a] Modelling and Digital Science Unit, Council for Scientific and Industrial Research, P.O. Box 91230, Auckland Park, 2006, Johannesburg, South Africa
[b] Faculty of Engineering and the Built Environment, University of Johannesburg, P.O. Box 524, Auckland Park, 2006, Johannesburg, South Africa

ARTICLE INFO

Article history:
Received 31 July 2008
Received in revised form 30 September 2009
Accepted 6 October 2009
Available online xxxx

Keywords:
Missing data
Neural networks
Data imputation
Dynamic programming
Genetic algorithms

ABSTRACT

This paper develops and presents a novel technique for missing data estimation using a combination of dynamic programming, neural networks and genetic algorithms (GA) on suitable subsets of the input data. The method proposed here is well suited for decision making processes and uses the concept of optimality and the Bellman's equation to estimate the missing data. The proposed approach is applied to an HIV/AIDS database and the results shows that the proposed method significantly outperforms a similar method where dynamic programming is not used. This paper also suggests a different way of formulating a missing data problem such that the dynamic programming is applicable to estimate the missing data.

1. Introduction

Decision making processes are highly dependent on the availability of data, from which information can be extracted. All scientific, business and economic decisions are somehow related to the information available at the time of making such decisions. It is for this reason that the problem of missing data afflicts a variety of research and application areas in fields such as engineering, economics, finance and many more. Most predictive and decision making models designed to use a specified number of inputs will breakdown when one or more inputs are not available. In many such applications, simply ignoring or deleting the incomplete record (known as case deletion) is not a favorable option, as it may bring more harm than good [2]. In a statistical model, case deletion can also lead to biased results and in applications such as machine control, case deletion may result in breakdown of machinery [23]. Many techniques to estimate missing data that are aimed at minimizing the bias or output error of a model have been extensively researched [17,24,25]. Most of these are statistical methods, one of the most successful being Bayesian multiple imputation [25]. It is unfortunate that most decision making tools such as the commonly used neural networks and many other computational intelligence techniques cannot be used for decision making if data are not complete. In such cases, the optimal decision output should nevertheless, still be maintained despite the missing data. The estimation of missing input vector elements requires a system that possesses the knowledge of certain characteristics such as correlations between variables, which are inherent in the input space. Computational intelligence techniques and maximum likelihood techniques do possess such characteristics and, as a result, are useful in the imputation of missing data [20].

* Corresponding author. Address: Modelling and Digital Science Unit, Council for Scientific and Industrial Research, P.O. Box 91230, Auckland Park, 2006, Johannesburg, South Africa. Tel.: +27 11 358 0051; fax: +27 11 358 0230.
E-mail address: fnelwamondo@csir.co.za (F.V. Nelwamondo).

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.10.008

This paper proposes a novel technique for missing data estimation, grounded on the theory of dynamic programing. The novel method proposed here uses neural networks and genetic algorithms (GA) on suitable subsets of the input data, and assumes some model that emits data, some of which are missing. The remainder of this paper is arranged as follows: Section 2 presents a literature review and discusses related methods. The problem is presented in detail in Section 3, followed by the background information in Section 4. Sections 5–7 present in detail, information on the proposed model together with a description of the base model. Lastly, experimental results are given followed by a discussion.

2. Literature review

The problem of missing data in decision making has been a subject of research for over four decades, with more emphasis on statistical methods [24]. There are many applications that some researchers have worked on, with a major interest in making decisions with incomplete information. Dokuchaev [9] presented some qualitative methods and empirical rules for incomplete information in stochastic models in financial investment. Research in the field of missing data also took off in machine learning. Cogill et al. [7] studied a decentralized control problem using dynamic programming. One notable similarity between their study and the one presented in this paper is that they are both motivated by the fact that in many applications, decisions must be made at each time period and often with incomplete information. Cogill et al. [7] and Qiao et al. [22] do not attempt to estimate missing data, but present a model that can work reasonably well in the presence of missing data. Another relevant work was conducted by Geffner and Bonet [11] when solving a high-level planning with incomplete information. In their approach, they use partially observable Markov decision processes, which they transform into controllers using dynamic programming. The goal of their work was to design a shell that accepts partially observable Markov decision processes (POMDP) and produce controllers. Clearly, they were not estimating missing data as it is done in this work.

Some of the researchers who attempted to estimate missing data using some of the tools in machine learning include Twala [28] and He [13] who investigated the problem using decision trees. A thorough review of the literature reveals that there has not been any work on this area using dynamic programming. Some work on this area is in the book by Marwala [18], where he only points out dynamic programming as an emerging technique. Nelwamondo [19] has applied dynamic programming in the missing data problem. The approach used by Nelwamondo breaks the problem into smaller sub-problems, and the sub-problems are broken down into even smaller sub-problems and these smaller sub-problems were solved strategically to solve the bigger problem of missing data. The next section defines the problem and will justify why the problem is solved using dynamic programming.

3. Problem formulation and assumptions

Suppose there exists some dataset D with n variables and some large number of observations of these variables. Further suppose that, at each observation, all n variables are recorded as if observed. When recorded, all the observations are used as input to some decision making system. There certainly will be a problem if at some time t, some variables are missing from the observation. This paper defines a set $S \in \{0, 1, \ldots, n\}$ of possible states of the world. In this case, states can be any variable that is missing. State 0 is the one where all variables are observed and the remaining states correspond to the features. By design, the model is assumed to only be at one state at a time and can jump to any state in the next observation. In other application, the states can also be defined as a combination of the missing variables, but this will not be the focus of this paper. The transition matrix for this process of changing states is considered in this paper to be unknown, but can be derived from the data. We also define a set of corresponding actions that depend on what state the missingness model is. There are three possible actions given as [*do-nothing, recall, estimate*]. *Do-nothing* is the action that will be optimal if no data are missing or the state is 0. *Recall* is taken when a similar case has been observed in the past such that the best policy now will be to recall the action taken then, and *estimate* occurs when the missing data have to be estimated as discussed later in this paper. There is also a reward model R, which offers the highest reward when the estimate is accurate. In this case, the reward is only associated with the states and not the combination of the state and action. What is meant here is that the reward model does not look at what action was taken. It instead, only looks at how close the estimates are to the real answers. It becomes clear that what is being solved is the optimality equation, where we want to derive the best policy, and this is done well using dynamic programming [3]. The next section presents some background theory of the tools used.

4. Background information

4.1. Auto-encoder neural networks

Auto-encoders, also known as auto-associative neural networks, are neural networks trained to recall the input space. Thompson et al. [27] distinguish two primary features of an auto-encoder network, namely, the auto-associative nature of the network and the presence of a bottleneck that occurs in the hidden layers of the network, resulting in a butterfly-like structure. In cases where it is necessary to recall the input, auto-encoders are preferred due to their remarkable ability to learn certain linear and non-linear interrelationships such as correlation and covariance inherent in the input space. In this paper, auto-encoders, as shown in Fig. 1 are constructed using the multi-layer perceptrons (MLP) networks and trained using back-propagation. There are many training algorithms that could have been used, such as those presented in [21].

Training estimates the weight vector $\vec{W}$ to ensure that the output is as close to the target vector as possible [15]. The problem of identifying the weights in the hidden layers is solved by maximizing the probability of the weight parameter using Bayes' rule [26] as follows:

$$P(\vec{W}|D) = \frac{P(D|\vec{W})P(\vec{W})}{P(D)} \tag{1}$$

where, D is the training data, $P(D|\vec{W})$ is the likelihood, and $P(D)$ is the maximum likelihood term, often referred to as the evidence term that balances between fitting the data well and avoiding overly complex models whereas $P(\vec{W})$ is the prior probability of $\vec{W}$. The input, $\vec{x}$, is transformed to the middle layer, a, using weights W_{ij} and biases b_i as follows [26]:

$$a_j = \sum_{i=1}^{d} \vec{W}_{ji}\vec{x}_i + b_j \tag{2}$$

where $j = 1$ and $j = 2$ represent the first and second layer, respectively. The input is further transformed using the activation function such as the hyperbolic tangent (*tanh*) or the sigmoid in the hidden layer. More information on neural networks can be found in [6].

4.2. Genetic algorithms

Genetic algorithms use the concept of "survival of the fittest" over consecutive generations to solve optimization problems [12,16]. As in biological evolution, the fitness of each population member in a generation is evaluated to determine whether it will be used in the breeding of the next generation. In creating the next generation, techniques (such as inheritance, mutation, natural selection, and recombination) that are common in the field of evolutionary biology are employed. The genetic algorithm implemented in this paper uses a population of string chromosomes, which represent a point in the search space [12]. Instead of GA, any other stochastic optimization tool could have been used. However, genetic algorithms were preferred due to their ability to find a global optimal solution quicker [30]. In this paper, all GA parameters were empirically determined. The optimal GA was obtained with 70% probability of reproduction, 3% mutation rate and 60% crossover rate. GA is implemented by following three main procedures which are selection, crossover and mutation. The algorithm listing in Fig. 2 illustrates how GA operates.

4.3. Background on dynamic programming

The term 'dynamic programming' originates from the term 'mathematical programming', which implies optimization of a mathematical expression [3]. Dynamic programming uses the concept of optimality, that can be translated to optimization in stages. It follows that, for an optimal sequence of decisions, each sub-sequence must be optimal [3]. The concept of dynamic programming can be a useful tool to the problem of missing data, that optimizes all the sub-steps in the solution. Using the concept of optimality, to obtain the best estimate of missing data, all steps leading to the solution need to be optimized. This concept has several advantages that can improve the method proposed by Abdella and Marwala [1], which shall be used as a baseline method in this paper.

A problem is solved step-wise, obtaining the optimal solution for every step, keeping track of recurrences and overlaps. A complex problem is broken into sub-problems which are even broken into subsub-problems if possible. The objective behind this process is to have very manageable problems. When these sub-problems are solved, all solutions are stored and are retrieved if a similar problem is encountered. The dynamic programming equation, well known as the Bellman equation for evaluation is defined as follows [3]:

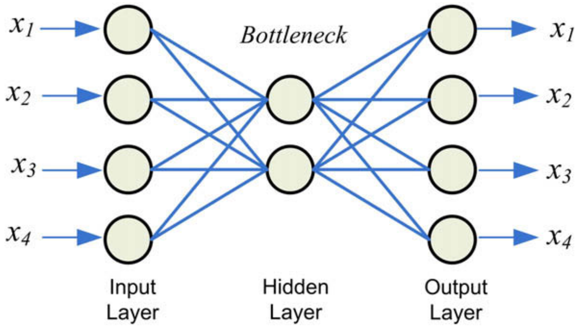

Fig. 1. The structure of a four-input, four-output auto-encoder.

```
BEGIN
  1    Create an initial population beginning at an initial
       generation, g=0.
  2    for each population P, evaluate each population member
       (chromosome) using the defined fitness evaluation
       function possessing the knowledge of the competition
       environment.
  3    Choose population members for breeding based on the
       fitness from (2) and selection algorithms such as the
       Roulette Wheel or Tournament can be used.
  4    using genetic operators such as inheritance, mutation
       and crossover, alter P(g) to produce P(g+1) from the
       fit chromosomes in P (g).
  5    repeat (2), (3) and (4) until the termination criteria
       is met.
 END
```

Fig. 2. Schematic representation of the genetic algorithm operation.

$$J(t) = \sum_{k=0}^{\infty} \gamma^k U(t+k) \tag{3}$$

where $0 < \gamma < 1$ is the discount factor and U is the utility function.

The training stage is aimed at minimizing the evaluation function. In this problem, dynamic programming is viewed as an optimization problem with [5]:

$$x_{k+1} = f_k(x_k, U_k, r_k), \quad k = 0, 1, \ldots, N-1 \tag{4}$$

where k is the discrete time, x_k is the state, which is the known data that will be relevant for future optimization. U_k is the decision that will be selected. N is the number of times the control has been applied and r can be viewed, in this case, as some form of a disturbance or some bias term. This leaves us with an additive cost function of the form [5]:

$$E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, U_k, r_k) \right\} \tag{5}$$

where $g(u)$ is the cost function and for this model to work, we assume that u_k is selected with knowledge of x_k [5].

5. Base model for imputation

The model proposed by Abdella and Marwala [1] is used as a baseline technique. This method combines the use of auto-associative neural networks with genetic algorithms to approximate missing data. A genetic algorithm is used to estimate the missing values by optimizing an objective function as presented shortly in this section. The complete vector combining the estimated and the observed values is fed into the auto-encoder as input and as shown in Fig. 3. Symbols X_k and X_u represent the known variables and the unknown or missing variable, respectively. The combination of X_k and X_u represent the full input space.

Considering that the method uses an auto-encoder, one will expect the input to be very similar to the output for a well chosen architecture of the auto-encoder. This is, however, only feasible on a dataset similar to the problem space from which the inter-correlations have been captured. In other words, the auto-encoder is trained to minimize the difference between its input and output vectors only for an input vector with sufficiently similar properties to those of the input vectors upon which it was trained. This arrangement is what allows the auto-encoder to estimate missing data. The difference between

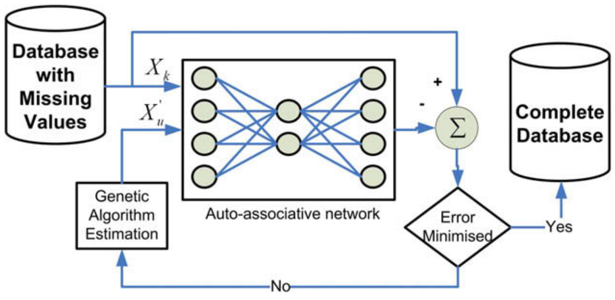

Fig. 3. Auto-associative neural network and genetic algorithm model for missing data imputation.

the target and the actual output is used as the error. The square of the error is used to create a function which has a global optimum where the difference between the target and the output is zero. By squaring this error, the zero-crossing of the linear error becomes the minimum error of the quadratic error. This leads to the following equation:

$$\varepsilon = \left(\begin{Bmatrix} X_k \\ X_u \end{Bmatrix} - f\left(\vec{W}, \begin{Bmatrix} X_k \\ X_u \end{Bmatrix} \right) \right)^2 \tag{6}$$

where X and $\vec{W}$ are input and weight vectors whereas X_k and unknown, X_u represent the known and unknown input variables, respectively. This equation is used as the objective function that is minimized using GA. More details about the base method can be found in [1]. As a further illustration of this method, consider an input vector with both known and missing variables. By "guessing" a value for the missing variable and evaluating the error in Eq. (6), one can get a sense of how correct the guessed value is, in relation to the ground accurate value.

6. Implementation of the proposed missing data estimator

6.1. Assumptions

The model proposed gives a reward $R(i,a)$ in state $i \in S$, when action a is taken. As before, S is the state space, with all possible states. In the process, a Markov assumption that the transition model only depends on the current state is made. From this assumption, it follows that history of the states transition has no effect on the current estimation. Another consideration is that, if an action is considered to be the best policy on one state, it does not necessarily mean that it should be taken without "applying the mind" into it. Thus, it does not mean that since the model took action a in state i, it has to continue taking the same action when the same state is revisited. As a result, once you take a wrong action, it does not necessarily mean that we have to keep doing the same wrong action. Each case is evaluated on its own merit such that the model decides on the best policy, as governed by the penalties given for such actions. On the other hand, the disadvantage can be that, a lot of resources are used whereas history could have just revealed what action was taken in the past, and no resources were going to be used, more especially if the action taken was indeed optimal.

6.2. Actual model

The concept of dynamic programing has mainly been applied to optimal control problems. This is due to the fact that in many control problems, decisions are mostly taken with incomplete information about the state of the system [8]. The missing data problem under investigation requires a similar approach to be taken on a large number of records. Generally, there are two approaches that are followed in the implementation of dynamic programming. These approaches can be described as follows:

(a) **Top-down approach** where the problem is divided into sub-problems whose solutions are acquired and stored for future use.
(b) **Bottom-up approach** where a sub-problem is anticipated and solved before hand and used to construct the solution to the bigger problem. It is however, not always instinctive to guess all problems before hand.

According to the Bellman's equation [3], if the policy function is optimal for the infinite summation, then it has to be true that irrespective of the initial estimate of the missing data, the remaining decisions should characterize an optimal policy. In this problem of missing data estimation, the principle of optimality is connected to the theory of optimal substructure. The error given in Eq. (6) is the penalty value that the model receives when a prediction has been made. Thus, a penalty of zero is only achieved when there is no error in prediction. But since there is an action which allows no value to be predicted, *Do-nothing*, it follows that the penalty equation needs to be adjusted to fit this scenario. The reward/penalty model is thus defined as follows:

$$R(a,i) = \begin{cases} -\varepsilon & \text{if estimation took place} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

A very strict rule is that the only time the model will not estimate the missing values is when all states are observed (i.e., when there is no missing data). In this model, the objective is to find a policy that will minimize the penalties given that there are some states that are not observed. Following from this, at any state i, the optimal policy π^* is the action (from the action space) that maximizes the expected utility amongst all the possible actions that are available in such state. Using the common Q function, used by the Markov decision process community [4,14]

$$\pi^*(i) = arg\max_x Q(i,a) \tag{8}$$

where the $Q(i,a)$ is defined in Eq. (10). If the optimal policy is chosen, the total value is given by:

$$V(i) = Q(i, \pi^*(i)) \tag{9}$$

where

$$Q(i,a) = R(i,a) + \sum_j T_{ij}^a V(j) \tag{10}$$

In this notation, $R(i,a)$ is, as before, the reward for taking action a in state i and T_{ij}^a is the transition matrix indicating the transition from state i to state j, when action a is taken. The top-down approach was adopted and the pseudo-code is presented in the Algorithm listing 1.

Algorithm 1. Auto-associative neural networks and genetic algorithms model for missing data imputation: dynamic programing approach

```
input: Given database with missing values
output: Imputed missing value
← To begin: read and code the data ← Granularize the data (using the
   preferred method)
← Read the next record in the database forall (records 1 → N) do
|  Find the appropriate category of the data
end
while Creating a model do
    ← Check if the category has appeared before
    forall records → N do
        ← IF record has been seen before
        → THEN Retrieve the correct model and use to impute
        ← ELSE:
        → Build the training data
        → Create a new model
        → Store the model;
        → THEN Determine the optimal policy
    end
end
```

After implementation it was found that the algorithm breaks down when there is a group of categories with no complete records. In this case the algorithm supplies no training data for the neural network. A solution is granularize as done by the fuzzy community [29]. This helps in finding data that would then fit the record in question. In an attempt to overcome the challenges mentioned in the preceding section, data were grouped for maximum homogeneity as presented next.

7. Grouping for maximum homogeneity

For the purpose of illustration, let us consider one variable from some dataset, namely *age*. An easy implementation will granularize *age* into four equal bins. This will lead to definite distinctions in age. However, a drawback of this is that borderline categories are not well presented. For borderline cases, in one year one person is in one group and the following year, one could be in another group, even though no significant change has taken place. The concept of overlapping bins brings light to this challenge [19] and is common in the fuzzy community [29].

Given the observed data of K elements with numerical measure a_i, categories are defined as follows [10]:

$$D = \sum_{i=1}^{K} w_i (a_i - \bar{a}_i)^2 \tag{11}$$

In this equation, w_i is some weight assigned to the element i and $\bar{a}_i$ denotes the weighted arithmetic mean of the numerical measures that fall within the group that element i belongs to. The parameter D here defines the squared distance in the *restricted problem* as defined in [10] and this parameter has to be minimized to the restriction (68% standard deviation in this case).

8. Experimental evaluation

8.1. Data and analysis

The dataset used in this test was obtained from the South African antenatal sero-prevalence survey of 2001. The data for this survey are obtained from questionnaires answered by pregnant women visiting selected public clinics in South Africa. Only women participating for the first time in the survey were eligible to answer the questionnaire. Data attributes used in this study are the *HIV status, education level, gravidity, parity, age, age of the father, race* and *region*. The HIV status is the decision and is represented in a binary form, where 0 and 1 represent negative and positive, respectively. Race is presented in 4 classes, where 0, 1, 2 and 3 represent African, coloured (mixed race), White and Asian, respectively. The education level was measured using integers representing the highest grade successfully completed, with 13 representing tertiary education.

Gravidity is the number of pregnancies, complete or incomplete, experienced by a female, and this variable is represented by an integer between 0 and 11. Parity is the number of times the individual has given birth and multiple births are counted as one. Both parity and gravidity are important, as they show the reproductive activity as well as the reproductive health state of the women. Age gap is a measure of the age difference between the pregnant woman and the prospective father of the child. A sample of this dataset is shown in Table 1.

During the pre-processing stage, all outliers were removed. As an example, all cases where parity exceeds the gravidity were removed as it is known that a woman cannot give birth more than she has been pregnant. All nine provinces of the country were represented and the age was found to be mainly between 16 and 45.

8.2. Testing criteria

Elements of the complete dataset were systematically removed. The program was then run on these now incomplete datasets and the imputed results were compared to the original values. The accuracy was measured as the percentage of numbers that were offset from the original value within a certain tolerance, and as follows:

$$Error = \frac{n_\tau}{N} \times 100\% \tag{12}$$

where n_τ is the number of predictions within a certain tolerance as shown in Table 2.

8.3. Experimental setup

The multi-layer perceptron neural networks were all trained using the scaled conjugate gradient learning algorithm with a linear activation function. The number of hidden nodes used was always two less than the number of input. Thompson et al. [27] has shown that auto-associative neural networks perform better when the number of hidden nodes is less than that of the input and output nodes. The GA was implemented using the floating point representation for 30 generations, with 100 chromosomes per generation. The mutation rate was set to 3%, with 70% probability of reproduction and 60% crossover rate. All these parameters were empirically determined.

8.4. Results and discussion

Fig. 4 shows the average percentages of the missing data to within each range for the tests done on one missing variable as described in Section 8.2. The vertical axis differentiates between the ranges listed in the previous subsection. The results shown here are averages after four runs on different data sections (see Tables 3 and 4).

These results presented above are compared to the results obtained using the baseline method. Fig. 5 presents the results of the baseline method.

Table 1
Extract of the HIV database used, with missing values.

Race	Region	Education	Gravidity	Parity	Age	Father	HIV
1	C	?	1	2	35	41	0
2	B	13	1	0	20	22	0
3	?	10	2	0	?	27	1
2	C	12	1	?	20	33	1
3	B	9	?	2	25	28	0
?	C	9	2	1	26	27	0
2	A	7	1	0	15	?	0
1	C	?	4	?	25	28	0
4	A	7	1	0	15	29	1
1	B	11	1	0	20	22	1

Table 2
Tolerances were used for each variable. For the variable HIV, there is no tolerance values as the output is either HIV negative or positive.

Variable	Tolerance
Age	To within 2, 5 and 10 years
Education level	To within 1, 3 and 5 grades
Gravidity	To within 1, 3, and 5 pregnancies
Parity	To within 1, 2 and 4 children
Father's age	To within 3, 7 and 15 years
HIV status	Not applicable

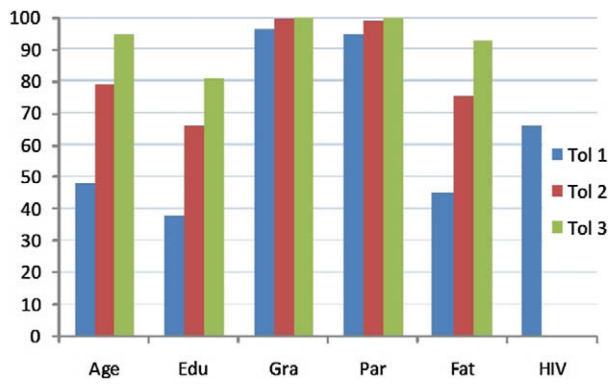

Fig. 4. Percentages of imputed data within a pre-determined tolerance when only one variable was missing for each record. Tolerances 1, 2 and 3 indicate the respective tolerances as presented in the previous subsection. The vertical axis shows the accuracy.

Table 3
Percentages of imputed data that fell with the ranges for records when only one variable is missing for each record.

	Age	Education	Gravidity	Parity	Father	HIV
Tolerance 1	48	38	96	95	45	66
Tolerance 2	79	66	99.6	99	75	–
Tolerance 3	95	81	100	99.9	93	–

Table 4
Percentages of imputed data that fell with the ranges for records when only one variable is missing for each record using the base method.

	Age	Education	Gravidity	Parity	Father	HIV
Tolerance 1	48	25	82.7	81.3	47	58
Tolerance 2	79	49	96	95	76	–
Tolerance 3	100	70.3	100	100	93	–

It is observed that age can only give insight into behavioral patterns. Whilst it is not possible to make assumptions about an individual's race based only on their age, statistical information can be gained into their sexual behavior. This idea is taken further by assuming that there is a probability distribution for determining the likelihood of someone to behave like someone else as a function of age. This distribution is assumed to be normally distributed around a candidate's age, based on the central limit theorem. In the model creation stage, a subset of data is picked such that 68% of the samples lie within one standard deviation of the age of the candidate.

It can be seen that the dynamic programming approach has an effect on the results. Results obtained from the dynamic model are better than the results obtained using the baseline approach. The experiment was run 10 times and the results

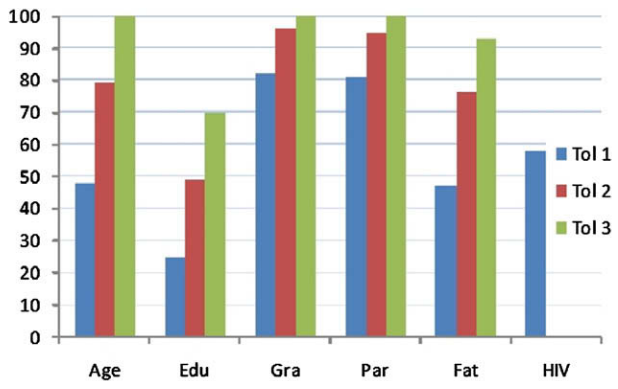

Fig. 5. Percentages of imputed data within a pre-determined tolerance when only one variable is missing for each record using the base method.

Table 5
Percentages of imputed data that fell within the ranges for records missing age and one other field.

	Education	Gravidity	Parity	Father	HIV
Tolerance 1	38	96	96	37	67
Tolerance 2	65	99.5	99	68	17
Tolerance 3	80	100	99.9	92	16
Tolerance 1 (Age)	50	47	50	35	48
Tolerance 2 (Age)	81	78	79	67	79
Tolerance 3 (Age)	96	95	95	90	96

presented here are the average results. This demonstrate that the results are better because of inclusion of dynamic programming and not because of GA finding better solutions. In the case of two missing data points, the combination of missing age and all the other variables was tested. Table 5 presents the results as above. The bottom three rows of each column correspond to age, whilst the top three correspond to the variable that heads the column.

It can be seen from this table that removing two fields has no noticeable difference to removing one, except with the combination of age and father's age. Both of these significantly lose predictability when they are both absent, indicating some dependence. A few tests were done on three and four missing fields and there were no noticeable differences, apart from when both age and father's age were missing. However when all six numeric fields were absent, the results dropped significantly across all variables.

9. Conclusion

A model using auto-associative neural networks and genetic algorithm was built to estimate missing data, using the principle of dynamic programing. The results indicate that dynamic programing adds many advantages to the base model. The difference is very significant and future work will look at the test of significance of the difference. Since the model was run more than 10 times for each variable, it is conclusive that credit has to be given to dynamic programming. The rationale as to why this method improves performance of the base model is that the base model assumes data variables to be somehow related to one another. From statistical analysis, it can be anticipated that parameters such as race have an influence on the other variables such as HIV. In this model, where these parameters are viewed as states, it shows that the states actually affect each other. Looking at this from a behavioral point, a person is likely to act more similarly to someone close to their age than someone much older or younger than them. However, with this model, it is left to the model to derive the policy that maximizes the prediction.

Acknowledgment

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. The first author further thanks the Harvard-South Africa Fellowship for an opportunity to complete this paper while at Harvard University. Thanks also go to Ishmael Msiza of the CSIR for proof-reading this document, and to the anonymous reviewers for their constructive comments.

References

[1] M. Abdella, T. Marwala, The use of genetic algorithms and neural networks to approximate missing data in database, Computing and Informatics 24 (2006) 1001–1013.
[2] P.D. Allison, Missing Data: Quantitative Applications in the Social Sciences, Sage, Thousand Oaks, CA, 2002.
[3] R. Bellman, Dynamic Programming, Princeton University Press, Princeton, 1957.
[4] R. Bellman, A Markovian decision process, Journal of the Mathematics and Mechanics 38 (1957) 716–719.
[5] D.P. Bertsekas, Dynamic Programming and Optimal Control, Athena Scientific, Belmont, MA, 2005.
[6] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[7] R. Cogill, M. Rotkowitz, B.V. Roy, S. Lall, An approximate dynamic programming approach to decentralized control of stochastic systems, Lecture Notes in Control and Information Sciences, Springer, Berlin 329 (2006) 243–256.
[8] R. Cogill, M. Rotkowitz, B.V. Roy, S. Lall, An approximate dynamic programming approach to decentralized control of stochastic systems, Lecture Notes in Control and Information Sciences 329 (2006) 243–256.
[9] N. Dokuchaev, Dynamic portfolio strategies: quantitative methods and empirical rules for incomplete information, Information Science Reference, IGI Global Publications, New York, 2002.
[10] W.D. Fisher, On grouping for maximum homogeneity, Journal of the American Statistical Society 53 (1958) 789–798.
[11] H. Geffner, B. Bonet, High-level planning and control with incomplete information using POMDPS, in: Fall AAAI Symposium on Cognitive Robotics, Orlando, FL, USA, 1998, pp. 10804–10811.
[12] D.E. Goldberg, Genetic Algorithms in Search Optimisation and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[13] Y. He, Missing data imputation for tree-based models, Ph.D. Thesis, University of California, Los Angels, 2006.
[14] R.A. Howard, Dynamic Programming and Markov Processes, MIT Press, 1960.
[15] N. Japkowicz, Supervised learning with unsupervised output separation, in: International Conference on Artificial Intelligence and Soft Computing, vol. 3, 2002, pp. 321–325.
[16] I. Kaya, A genetic algorithm approach to determine the sample size for attribute control charts pages, Information Sciences 179 (10) (2009) 1552–1566.
[17] R.J.A. Little, D.B. Rubin, Statistical Analysis with Missing Data, Wiley, New York, 1987.

[18] T. Marwala, Computational Intelligence for Missing Data Imputation, Estimation, and Management: Knowledge Optimization Techniques, Information Science Reference, USA, 2009.
[19] F.V. Nelwamondo, Computational intelligence techniques for missing data imputation, Ph.D. Thesis, University of the Witwatersrand, Johannesburg, 2008.
[20] F.V. Nelwamondo, T. Marwala, Rough set theory for the treatment of incomplete data, in: Proceedings of the IEEE International Conference on Fuzzy Systems, London, UK, 2007, pp. 338–343.
[21] R. Pasti, L.N. de Castro, Bio-inspired and gradient-based algorithms to train MLPS: the influence of diversity, Information Sciences 179 (10) (2009) 1441–1453.
[22] W. Qiao, Z. Gao, R.G. Harley, Continuous online identification of non-linear plants in power systems with missing sensor measurements, in: IEEE International Joint Conference on Neural Networks, Montreal, 2005, pp. 1729–1734.
[23] P.L. Roth, F.S. Switzer III, A Monte Carlo analysis of missing data techniques in a HRM setting, Journal of Management 21 (1995) 1003–1023.
[24] D.B. Rubin, Multiple Imputation for Nonresponse in Surveys, Wiley, New York, 1987.
[25] J. Schafer, Analysis of Incomplete Multivariate Data, Chapman & Hall, 1997.
[26] B.B. Thompson, R.J. Marks, J.J. Choi, Implicit learning in auto-encoder novelty assessment, in: IEEE International Joint Conference on Neural Networks, vol. 3, 2002, pp. 2878–2883.
[27] B.B. Thompson, R.J. Marks, M.A. El-Sharkawi, On the contractive nature of auto-encoders: application to sensor restoration, in: Proceedings of the International Joint Conference on Neural Networks, vol. 4, 2003, pp. 3011–3016.
[28] B.E.T.H. Twala, Effective techniques for handling incomplete data using decision trees, Ph.D. Thesis, The Open University, UK, 2005.
[29] L.A. Zadeh, Is there a need for fuzzy logic?, Information Sciences 178 (13) (2008) 2751–2779
[30] J. Zhang, J. Zhuang, H. Du, S. Wang, Self-organizing genetic algorithm based tuning of PID controllers, Information Sciences 179 (6) (2009) 1007–1018.