Studies in Big Data 48

Collins Achepsah Leke
Tshilidzi Marwala

Deep Learning and Missing Data in Engineering Systems

Springer

Studies in Big Data

Volume 48

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

The series “Studies in Big Data” (SBD) publishes new developments and advances in the various areas of Big Data- quickly and with a high quality. The intent is to cover the theory, research, development, and applications of Big Data, as embedded in the fields of engineering, computer science, physics, economics and life sciences. The books of the series refer to the analysis and understanding of large, complex, and/or distributed data sets generated from recent digital sources coming from sensors or other physical instruments as well as simulations, crowd sourcing, social networks or other internet transactions, such as emails or video click streams and others. The series contains monographs, lecture notes and edited volumes in Big Data spanning the areas of computational intelligence including neural networks, evolutionary computation, soft computing, fuzzy systems, as well as artificial intelligence, data mining, modern statistics and operations research, as well as self-organizing systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at http://www.springer.com/series/11970

Collins Achepsah Leke · Tshilidzi Marwala

Deep Learning and Missing Data in Engineering Systems

Springer

Collins Achepsah Leke
Faculty of Engineering and Built Environment
University of Johannesburg
Auckland Park, South Africa

Tshilidzi Marwala
Faculty of Engineering and Built Environment
University of Johannesburg
Auckland Park, South Africa

ISSN 2197-6503 ISSN 2197-6511 (electronic)
Studies in Big Data
ISBN 978-3-030-01179-6 ISBN 978-3-030-01180-2 (eBook)
https://doi.org/10.1007/978-3-030-01180-2

Library of Congress Control Number: 2018960236

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Deep Learning and Missing Data in Engineering Systems discuss concepts and applications of artificial intelligence, specifically, deep learning. The artificial intelligence techniques that are studied include multilayer autoencoder networks and deep autoencoder networks. Also studied in this book are computational and swarm intelligence techniques which include ant colony optimization, ant lion optimizer, bat algorithm, cuckoo search optimization, firefly algorithm and invasive weed optimization algorithm. In addition, this book explores using deep autoencoder networks with a varying number of hidden layers. This book also studies the reconstruction of images from reduced dimensions obtained from the bottleneck layer of the deep autoencoder.

These techniques are used to solve the missing data problem in an image recognition and reconstruction context.

This book is an interesting reference for graduate students, researchers and artificial intelligence practitioners with interest in deep learning and image processing.

Auckland Park, South Africa
July 2018

Collins Achepsah Leke, Ph.D.
Tshilidzi Marwala, Ph.D.

Acknowledgements

We would like to thank the following people for their contribution to this manuscript: Msizi Khoza, Dr. Mlungisi Duma, Prof. Fulufhelo Nelwamondo, Dr. Vukosi Marivate, Dr. Marcos Alvares, Dr. Pramod Parida, Dr. Ali Hasan, Dr. Ilyes Boulkaibet and Dr. Satyakama Paul.

We also thank colleagues and practitioners that have collaborated directly and indirectly to the writing of the manuscript including Dr. Alain Richard Ndjiongue, Stephan Martin Nlom, Mokesioluwa Fanoro, Chwayita Macanda and Nqobile Dudu. We thank Prof. Bhekisipho Twala.

We dedicate this book to Leke Nee Nchangeh Agatha Fonkeng, Leke Nchabanu Gwendoline, Dr. Leke Clarence Fonkeng, Dr. Leke Betechuoh Brain, Mbali Denga Marwala, Lwazi Thendo Marwala, Nhlonipho Khathushelo Marwala and Dr. Jabulile Vuyiswa Manana.

July 2018

Collins Achepsah Leke
Tshilidzi Marwala

Contents

About the Authors

Collins Achepsah Leke born in Yaounde (Cameroon) is a postdoctoral research fellow at the University of Johannesburg. He holds a Bachelor of Science with Honours in Computer Science from the University of the Witwatersrand, a Master of Engineering from the University of Johannesburg and Ph.D. in Engineering from the University of Johannesburg. His research interests include the applications of computational intelligence to engineering, computer science, finance, social science and medicine.

Tshilidzi Marwala born in Venda (Limpopo, South Africa) is the Vice Chancellor and Principal of the University of Johannesburg beginning. Previously, he was the Deputy Vice Chancellor for Research and Internationalization and the Executive Dean of the Faculty of Engineering and the Built Environment both at the University of Johannesburg. From 2003 to 2008, he progressively held the positions of Associate Professor, Full Professor, the Carl and Emily Fuchs Chair of Systems and Control Engineering as well as the SARChI Chair of Systems Engineering at the Department of Electrical and Information Engineering at the University of the Witwatersrand. From 2001 to 2003, he was the Executive Assistant to the technical director at South African Breweries. From 2000 to 2001, he was a postdoctoral research associate at the Imperial College (then University of London). He holds a Bachelor of Science in Mechanical Engineering (magna cum laude) from Case Western Reserve University (USA) in 1995, a Master of Mechanical Engineering from the University of Pretoria in 1997 and a Ph.D. specializing in Artificial Intelligence and Engineering from the University of Cambridge in 2000. Marwala completed the Advanced Management Program (AMP) at Columbia University Businesses School in 2017 and completed a Program for Leadership Development (PLD) at Harvard Business School in 2007. Tshilidzi is a registered professional engineer, a Fellow of TWAS (The World Academy of Sciences), the Academy of Science of South Africa, the African Academy of Sciences and the South African Academy of Engineering. He is a Senior Member of the IEEE (Institute of Electrical and Electronics Engineering) and a distinguished member of the ACM

(Association for Computing Machinery). His research interests are multidisciplinary, and they include the theory and application of artificial intelligence to engineering, computer science, finance, social science and medicine. He has an extensive track record in human capacity development having supervised 47 Master's and 28 Doctoral students to completion. Some of these students have proceeded with their doctoral and postdoctoral studies at leading universities such as Harvard, Oxford, Cambridge, British Columbia, Rutgers, Purdue, Chiba and Waseda. He has published 13 books in artificial intelligence, one of these has been translated into Chinese, over 300 papers in journals, proceedings, book chapters and magazines and holds four patents. He is an associate editor of the International Journal of Systems Science (Taylor and Francis Publishers). He has been a visiting scholar at Harvard University, University of California at Berkeley, Wolfson College of the University of Cambridge and Nanjing Tech University as well as a member of the programming council of the Faculty of Electrical Engineering at the Silesian University of Technology in Poland. He has received more than 45 awards including the Order of Mapungubwe and was a delegate to the 1989 London International Youth Science Fortnight (LIYSF) when he was in high school. His writings and opinions have appeared in the magazines New Scientist, The Economist and Time magazine.

Chapter 1
Introduction to Missing Data Estimation

1.1 Introduction

The presence of missing data affects the quality of the dataset, which impacts on the analysis and interpretation of the data. There are several reasons that could lead to data being missing in a dataset with some being more predominant than others. The first well-known reason is participants' denying revealing some personal and sensitive information, for example monthly income. The second main reason is the failure of systems meant to capture and store the data in databases. Another main reason is interoperability whereby information exchanged between systems may be subjected to missing data. In several cases, data collectors implement rugged procedures to render immune any data gathering system to the presence of missing data. Nonetheless, it is rather unfortunate that in spite of all these attempts, the presence of missing data remains a major issue in data analysis tasks. It is rare that the actual reason behind the missing data problem is known a priori, particularly in engineering problems. As a result, techniques aimed at addressing the missing data issue are normally unsuccessful. The incompleteness of available data has an adverse effect on decision-making processes courtesy of the reliance of decisions on complete information.

Many existing decision support systems such as the commonly used support vector machines, neural networks and several computational intelligence methods are predictive frameworks that use known data as input features and predict an output. Such models are incapable of operating and yielding decisions when one or more of the inputs having missing information. As a result, they cannot be used as part of the decision-making framework if some features within the dataset are incomplete. The objective of a missing data estimation procedure is normally to make the best decision. Achieving this objective requires that the right approximation of the missing data be made. With the missing data having been approximated, it is then possible to use the classification and regression tools for decision-making.

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_1

It is noteworthy differentiating between estimation and imputation of missing data. Missing data imputation basically refers to dealing with the missing data. This could involve deleting the set of data with missing information or by applying methods like listwise deletion, or simply estimating the missing data. Therefore, in this chapter, missing data estimation is considered a subset of missing data imputation. Generally, it has been observed that in a variety of datasets in the social sciences, missing data imputation is a viable means of dealing with the missing data. This is courtesy of the fact that in the social sciences, the objective of the statistical analysis task is to estimate statistical parameters such as standard deviations and means. In sectors such as in engineering, where data is often required for manual decision support or automated decision support, deleting any record with missing data is ideally not an option. As such, in the majority of instances, approximation of the missing data is the required approach. It is for this reason that in engineering problems, it is more appropriate using the term missing data estimation as opposed to missing data imputation.

1.2 The Evolution of Missing Data Estimation Methods

Prior to the 1970s, the issue of missing data was solved via editing, a setting in which a missing value was logically implied using additional data that were given. In 1976, a method for inference from incomplete data was developed. After this, Dempster et al. (1997) came up with the expectation maximization (EM) algorithm that led to the use of the maximum likelihood (ML) missing data estimation approaches. Less a decade later, Little and Rubin (1987) identified the limits of single imputations and case deletion. They introduced multiple imputations (MI). Multiple imputations would not have been feasible without concurrent progress in computational power. This is because they require a lot computational power (Ho et al. 2001; Faris et al. 2002; Hui et al. 2004; Sartori et al. 2005). From the 1990s till date, several missing data estimation techniques have been developed and applied to different sectors. Recently, researchers have begun studying how sensitive the missing data estimation results are on the decision-making process outcomes that use the estimated variable values. There has been research conducted to come up with novel methods to approximate missing data entries. For instance, artificial intelligence methods like neural networks and optimization algorithms like evolutionary computation algorithms are some of the approaches increasingly being used in several missing data estimation tasks (Dhlamini et al. 2006; Nelwamondo et al. 2007a; Nelwamondo and Marwala 2007a, 2008). Some of the techniques applied to the missing data imputation problem include rough sets (Nelwamondo and Marwala 2007b), the semi-hidden Markov models (Yu and Kobayashi 2003) and fuzzy approaches (Gabrys 2002; Nelwamondo and Marwala 2007c), Hopfield neural networks (Wang 2005) and genetic algorithms (Junninen et al. 2004; Abdella 2005; Abdella and Marwala 2005).

1.3 Missing Data Proportions

Missing data in datasets influences the analysis, inferences and conclusions reached based on the information. The impact on machine learning algorithm performances become more significant with an increase in the proportion of missing data in the dataset. Researchers have shown that the impact on machine learning algorithms is not as significant when the proportion of missing data is small in large-scale datasets (Ramoni and Sebastiani 2001; Tremblay et al. 2010; Polikar et al. 2010). This could be attributed to the fact that certain machine learning algorithms inherently possess frameworks to cater to certain proportions of missing data. With an increase in missing data proportions, for example cases where the proportion is greater than 25%, it is observed that tolerance and performance levels of machine learning algorithms decrease significantly (Twala 2009). It is because of these reduced levels in tolerance and performance that more complex and reliable approaches to solve the problem of missing data are required.

1.4 Missing Data Mechanisms

As stated before, it is very vital to identify the reason why data are missing. When the explanation is known, a suitable method for missing data imputation may then be chosen or derived, resulting in higher effectiveness and prediction accuracy. In many situations, data collectors may be conscious of such reasons, whereas statisticians and data users may not have that information available to them when they perform the analysis. In such scenarios, data users may have to use other techniques that can assist in data analysis to comprehend how missing data are related to observed data and as a result, possible reasons may be derived. A variable or a feature in the dataset is viewed as a mechanism, if it assists in explaining why other variables are missing or not missing. In datasets collected through surveys, variables that are mechanisms are frequently associated with details that people are embarrassed to divulge. However, such information can often be derived from the data that have been given. As an example, low-income people may be embarrassed to disclose their income but may disclose their highest level of education. Data users may then use the supplied educational information to acquire an insight into the income. Let us assume that the variable Y is the complete dataset, then: $Y = \{Y_o, Y_m\}$.

Here Y_o is the observed component of Y while Y_m is the missing component of Y. Any scenario whereby certain or all feature variables within a dataset have missing data entries or contain data entries which are not exactly characterized within the bounds of the problem domain is termed missing data (Rubin 1978). The presence of missing data leads to several issues in a variety of sectors that depend on the availability of complete and quality data. This has resulted in different methods being introduced with their aim being to address the missing data problem in varying disciplines (Rubin 1978; Allison 2000). Handling missing data in an acceptable way

is dependent upon the nature of the missingness. There are currently four missing data mechanisms in the literature and these are missing completely at random (MCAR), missing at random (MAR), a non-ignorable case or missing not at random (MNAR) and missing by natural design (MBND).

1.4.1 Missing Completely at Random (MCAR)

The MCAR case is observed when the possibility of a feature variable having missing data entries is independent of the feature variable itself or of any of the other feature variables within the dataset. Essentially, this means that the missing data entry does not depend on the feature variable being considered or any of the other feature variables in the dataset. This relationship is expressed mathematically as Little and Rubin (2014):

$$P(M|Y_{\mathrm{o}}, Y_{\mathrm{m}}) = P(M) \tag{1.1}$$

where $M \in \{0, 1\}$ represents an indication of the missing value. $M = 1$ if Y is known and $M = 0$ if Y is unknown/missing. Y_{o} represents the observed values in Y while Y_{m} represents the missing values of Y. From Eq. (1.1), the probability of a missing entry in a variable is not related to Y_{o} or Y_{m}. For instance, let us assume that in modelling software defects in relation to development time, if the missingness is in no way linked to the missing values of the rate of defects itself and at the same time not linked to the values of the development time, the data is said to be MCAR. Researchers have successfully addressed cases where the data is MCAR. Silva-Ramirez et al. (2011) successfully applied multilayer perceptrons (MLPs) for missing data imputation in datasets with missing values. Other research work done on this mechanism could be found in Pigott (2001), Nishanth and Ravi (2013).

1.4.2 Missing at Random (MAR)

The MAR case is observed when the possibility of a specific feature variable having missing data entries is related to the other feature variables in the dataset. However, this missing data does not depend on the feature variable itself. MAR means the missing data in the feature variable is conditional on any other feature variable in the dataset but not on that being considered (Scheffer 2000). For example, consider a dataset with two related variables, monthly expenditure and monthly income. Assume for instance that all high-income earners deny revealing their monthly expenditures while low-income earners do provide this information. This implies that in the dataset, there is no monthly expenditure entry for high-income earners, while for low-income earners, the information is available. The missing monthly income entry is linked

to the income earning level of the individual. This relationship can be expressed mathematically as Marwala (2009):

$$P(M|Y_o, Y_m) = P(M|Y_o) \tag{1.2}$$

where $M \in \{0, 1\}$ is the missing data indicator, and $M = 1$, if Y is known, with $M = 0$ if Y is unknown/missing. Y_o represents the observed values in Y while Y_m represents the missing values of Y. Equation (1.2) indicates that the probability of a missing entry given an observable entry and a missing entry is equivalent to the probability of the missing entry given the observable entry only. Considering the example described in Sect. 1.1.3.1, the software defects might not be revealed because of a certain development time. Such a scenario points to the data being MAR. Several studies have been conducted in the literature where the missing data mechanism is MAR, for example Nelwamondo et al. (2007b) performed a study to compare the performance of expectation maximization and a GA-optimized AANN and it was revealed that the AANN is a better method than the expectation maximization. Further research on this mechanism was performed in Garca-Laencina et al. (2009), Poleto et al. (2011), Liu and Brown (2013).

1.4.3 Non-ignorable Case or Missing not at Random (MNAR)

The third missing data mechanism is the missing not at random or non-ignorable case. The MNAR case is observed when the possibility of a feature variable having a missing data entry depends on the value of the feature variable itself irrespective of any alteration or modification to the values of other feature variables in the datasets (Allison 2000). In scenarios such as these, it is impossible to estimate the missing data by making use of the other feature variables in the dataset since the nature of the missing data is not random. MNAR is the most challenging missing data mechanism to model and these values are quite tough to estimate (Rubin 1978). Let us consider the same scenario described in the previous subsection. Assume for instance that some high-income earners do reveal their monthly expenditures while others refuse, and the same for low-income earners. Unlike the MAR mechanism, in this instance the missing entries in the monthly expenditure variable cannot be ignored because they are not directly linked to the income variable or any other variable. Models developed to estimate this kind of missing data are very often not biased. A probabilistic formulation of this mechanism is not easy because the data in the mechanism is neither MAR nor MCAR.

1.4.4 Missing by Natural Design (MBND)

This is a mechanism whereby the missing data occurs because it cannot be measured physically (Marwala 2009). It is impossible to measure these data entries; however, they are quite relevant in the data analysis procedure. Overcoming this problem requires that mathematical equations be formulated. This missing data mechanism mainly applies to mechanical engineering and natural science problems. Therefore, it will not be used in this thesis for the problem under consideration.

1.5 Missing Data Patterns

The way in which missing data occurs can be grouped into three patterns given by Tables 1.1, 1.2, 1.3. Table 1.1 depicts a univariate pattern which is a scenario described by the presence of missing data in only one feature variable as seen in column I7. Table 1.2 depicts an arbitrary missing data pattern, which is a scenario whereby the missing data occurs in a distributed and random manner. The last pattern is the monotone missing data pattern which is shown in Table 1.3. This pattern is also referred to as a uniform pattern as it occurs in cases whereby the missing data can be present in more than one feature variable and, it is easy to understand and recognize (Ramoni and Sebastiani 2001).

The missing data pattern considered in this book is the arbitrary pattern and the mechanisms are the missing at random and missing completely at random mechanisms.

Table 1.1 Univariate missing data pattern

Sample	I1	I2	I3	I4	I5	I6	I7
1	0.38	0.18	0.20	0.19	0.75	0.67	0.96
2	0.69	0.11	0.08	0.41	0.65	0.63	?
3	0.17	0.79	0.66	0.53	0.95	0.43	?
4	0.19	0.24	0.15	0.91	0.46	0.82	?

Table 1.2 Arbitrary missing data pattern

Sample	I1	I2	I3	I4	I5	I6	I7
1	0.38	?	0.20	0.19	0.75	0.67	0.96
2	0.69	0.11	0.08	0.41	?	0.63	0.04
3	0.17	0.79	?	0.53	0.95	0.43	0.054
4	?	0.24	0.15	0.91	0.46	0.82	?

Table 1.3 Monotone missing data pattern

Sample	I1	I2	I3	I4	I5	I6	I7
1	0.38	0.18	0.20	0.19	0.75	0.67	?
2	0.69	0.11	0.08	0.41	0.65	?	?
3	0.17	0.79	0.66	0.53	?	?	?
4	0.19	0.24	0.15	?	?	?	?

1.6 Classical Missing Data Techniques

Depending on how data goes missing in a dataset, there currently exist several data imputation techniques that are being used in statistical packages (Yansaneh et al. 1998). These techniques include basic approaches such as casewise data deletion and move on to approaches that are characterized by the application of more refined artificial intelligence and statistical methods. The subsections that follow present some of the most commonly applied missing data imputation methods. We begin with basic and naive approaches and carry on presenting more complex and competent mechanisms. There are a variety of classical missing data imputation techniques courtesy of their simplicity and ease of implementation. The techniques presented in this section are listwise or casewise deletion, pairwise deletion, mean substitution, stochastic imputation with expectation maximization, hot and cold deck imputation, multiple imputation and regression methods.

1.6.1 Listwise or Casewise Deletion

A lot of statistical approaches will get rid of an entire record if it is seen that any of the columns in the record has a missing data entry. Such an approach is termed casewise or listwise data deletion and is a scenario, whereby in the event of any of the columns in a record having a missing value for a feature variable, the entire record is deleted from the dataset. Listwise data deletion is the easiest and most basic way to handle the problem of missing data as well as being the least recommended option for the problem as it tends to significantly reduce the number of records in the dataset which are necessary for the data analysis task, and by so doing, it reduces the accuracy of the findings from the analysis of the data. Applying this technique is a possibility if the ratio of records with missing data to records with complete data is very small. If this is not the case, making use of this approach may result in the estimates of the missing data being biased.

1.6.2 Pairwise Deletion

The pairwise data deletion approach operates by performing the analysis required by using pairwise data. The implication of this is that records with missing data will be used in analysis tasks if and only if the feature variable with the missing data in that record is not needed. The benefit from doing this is the number of records used for analysis will often be more than if one were to use the listwise data deletion approach. However, this approach results in biased missing data estimates, which is a bad outcome for the dataset when the missing data mechanism is MAR or MNAR. On the contrary, this approach is quite competent if the data is MCAR.

1.6.3 Mean–Mode Substitution

This approach works by substituting the missing data entries with the value of the mean or mode of the available data in the feature variable(s). It has a high possibility of yielding biased estimates of the missing data just like the pairwise data deletion approach (Allison 2002), and, it is not a highly recommended approach to solve this problem. For dataset feature variables with continuous or numerical values, the missing data entries are substituted by the mean of the related variables. On the other hand, for feature variables with categorical or nominal values, the missing data are substituted by using the most common or modal value of the respective feature variable (Little and Rubin 2014). These techniques are most effective when the data is assumed to be MCAR. Mean–mode substitution has been used with success in previous research (Kalousis and Hilario 2000; Pérez et al. 2002).

1.6.4 Imputation

Imputation in statistics is the process of replacing missing data with substituted values, thus addressing the pitfalls caused by the presence of missing data. Imputation techniques can be categorized into single and multiple imputation. Single imputation comprises of replacing a missing value with only one estimated value while with multiple imputation, each missing entry is replaced with a set of M estimated values.

1.6.4.1 Single-Imputation-Based Techniques

Expectation Maximization
Expectation maximization (EM) is a model-based imputation technique designed for parameter estimation in probabilistic methods with missing data (Dempster et al. 1997). EM is comprised of a two-step process, with the first step, known as the

E-step, being the process of estimating a probability distribution over completions of missing data given the model. The M-step is the second step, and identifies parameter estimates that maximize the complete data log-likelihood obtained from the E-step. The M-step has as stopping criteria either convergence being attained, or number of iterations being reached (Dempster et al. 1997). Details about this algorithm can be found in Dempster et al. (1997). It is applicable in both single and multiple imputation procedures and has been shown to perform better than the techniques described above (Twala 2009; Nelwamondo et al. 2007b; Marwala 2009). This technique works best on the assumption that the data is MAR.

Hot Deck and Cold Deck Imputation
These methods fall under the category of donor-based imputation techniques. Donor-based imputation entails substituting missing entries with data from other records.

Hot Deck imputation is a method in which missing data entries are filled in with values from other records. This is achieved by Little and Rubin (2014):

- Splitting instances into clusters of similar data. This can be done using methods such as k-nearest neighbour.
- Missing entries are replaced with values from instances that fall in the same class.

Cold deck imputation on the other hand entails substituting the missing data by a constant value obtained from other sources (Little and Rubin 2014). Hot or cold deck imputation are popular owing to their simplicity as well as there being no need to make strong assumptions about the model used to fit the data. It is worth mentioning though that the imputation strategy does not necessarily lead to a reduction in bias, in relation to the incomplete dataset.

1.6.4.2 Multiple-Imputation-Based Techniques

Multiple Imputation (MI) is an approach whereby missing data entry are substituted with a set of M approximated values. In Rubin (1978), MI is described in three consecutive steps. The first step entails substituting the missing data entries in the dataset with M different values. This results in M different datasets being obtained with complete records. Step two of the process entails analysing the different M datasets with complete records by applying complete data analysis techniques. Finally, in step three, the results from the M datasets are combined based on the analysis done in step two. This result referred to from step two indicates which of the M datasets obtains the best state of the missing data entries or yields the better conclusions and inferences. This approach is a better one than the single imputation approaches. It also makes use of the advantages of the EM and likelihood estimation approaches, with the popular traits of the hot deck imputation method to obtain new data matrices that will be processed (Scheffer 2000; Allison 2002). The three steps mentioned above can be further explained in the points below:

- Make use of a reliable model that incorporates randomness to estimate the missing values;
- Generate M complete datasets by repeating the process M times;
- Apply complete data analysis algorithms to perform analysis of the components of the datasets obtained;
- From the M complete datasets obtained, calculate the overall value of the estimates by averaging the values from the M datasets.

This method depends on the assumption that the data is MAR and originates from a multivariate uniform distribution.

1.6.5 Regression Methods

This approach involves generating a regression equation that depends on a record with all the data available for a given feature variable. To achieve this, the feature variable with the missing data is considered as being the dependent variable in the equation, with all the other feature variables considered the independent variables (predictors). Records with missing values have these values being obtained as estimates from using the regression equation with the feature variable of interest being the output and all the others being the model equation inputs (Little and Rubin 2014).

The process of generating regression equations is done repeatedly and in order, for the feature variables with missing data entries until all such missing entry values are estimated and substituted. This means that a feature variable vj having missing data entries will have a model created for it using records with known values for the other variables. Applying this method to estimate the missing data entry in sample 2 of Table 1.2, the regression equation that is to be fitted will consider the variables $I1$, $I2$, $I3$, $I4$, $I6$ and $I7$. This results in the equation below:

$$I5 = i_1 I1 + i_2 I2 + i_3 I3 + i_4 I4 + i_6 I6 + i_7 I7 + \epsilon \tag{1.3}$$

The regression equation constitutes the estimate terms i_i as well as the error, ϵ. It can subsequently be applied to approximate missing data entries by replacing $I1$, $I2$, $I3$, $I4$, $I6$ and $I7$ by their known values.

1.7 Machine Learning Approaches to Missing Data

Several approaches in computational intelligence have been developed to address the problems of missing data and the drawbacks of statistical techniques covered in Sect. 1.1.5. Some of these techniques are tree based or based on biological concepts.

1.7.1 Decision Trees

Decision trees are supervised learning models aimed at separating data into consistent clusters for classification or regression analysis. A decision tree is acyclic by default and consists of a root node, leaf nodes, internal nodes and edges. The root node indicates the onset of the tree with the leaf nodes representing the end of the tree which either presents the outcome or the class label. The internal node stores details about the attribute used for splitting data at each node. The edges are links between the nodes and contain details about splits. The outcome of a record is obtained by processing the information across the tree from the root node to the leaf node (Twala and Cartwright 2010).

Using decision trees to perform the missing data estimation task entails building a tree for each feature variable with missing data entries. This feature variable is considered the class-label with the actual class label forming part of the input feature set. The building of the tree is done using records with known class labels and the missing data entries are substituted with a corresponding tree (Twala and Cartwright 2010). Let us say, for example that a dataset has attributes *I1*, *I2*, *I3* and a class-label *L*, which is obtained as such: *L (I1, I2, I3)*. Assume *I1* has missing values, *I1* will be considered the class-label while *L* will be regarded as one of the input feature variables. The new class-label will be obtained by: *I1(L, I2, I3)*. If *I2* has the missing data, the new output is obtained using: *I2(I1, L, I3)* and *I2* is considered the new class-label. This procedure is executed until all feature variables with missing data are complete.

The strategy described above operates the way a single imputation method does, and has been applied successfully (Twala and Cartwright 2010); Twala et al. 2008; Twala and Phorah 2010). It is unknown whether the sequence in which the missing values are substituted influence the estimates and, the method works best when the data is assumed to be MCAR.

1.7.2 Artificial Neural Networks (ANNs)

An ANN is referred to as a probabilistic model that is used to process data in the way that the biological nervous system does (Abdella and Marwala 2005b). The human brain is a very good example of such a system. It can also be defined as a collection and combination of elements whose performance is dependent on the elements, which in a neural network are neurons. The neurons are connected to one another and the nature of these connections also influence the performance of the network. The fundamental processing unit of a neural network is what is referred to as a neuron (Ming-Hau 2010). A neural network comprises of four important components, these being Haykin (1999):

1. At any stage in the biorhythm of the neural network, each neuron has an activation function value;

2. Each neuron is connected to every other neuron, and these connections are what determine how the activation level of a neuron becomes the input for another neuron. Each of these connections is allocated a weight value;
3. At a neuron, an activation function is applied to all the incoming inputs to generate a new input for neurons in the output layer or subsequent hidden layers, and;
4. A learning algorithm that is used to adjust the weights between neurons when given an input–output pairing.

A predominant feature of a neural network is the capability it possesses to accommodate and acclimatize to its environment with the introduction of new data and information. It is with this in mind that learning algorithms were created, and they are very important in determining how competent a neural network will and can be. A neural network is applicable in several domains such as in the modelling of problems of a highly complicated nature because of how relatively easy it is for them to derive meaning from complex data, identify patterns and trends, which are very convoluted for other computational models (Abdella and Marwala 2005a). Trained neural networks are applicable in prediction tasks where the aim is to determine the outcome of a new input record after having been presented with similar information during the training process (Abdella and Marwala 2005a). Their inherent ability to adapt with ease to being presented with new non-linear information makes them favourable to be used to solve non-linear models.

Neural networks have been observed to be highly efficient and capable in obtaining solutions to a variety of tasks with most of these being forecasting and modelling, expert systems and signal processing tasks (Haykin 1999). The organization of the neurons in a neural network affects the processing capability of the said network as well as an influence on the way in which information moves between the layers and neurons.

1.7.2.1 Auto-associative Neural Network

Autoencoder networks are defined as networks that try to regenerate the inputs as outputs in the output layer (Lu and Hsu 2002), and this guarantees that the network predicts new input values as the outputs when presented with new inputs. These autoencoders are made up of one input layer and one output layer both having the same number of neurons, resulting in the term auto-associative (Lu and Hsu 2002). Besides these two layers, a narrow-hidden layer exists, and it is important that this layer contains fewer neurons than there are in the input and output layers with the aim of this being to apply encoding and decoding procedures when trying to find a solution to a given task (Mistry et al. 2008). These networks have been used in a variety of applications (Hines et al. 1998; Atalla and Inman 1998; Smauoi and Al-Yakoob 2003; Marwala 2001; Marwala and Chakraverty 2006; Marwala 2013). The main concept that defines the operation of autoencoder networks is the notion that the mapping from the input to the output, $x^{(i)} \rightarrow y^{(i)}$ stores very important information and the key inherent architecture present in the input, $x^{(i)}$, that is contrarily hypothetical

(Hines et al. 1998; Leke and Marwala 2016). An autoencoder takes x as the input and transcribes it into y, which is a hidden representation of the input, by making use of a mapping function, f_θ, that is deterministic. This function is expressed as Leke and Marwala (2016), Issacs (2014):

$$f_\theta(x) = s(Wx + b). \tag{1.4}$$

The parameter, θ, is made up of the weights W and biases b. s represents the sigmoid activation function which is given by

$$s = \frac{1}{1 + e^{-x}}. \tag{1.5}$$

y is then mapped to a vector, z, representing the reconstruction of the inputs from this hidden representation. This reconstructed vector is obtained by using the following equations (Leke and Marwala 2016):

$$z = g_{\theta'}(y) = s\left(W'y + b'\right), \tag{1.6}$$

or

$$z = g_{\theta'}(y) = W'y + b'. \tag{1.7}$$

In the above equations, θ' is made up of the transpose of the weights matrix and the vector of biases from Eq. (1.4). Equation (1.6) is the autoencoder output function with a sigmoid transfer function Eq. (1.5), while Eq. (1.7) is the linear output equation. After these operations, the network can then be fine-tuned by applying a supervised learning approach (Isaacs 2014). The network is said to have tied weights when the weights matrix is transposed. The vector z is not described as an exacting transformation of x, but rather as the criteria of a distribution $p(X|Z = z)$ in probabilistic terms. The hope is that these criteria will result in x with high probability (Isaacs 2014). The resulting equation is as follows (Leke and Marwala 2016):

$$p(X|Y = y) = p(X|Z = g_{\theta'}(y)). \tag{1.8}$$

This leads to an associated regeneration disparity that forms the basis for the objective function to be used by the optimization algorithm. This disparity is usually represented by Leke and Marwala (2016):

$$L(x, z) \propto -\log p(x|z). \tag{1.9}$$

α indicates proportionality. The equation above could be represented by Sartori et al. (2005):

$$\delta_{AE}(\theta) = \sum\nolimits_t L\left(x^{(t)}, g_\theta\left(f_\theta\left(x^{(t)}\right)\right)\right). \tag{1.10}$$

Autoencoder networks have been used in a variety of application areas, by several researchers, with the focus being on the problem of missing data (Abdella and Marwala 2005a; Baek and Cho 2003; Tim et al. 2004; Brain et al. 2006).

1.7.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a classification model capable of solving both linear and non-linear complex problems (Steeb 2008; Hastie et al. 2008). In linear problems, the model tries to identify a maximal marginal hyper-plane with the greatest margin. This hyper-plane must obey the following expression (Steeb 2008; Hastie et al. 2008):

$$f(x) = \begin{cases} 1, w.x + b \geq 1 \\ -1, w.x + b \leq 1 \end{cases}, \tag{1.11}$$

where w and x represent the weight and input vectors, respectively, and b indicates the bias. Larger margins are preferable as they increase the accuracy of classifications. In scenarios where the data dimensions are linearly inseparable, the data requires transformation into higher dimensions. The model identifies an optimal hyper-plane capable of separating the variables of the classes in the new high dimensional space. Kernel functions which are used to map the original data into higher dimensions could be expressed mathematically as Suykens and Vandewalle (1999), Hearst et al. (1998), Burges (1998):

$$K\left(x_i, x_j\right) = \varphi(x_i).\varphi\left(x_j\right), \tag{1.12}$$

where $\varphi(x_i)$ and $\varphi\left(x_j\right)$ are the non-linear mapping functions. Some frequently used kernel functions are Suykens and Vandewalle (1999), Hearst et al. (1998), Burges (1998): the polynomial, sigmoid and Gaussian radial functions.

1.8 Machine Learning Optimization

As previously mentioned, constructing models for handling missing data can be complex and computationally expensive. Successful models employ an optimization technique to construct a model that best fits the training set. In this section, we highlight various strategies that have been employed as optimization techniques in missing data problems.

1.8.1 Genetic Algorithm (GA)

Genetic algorithm (GA) is an evolutionary computational technique designed to search for global optimum solutions to complex problems. It was inspired by Darwin's theory of natural evolution. Genetic algorithms use the notion of survival of the fittest, where the strongest individuals are selected for reproduction until the best solution is found or the number of cycles is completed. The processes involved in a genetic algorithm are selection, crossover, mutation and recombination. The selection process involves selecting the strongest parent individuals using a probabilistic technique for the crossover process. During crossover, a crossover point is chosen between the parent individuals. Data is swapped or exchanged from the starting point of an individual to the crossover point. The outcome is two children. At this point, if the children are stronger than their parents, they can be used to replace one or both parents. Mutation is performed by randomly selecting a gene and inverting it. Mutation is given a low probability value, which means that it occurs less than the crossover process. Recombination process evaluates the fitness of the children or newly generated individual to determine if they can be merged with the current population.

As previously mentioned, genetic algorithms have been applied to optimize neural networks (Nelwamondo et al. 2007b; Marwala 2009; Brain et al. 2006) by searching for individuals that maximize the objective function, prior to imputation. This algorithm is classified within the domain of computational intelligence as per Marwala (2010) and has been further used to address the missing data problem (Abdella and Marwala 2005b; Brain et al. 2006; Abdella 2005).

1.8.2 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a search technique based on a collective behaviour of birds within a flock. The goal of the technique is to simulate the random and unpredictable movement of a flock of birds, with the intent of finding patterns that govern the birds' ability to move at the same time and change direction whilst regrouping in an optimal manner (Kennedy and Eberhart 1995; Engelbrecht 2006).

PSO particles move through a search space. A change in position of the particles within the space is based on a socio-psychological tendency of each particle copying or emulating the success of the neighbouring particle and their own success. These changes are influenced by the knowledge or experience of surrounding particles and the particle themselves. Therefore, the search behaviour of one particle is affected by the behaviour of other particles or itself. The collective behaviour of particles within a swarm permits the discovery of global optimal solutions in high dimensional search spaces (Kennedy and Eberhart 1995; Engelbrecht 2006). This algorithm is also classified within the domain of computational intelligence as per Marwala (2010) and has also been used to address the missing data problem in Leke et al. (2014).

1.8.3 Simulated Annealing (SA)

Simulated annealing (SA) is an optimization technique that uses the concept of cooling metal substances with the goal of condensing matter into a crystalline solid. It can be considered as a procedure used to find an optimal solution. The main characteristics of simulated annealing are that

- it can find a global optimum solution;
- it is easy to implement for complex problems and,
- it solves complex problems and cost functions with various numbers of variables.

The drawbacks of simulated annealing are Marwala and Lagazio (2011) as follows:

- It takes many iterations to find an optimal solution;
- The cost function is computationally expensive to estimate;
- It is inefficient if there are many local minimum points;
- It depends on the nature of the problem, and;
- It is difficult to determine the temperature cooling technique.

Research in SA has shown that it performs marginally better than GA and PSO techniques (Marwala 2009; Marwala and Lagazio 2011). However, for problems involving datasets with high dimensions, GA and PSO are recommended over SA because of these drawbacks. In addition, this algorithm is classified within the domain of computational intelligence as per Marwala (2010) and has been further used to address the missing data problem (Leke et al. 2014).

1.9 Challenges to Missing Data Imputation

A lot of the techniques mentioned above do not result in optimal solutions to the missing data issue. However, they lead to biases except in a few unique and specialized scenarios. Listwise deletion technique has been observed to do well for the MCAR mechanism and specifically for large databases. However, it is not easy to establish that the data observes the MCAR mechanism. Using the mean substitution technique is unjustifiable as it alters the variance for any condition. Many issues arise while imputing missing data. Some of these entails realizing which technique to use based on the imputation task. For instance, is the choice of a data modelling technique important? It was found in a literature review that the expectation maximization (EM) method still gets used a lot (Shinozaki and Ostendorf 2008; Stolkin et al. 2008). Therefore, the question that arises is: is the EM algorithm the state of art? In recent times, models that use a combination of computational intelligence and optimization algorithms have been reported in the literature (Abdella and Marwala 2005b; Leke and Marwala 2016; Abdella 2005) especially for highly non-linear scenarios. The question that arises is: Which computational intelligence method is

best suited to missing data estimation? In addition, which optimization method is ideal for the missing data problem? Avoiding this question has led researchers like Twala (2005) to adopt a multiple model approach, especially in scenarios where the technique to use is not obvious. The question that then arises is: What is the best way to combine different models to achieve an efficient missing data estimation result? These are just a few questions that need answering.

1.10 Conclusions

This chapter begins by presenting some historical overview of missing data estimation methods followed by a background summary of the missing data mechanisms and patterns, as well as classical techniques used for handling missing data. We also discussed modern approaches for handling missing data and their applications. Furthermore, we present some challenges common in missing data imputation approaches. The chapter is important because it illustrates the importance of understanding missing data and how various methods have evolved over the years to handle the problem.

References

Abdella, M., & Marwala, T. (2005a). The use of genetic algorithms and neural networks to approximate missing data in database. *24*, 577–589.

Abdella, M. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. Unpublished master's thesis, University of the Witwatersrand, Johannesburg.

Abdella, M., & Marwala, T. (2005b). Treatment of missing data using neural networks. In: *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 1, 598–603

Allison, P. D. (2000). Multiple imputation for missing data. *Sociological Methods & Research, 28*(3), 301–309.

Allison, P. D. (2002). *Missing data*. Thousand Oaks: Sage Publications.

Atalla, M. J., & Inman, D. J. (1998). On model updating using neural networks. *Mechanical Systems and Signal Processing, 12,* 135–161.

Baek, K., & Cho, S. (2003). Bankruptcy prediction for credit risk using an auto-associative neural network in Korean firms. In: *IEEE Conference on Computational Intelligence for Financial Engineering*, pp. 25–29, Hong Kong, China.

Brain, L. B., Marwala, T., & Tettey, T. (2006). Autoencoder networks for HIV classification. *Current Science, 91*(11), 1467–1473.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*(2), 121–167.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1997). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society, 39*(1), 1–38.

Dhlamini, S. M., Nelwamondo, F. V., & Marwala, T. (2006). Condition monitoring of HV bushings in the presence of missing data using evolutionary computing. *Transactions on Power Systems, 1*(2), 280–287.

Engelbrecht, A. P. (2006). Particle swarm optimization: Where does it belong? In: *Proceedings of IEEE Swarm Intelligence Symposium*, pp. 48–54.

Faris, P. D., Ghali, W. A., Brant, R., Norris, C. M., Galbraith, P. D., & Knudtson, M. L. (2002). Multiple imputation versus data enhancement for dealing with missing data in observational health care outcome analyses. *Journal of Clinical Epidemiology, 55*(2), 184–191.

Gabrys, B. (2002). Neuro-fuzzy approach to processing inputs with missing values in pattern recognition problems. *International Journal of Approximate Reasoning, 30,* 149–179.

Garca-Laencina, P., Sancho-Gmez, J., Figueiras-Vidal, A., & Verleysen, M. (2009). K nearest neighbours with mutual information for simultaneous classification and missing data imputation. *Neurocomputing, 72*(7–9), 1483–1493.

Hastie, T., Tibshirani, R., & Friedman, J. (2008). *The elements of statistical learning: Data mining, inference, and prediction*. New York: Springer.

Haykin, S. (1999). *Neural networks* (2nd ed.). New Jersey: Prentice-Hall.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications, 13*(4), 18–28.

Hines, J. W., Robert, E. U., & Wrest, D. J. (1998). Use of autoassociative neural networks for signal validation. *Journal of Intelligent and Robotic Systems, 21*(2), 143–154.

Ho, P., Silva, M. C. M., & Hogg, T. A. (2001). Multiple imputation and maximum likelihood principal component analysis of incomplete multivariate data from a study of the ageing of port. *Chemometrics and Intelligent Laboratory Systems, 55*(1–2), 1–11.

Hui, D., Wan, S., Su, B., Katul, G., Monson, R., & Luo, Y. (2004). Gap-filling missing data in eddy covariance measurements using multiple imputation (MI) for annual estimations. *Agricultural and Forest Meteorology, 121*(1–2), 93–111.

Isaacs, J. C. (2014). Representational learning for sonar ATR. In SPIE Defense + Security. In: *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XIX*. International Society for Optics and Photonics, vol. 9072, p. 907203. https://doi.org/10.1117/12.2053057.

Junninen, H., Niska, H., Tuppurainen, K., Ruuskanen, J., & Kolehmainen, M. (2004). Methods for imputation of missing values in air quality data sets. *Atmospheric Environment, 38*(18), 2895–2907.

Kalousis, A., & Hilario, M. (2000). Supervised knowledge discovery from incomplete data. In: *Proceedings of the 2nd International Conference on Data Mining*. WIT Press. http://cui.unige.ch/AI-group/research/metal/Papers/missingvalues.ps. Accessed Oct 2016.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization (PSO). In: *Proceedings of IEEE International Conference on Neural Networks (ICNN)*, Perth, Australia, vol. 4, pp. 1942–1948.

Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In: *International Conference in Swarm Intelligence*. Springer International Publishing, pp. 259–270.

Leke, C., Twala, B., & Marwala, T. (2014). Modeling of missing data prediction: Computational intelligence and optimization algorithms. In: *2014 IEEE International Conference on Systems, Man and Cybernetics (SMC),* pp. 1400–1404.

Little, R., & Rubin, D. (2014). *Statistical analysis with missing data* (Vol. 333). New York: Wiley.

Little, R. J. A., & Rubin, D. B. (1987). *Statistical analysis with missing data*. New York: Wiley.

Liu, Y., & Brown, S. D. (2013). Comparison of five iterative imputation methods for multivariate classification. *Chemometrics and Intelligent Laboratory Systems, 120,* 106–115.

Lu, P. J., & Hsu, T. C. (2002). Application of autoassociative neural network on gas-path sensor data validation. *Journal of Propulsion and Power, 18*(4), 879–888.

Marwala, T. (2010). *Finite element model updating using computational intelligence techniques: Applications to structural dynamics*. Heidelberg: Springer.

Marwala, T., & Lagazio, M. (2011). *Militarized conflict modeling using computational intelligence techniques*. London: Springer.

Marwala, T. (2009). *Computational intelligence for missing data imputation: Estimation and management knowledge optimization techniques*. Hershey, New York: Information Science Reference.

Marwala, T. (2001). Probabilistic fault identification using a committee of neural networks and vibration data. *Journal of Aircraft, 38*(1), 138–146.

Marwala, T., & Chakraverty, S. (2006). Fault classification in structures with incomplete measured data using autoassociative neural networks and genetic algorithm. *Current Science, 90*(4), 542–549.

Marwala, T. (2013). *Economic modelling using artificial intelligence methods*. London: Springer.

Ming-Hau, C. (2010). Pattern recognition of business failure by autoassociative neural networks in considering the missing values. *International Computer Symposium (ICS)* (pp. 711–715). Taiwan: Taipei.

Mistry, J., Nelwamondo, F., & Marwala, T. (2008). Estimating missing data and determining the confidence of the estimate data. In: *Seventh International Conference on Machine Learning and Applications*, San Diego, CA, USA, pp. 752–755.

Nelwamondo, F. V., Mohamed, S., & Marwala, T. (2007a). Missing data: A comparison of neural network and expectation maximization techniques. *Current Science, 93*(11), 1514–1521.

Nelwamondo, F. V., & Marwala, T. (2007a). Handling missing data from heteroskedastic and nonstationary data. *Lecture Notes in Computer Science, 4491*(1), 1297–1306

Nelwamondo, F. V., & Marwala, T. (2007b). Rough set theory for the treatment of incomplete data. In: *Proceedings of the IEEE Conference on Fuzzy Systems*, London, UK, pp. 338–343.

Nelwamondo, F. V., & Marwala, T. (2007c). Fuzzy ARTMAP and neural network approach to online processing of inputs with missing values. *SAIEE Africa Research Journal, 98*(2), 45–51.

Nelwamondo, F. V., Mohamed, S., & Marwala, T. (2007b). Missing data: A comparison of neural network and expectation maximisation techniques. *Current Science, 93*(12), 1514–1521.

Nelwamondo, F. V., & Marwala, T. (2008). Techniques for handling missing data: applications to online condition monitoring. *International Journal of Innovative Computing, Information and Control, 4*(6), 1507–1526.

Nishanth, K. J., & Ravi, V. (2013). A computational intelligence based online data imputation method: An application for banking. *Journal of Information Processing Systems, 9*(4), 633–650.

Pérez, A., Dennis, R. J., Gil, J. F. A., Róndon, M. A., & López, A. (2002). Use of the mean, hot deck and multiple imputation techniques to predict outcome in intensive care unit patients in Colombia. *Journal of Statistics in Medicine, 21*(24), 3885–3896.

Pigott, T. D. (2001). A review of methods for missing data. *Educational Research and Evaluation, 7*(4), 353–383.

Poleto, F. Z., Singer, J. M., & Paulino, C. D. (2011). Missing data mechanisms and their implications on the analysis of categorical data. *Statistics and Computing, 21*(1), 31–43.

Polikar, R., De Pasquale, J., Mohammed, H. S., Brown, G., & Kuncheva, L. I. (2010). Learn++mf: A random subspace approach for the missing feature problem. *Pattern Recognition, 43*(11), 3817–3832.

Ramoni, M., & Sebastiani, P. (2001). Robust learning with missing data. *Journal of Machine Learning, 45*(2), 147–170.

Rubin, D. (1978). Multiple imputations in sample surveys-a phenomenological Bayesian approach to nonresponse. *Proceedings of the survey research methods section of the American Statistical Association, 1,* 20–34.

Sartori, N., Salvan, A., & Thomaseth, K. (2005). Multiple imputation of missing values in a cancer mortality analysis with estimated exposure dose. *Computational Statistics & Data Analysis, 49*(3), 937–953.

Scheffer, J. (2000). Dealing with missing data. Research Letters in the Information and Mathematical Sciences. 3:153–160. (last accessed: 18-March-2016). [Online]. Available: http://www.massey.ac.nz/wwiims/research/letters.

Shinozaki, T., & Ostendorf, M. (2008). Cross-validation and aggregated EM training for robust parameter estimation. *Computer Speech & Language, 22*(2), 185–195.

Silva-Ramirez, E.-L., Pino-Mejias, R., Lopez-Coello, M., & Cubiles-de-la Vega, M.-D. (2011). Missing value imputation on missing completely at random data using multilayer perceptrons. *Neural Networks, 24*(1), 121–129.

Smauoi, N., & Al-Yakoob, S. (2003). Analyzing the dynamics of cellular flames using karhunen-loeve decomposition and autoassociative neural networks. *Society for Industrial and Applied Mathematics, 24,* 1790–1808.

Steeb, W.-H. (2008). *The Nonlinear Workbook*. Singapore: World Scientific.

Stolkin, R., Greig, A., Hodgetts, M., & Gilby, J. (2008). An EM/E-MRF algorithm for adaptive model-based tracking in extremely poor visibility. *Image and Vision Computing, 26*(4), 480–495.

Suykens, J. A. K., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters, 9*(3), 293–300.

Tim, T., Mutajogire, M., & Marwala, T. (2004). *Stock market prediction using evolutionary neural networks* (pp. 123–133). PRASA: Fifteenth Annual Symposium of the Pattern Recognition.

Tremblay, M. C., Dutta, K., & Vandermeer, D. (2010). Using data mining techniques to discover bias patterns in missing data. *Journal of Data and Information Quality, 2*(1), 1–19.

Twala, B. (2009). An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence, 23*(5), 373–405.

Twala, B., & Cartwright, M. (2010). Ensemble missing data techniques for software effort prediction. *Intelligent Data Analysis., 14*(3), 299–331.

Twala, B. E. T. H., Jones, M. C., & Hand, D. J. (2008). Good methods for coping with missing data in decision trees. *Pattern Recognition Letters, 29*(7), 950–956.

Twala, B., & Phorah, M. (2010). Predicting incomplete gene microarray data with the use of supervised learning algorithms. *Pattern Recognition Letters, 31,* 2061–2069.

Twala, B. E. T. H. (2005). Effective techniques for handling incomplete data using decision trees. Unpublished doctoral dissertation, The Open University, UK.

Wang, S. (2005). Classification with incomplete survey data: A Hopfield neural network approach. *Computers & Operations Research, 24,* 53–62.

Yansaneh, I. S., Wallace, L. S., & Marker, D. A. (1998). Imputation methods for large complex datasets: An application to the Nehis. In: *Proceedings of the Survey Research Methods Section*, pp. 314–319.

Yu, S., & Kobayashi, H. (2003). A hidden semi-Markov model with missing data and multiple observation sequences for mobility tracking. *Signal Processing, 83*(2), 235–250.

Chapter 2
Introduction to Deep Learning

2.1 Introduction

Deep learning, also referred to as hierarchical learning or deep structured learning, forms a part of the expansive family of machine learning techniques that rely on learning data representations, contrary to conventional task-specific methods. The learning of data representations could be by the way of semi-supervised, supervised or unsupervised approaches. Deep learning models mimic the information processing and communication patterns observed in the biological nervous system such as neural coding that makes an attempt to define and describe the interrelationships that exist between several stimuli and associated neuronal responses in the brain.

Deep learning is made up of a variety of techniques in the field of machine learning that make use of a deluge of non-linear nodes which are arranged into multiple layers that extract and convert feature variable values from the input vector (Deng et al. 2013; Deng and Yu 2014). The individual layers of such a system have as input, the outputs from preceding layers, except for the input layer which just receives signals or the input vectors from the outside environment. Also, during training of the systems, unsupervised or supervised techniques could be applied. This brings about the possible application of these models in supervised learning tasks like classification, and unsupervised tasks like pattern analysis. Deep learning models are also based on the extraction of higher level features from lower level features to obtain a stratified portrayal of the input data via an unsupervised learning approach on the different levels of the features (Deng and Yu 2014). A ranking of notions and theories is obtained by learning different layers of portrayals of the data that represent varying levels of absorption of the data. Some of the deep learning frameworks in the literature are deep belief networks (DBNs) (Hinton et al. 2006; Hinton 2009), deep/stacked autoencoder networks (DAEs/SAEs) (Vincent et al. 2010; Larochelle et al. 2009) and convolutional neural networks (CNNs) (Alex et al. 2012; LeCun and Bengio 1995). These deep learning frameworks have been used in a variety of sectors such as natural language processing, speech recognition, audio recognition, object

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_2

recognition and detection and computer vision. The deep learning technique used in this book is the stacked autoencoder (SAE) which is built from restricted Boltzmann machines trained in an unsupervised manner using the contrastive divergence method and subsequently joined to form the encoder and decoder parts of the network, which is then trained in a supervised learning manner using the stochastic gradient descent algorithm. The motivation behind using an SAE is that it is trained in such a way that the hidden layer maintains all the information about the input.

Deep learning is a branch of machine learning algorithms that

- uses a deluge of several layers of non-linear processing nodes for the extraction and transformation of features. Successive layers use the outputs from the previous layers as input.
- learn in unsupervised (e.g. pattern analysis) and/or supervised (e.g. classification) manners.
- learn several levels of representations that are related to different levels of abstraction. These levels represent a hierarchy of concepts.

2.2 Restricted Boltzmann Machines (RBMs)

Prior to defining an RBM, we begin by explaining what a Boltzmann machine (BM) is. It is a bidirectionally connected network of stochastic processing units, which can be interpreted as a neural network (Salakhutdinov et al. 2007). It can be used to assimilate key properties of an anonymous probability distribution based on cases from the distribution. This is typically a challenging procedure. The learning procedure can be simplified by imposing constraints on the architecture of the network which leads to restricted Boltzmann machines (Salakhutdinov and Hinton 2009). RBMs can be defined as probabilistic, undirected, parametrized graphical models also referred to as Markov random fields (MRFs). RBMs have received a lot of attention in the aftermath of being proposed as building blocks of multilayered architectures called deep networks (Salakhutdinov and Hinton 2009; Leke and Marwala 2016). The concept behind deep networks is that the hidden neurons excerpt relevant features from the input data, which then serve as input to another RBM (Tieleman 2008). The goal in assembling the RBMs is to obtain higher level portrayals of the data by learning features from features (Tieleman 2008). RBMs which are also MRFs linked to binary undirected graphs are made up of m visible units, $V = (V_1, \ldots, V_m)$ to mean detectable data, and n hidden units, $H = (H_1, \ldots, H_n)$ that record the relationship between variables in the input layer (Hinton et al. 2006; Hinton 2010). The variables V take on values in the range $[0, 1]^{m+n}$, whereas the variables H take on values in the range $\{0, 1\}^{m+n}$. The joint probability distribution which is obtained from the Gibbs distribution requires an activation function given by (Tieleman 2008; LeCun et al. 2015):

$$E(v, h) = -h^T W v - b^T v - c^T h. \tag{2.1}$$

In scalar form, (2.1) is expressed as (Tieleman 2008; LeCun et al. 2015):

$$E(v, h) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i. \tag{2.2}$$

In Eq. (2.2), w_{ij} represents a real-valued weight between the input unit V_j and the hidden unit H_i. This value is the most essential part of an RBM. The parameters b_j and c_i represent real-valued bias terms associated with the jth visible variable and the ith hidden variable (Leke and Marwala 2016). In a scenario where w_{ij} is less than zero and $v_j = h_i = 1$, a high energy is obtained courtesy of a decrease in probability. However, if w_{ij} is greater than zero, and $v_j = h_i = 0$, a lower energy value is obtained because of an increase in probability. If b_j is less than zero and $v_j = 1$, a low probability is achieved due to an increase in energy (Tieleman 2008; LeCun et al. 2015). This points to an inclination for v_j to be equal to zero, rather than v_j being equal to one. On the other hand, if b_j is greater than zero and $v_j = 0$, a high probability is achieved due to a decrease in energy. This points to an inclination for v_j to be equal to one, rather than v_j being equal to zero. The second term in Eq. (2.2) is influenced by the value of b_j with a value less than zero decreasing this term, while a value greater than zero increases the term. The third term of Eq. (2.2), on the other hand, is influenced by the value of c_i in the same way as b_j affects the second term. The Gibbs distribution or probability from (2.1) or (2.2) can be obtained by Tieleman (2008), LeCun et al. (2015):

$$\begin{aligned} p(v, h) &= \frac{e^{-E(v,h)}}{Z}, \\ &= \frac{e^{(h^T W v + b^T v + c^T h)}}{Z}, \\ &= \frac{e^{(h^T W v)} e^{(b^T v)} e^{(c^T h)}}{Z}. \end{aligned} \tag{2.3}$$

In this equation, Z represents an intractable partition function while all the exponential terms represent factors of a Markov network with vector nodes (Leke and Marwala 2016). The intractable nature of Z is due to the exponential number of values it can assume. In RBMs, the intractable partition function is obtained by (Tieleman 2008; LeCun et al. 2015):

$$Z = \sum_{v,h} e^{-E(v,h)}. \tag{2.4}$$

The variable, h, is conditionally independent of v, and vice versa, and this is yet another important trait of an RBM. This is because no nodes in the same layer are connected. Mathematically, this can be expressed as (Tieleman 2008; LeCun et al. 2015):

$$p(h|v) = \prod_{i=1}^{n} p(h_i|v), \tag{2.5}$$

and

$$p(v|h) = \prod_{i=1}^{m} p(v_i|h). \tag{2.6}$$

2.3 Contrastive Divergence (CD)

In training an RBM, the goal is to reduce the mean negative log-likelihood or loss by as much as possible without any form of regularization (Leke and Marwala 2016). This is done by making use of the stochastic gradient descent algorithm because it can handle high-dimensional datasets better than others. The loss is expressed as (Tieleman 2008; Tieleman and Hinton 2009)

$$\text{loss} = \frac{1}{T} \sum_{t} -\log p\left(v^{(t)}\right). \tag{2.7}$$

This can be achieved by calculating the partial derivative of the loss function with respect to a parameter, θ, as follows (Tieleman 2008; Tieleman and Hinton 2009):

$$\frac{\partial\left(-\log p\left(v^{(t)}\right)\right)}{\partial\theta} = E_h\left[\frac{\partial E\left(v^{(t)}, h\right)}{\partial\theta}|v^{(t)}\right] - E_{v,h}\left[\frac{\partial E(v,h)}{\partial\theta}\right]. \tag{2.8}$$

The first term in (2.20) defines the expectation over the distribution of the data. This is coined the positive phase. v and h are the same variables used in Eqs. (2.1)–(2.6). The second term referred to as the negative phase represents the expectation over the distribution of the model. Due to an exponential sum being needed over the v and h variables, the calculation of these partial derivatives is both intractable and difficult (Carreira-Perpin and Hinton 2005). In addition, achieving estimates of the log-likelihood gradient which are unbiased normally needs several steps of sampling. Recently, though it has been revealed that estimates which are obtained from executing the Markov chain for a few iterations can suffice during the model training process. From this emerged the contrastive divergence (CD) method (Tieleman 2008; Tieleman and Hinton 2009). CD can be defined as a technique for training undirected graphical models of a probabilistic nature. The hope is that this eradicates the double expectation procedure in the negative phase of Eq. (2.20) and rather sheds the spotlight on estimation. It essentially uses a Monte-Carlo estimate of the expectation over one input data point (Carreira-Perpin and Hinton 2005). An extension of the CD algorithm is the k-step CD learning technique (CD-k) which states that instead of approximating the second term in Eq. (2.20) by a case from the distribution of the

model, a Gibbs chain could be executed for only k steps with k often set to 1. $v^{(0)}$ is a training sample from the training set of data used to initialize the Gibbs chain, and, this produces the sample $v^{(k)}$ after k steps. Each time step comprises of a sample $h^{(t)}$ obtained from a probability $p(h|v^{(t)})$ as well as a sample $v^{(t+1)}$, subsequently, obtained from $p(v|h^{(t)})$.

The partial derivative of the log-likelihood with respect to θ for a single training sample, $v^{(0)}$, is approximated by Tieleman (2008), Tieleman and Hinton (2009):

$$\mathrm{CD}_k\left(\theta, v^{(0)}\right) = -\sum_h p\left(h|v^{(0)}\right)\frac{\partial E\left(v^{(0)}, h\right)}{\partial \theta} + \sum_h p(h|v^{(k)})\frac{\partial E\left(v^{(k)}, h\right)}{\partial \theta}. \tag{2.9}$$

Due to the fact that $v^{(k)}$ is not drawn from a stationary distribution of the model, the estimates from Eq. (2.9) are biased (Leke and Marwala 2016). As $k \to \infty$, the bias vanishes. An additional factor that points to the biased nature of the CD algorithm is the fact that it maximizes the disparity between two Kullback–Liebler (KL) divergences (Tieleman 2008; Tieleman and Hinton 2009):

$$\mathrm{KL}(q|p) - \mathrm{KL}(p_k|p). \tag{2.10}$$

Here, p_k defines the distribution of the visible variables after k steps of the Markov chain while q represents the empirical distribution. If the chain is observed to have already attained stationarity, then $p_k = p$, therefore, $KL(p_k|p) = 0$, and with this, the error from the CD estimates vanishes. More information on the CD algorithm can be found in Hinton (2002).

2.4 Deep Belief Networks (DBNs)

A DBN is a type of deep neural network that in essence is a probabilistic generative model that comprises several layers of hidden variables (Hinton et al. 2006). These networks have both directed and undirected edges. It is trained using a series of RBMs, often autoencoders, with an additional layer that forms a Bayesian network (Hinton and Salakhutdinov 2006). The use of RBMs means the presence of no intra-layer connections. Furthermore, the performance of a DBN depends heavily on the initialization of the nodes, hence, the layers make use of unsupervised pre-training using the stacking of RBMs procedure, which incorporates CD (Hinton et al. 2006; Le Roux et al. 2008). There are two parts to understanding a DBN being an RBM and a belief network. A belief network (BN) is a directed acyclic graph made up of stochastic binary unit layers where each connected layer has some weights (Hinton et al. 2006). These stochastic binary units have state, 0 or 1, and the probability of being turned on (becoming 1) is determined by a bias and weighted input from other units, represented as (Hinton et al. 2006; Le Roux et al. 2008):

$$p(u_i = 1) = \frac{1}{1 + e^{-\left(b_i + \sum_j u_j w_{ji}\right)}}, \tag{2.11}$$

where u is a stochastic unit, b is the bias associated with that unit, and w is the weight parameter.

Some of the units are visible units and with these there are two main problems which are aimed to be solved. These are (Hinton et al. 2006):

1. The inference problem: whereby the states of the unobserved units are to be inferred.
2. The learning problem: whereby the interactions between units are adjusted to ensure that the network is capable of generating the observed data in the visible units.

A DBN is a multilayered belief network whereby each layer is an RBM stacked against one another to form the DBN (Hinton et al. 2006; Le Roux et al. 2008). The initial step in training a DBN using a basic approach is to learn a layer of features from the visible units by way of the CD algorithm described in Sect. 2.3 (Hinton et al. 2006; Le Roux et al. 2008). The subsequent step involves treating the activations of previously trained features as visible units and then learning features from features in the subsequent hidden layers (Hinton et al. 2006; Le Roux et al. 2008). The final step entails training the entire DBN in a supervised learning manner, which fine-tunes the network parameters. As stated before, RBMs can be stacked together and trained in a greedy way to form these DBNs which are graphical models that can learn to extract deep hierarchical representations of the input training data (Hinton et al. 2006; Le Roux et al. 2008). These frameworks model the joint distribution between observed data vector, x, and the l hidden layers h^k as follows (Hinton et al. 2006; Le Roux et al. 2008):

$$P\left(x, h^1, \ldots, h^l\right) = \left(\prod_{k=0}^{l-2} P(h^k | h^{k+1})\right) P\left(h^{l-1}, h^l\right), \tag{2.12}$$

where $x = h^0$, $P(h^{k-1}|h^k)$ represents the visible unit's conditional distribution conditioned on the hidden units of the RBM at level k. Also, the visible-hidden joint distribution in the top-level of the RBM is $P(h^{l-1}|h^l)$. This can be seen in Fig. 2.1 (Hinton et al. 2006).

The idea behind a greedy layerwise unsupervised learning procedure is applicable to DBNs with RBMs as the building blocks for each layer (Hinton et al. 2006). This procedure is as follows (Bengio et al. 2007; Hinton et al. 2006):

- Train the first layer as an RBM that models the raw input data, $x = h^0$ as its visible layer.
- This layer is then used to obtain representations of the input that will become the input data for the second layer. These representations can be selected as being samples of $p(h^{(1)}|h^{(0)})$ or the mean activations $p(h^{(1)} = 1|h^{(0)})$.

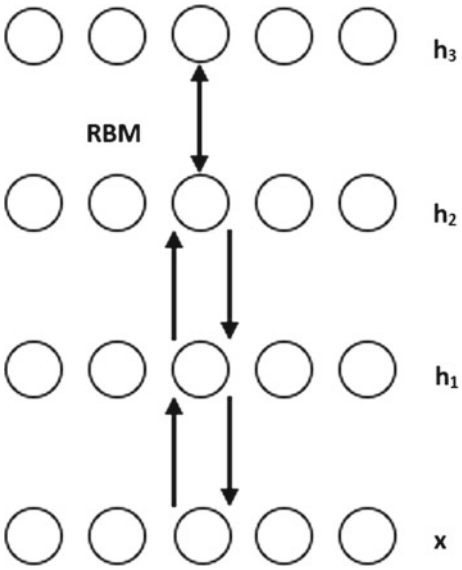

Fig. 2.1 Illustration of a deep belief network

- Train the second layer as an RBM considering the transformed data (mean or samples activations) as training examples (for the visible layer of that RBM).
- These procedures are implemented for the desired number of layers, at each point in time propagating either samples or mean values upward.
- The network created from the steps above is fine-tuned. This includes all the parameters of the DBN with respect to a proxy for the DBN log-likelihood, or with respect to a supervised learning criterion (this is possible after extra learning steps have been added to transform the learned representations into supervised predictions, for example, a linear classifier).

To validate that this approach to training a DBN works, let us consider a two-layer DBN with hidden layers $h^{(1)}$ and $h^{(2)}$. These hidden layers have corresponding weight parameters, $W^{(1)}$ and $W^{(2)}$. It was established in Hinton et al. (2006) that $\log p(x)$ can be rewritten as

$$\log p(x) = KL(Q(h^{(1)}|x)||p(h^{(1)}|x)) + H_{Q(h^{(1)}|x)} + \sum_{h} Q(h^{(1)}|x)\Big(\log p\Big(h^{(1)}\Big) + \log p(x|h^{(1)})\Big). \tag{2.13}$$

$KL(Q(h^{(1)}|x)||p(h^{(1)}|x))$ is the Kullback–Liebler divergence between the posterior distribution, $Q(h^{(1)}|x)$, of the first RBM if it were standalone, and the probability distribution, $p(h^{(1)}|x)$, for the same layer, but defined by the entire DBN. This is possible when the prior distribution, $p(h^{(1)}|h^{(2)})$, defined by the top-level RBM is considered. The second term, $H_{Q(h^{(1)}|x)}$, represents the entropy of the distribution, $Q(h^{(1)}|x)$.

Furthermore, it can be shown that if both hidden layers are initialized such that

$$W^{(2)} = W^{(1)^T}, \tag{2.14}$$

then the following is true

$$Q\left(h^{(1)}|x\right) = p(h^{(1)}|x), \tag{2.15}$$

and the KL divergence term is zero (Hinton 2009; Hinton et al. 2006; Bengio et al. 2007). If the first-level RBM is trained and its parameters, $W^{(1)}$, kept constant, optimizing the equation above with respect to $W^{(2)}$ can therefore only give the likelihood, $p(x)$.

In addition, it can be observed that if the terms that depend only on $W^{(2)}$ are isolated, the following equation is obtained (Hinton et al. 2006):

$$\sum_h Q(h^{(1)}|x)p\left(h^{(1)}\right). \tag{2.16}$$

The result obtained from optimizing the above equation with respect to $W^{(2)}$ results in the training of a second stage RBM which makes use of the output of $Q\left(h^{(1)}|x\right)$ as the training distribution, when x is sampled from the training distribution for the first RBM (Hinton et al. 2006).

2.5 Convolutional Neural Networks (CNNs)

CNNs are neural networks that have proven to be highly efficient in research like image recognition and classification (Alex et al. 2012; Krizhevsky et al. 2012; Simard et al. 2003). These networks have also been successfully applied in facial recognition, object detection and identification and traffic sign recognition (Sermanet and LeCun 2011). CNNs have also been applied effectively in a variety of Natural Language Processing tasks like sentence classification and sentiment analysis (Kim 2014; Hu et al. 2014). CNNs mainly comprise four procedures and these are as follows:

1. Convolution,
2. Pooling or Subsampling,
3. Non-Linearity (ReLU) and
4. Classification.

These procedures are the building blocks of CNNs and to explain these, we will use the classification of images as an example. Images can be characterized by a matrix that contains the values of the image pixels. It is easier to generate such a representation if it is a two-dimensional image such as a grayscale image. The pixel values are in the range [0 255], with 0 representing black and 255 representing white.

2.5.1 Convolution

The first procedure is convolution from which CNNs derive their name. The initial aim of convolution in CNNs is to identify features present in the input matrix representation of the image. Convolution conserves the dimensional links that exist between pixels by identifying the features of the input image using small squares of the input matrix (Simard et al. 2003). Let us consider a 6×6 image with normalized pixel values in the range [0, 1] as shown below.

1	1	1	0	0	0
0	0	1	1	1	0
0	0	0	1	1	1
0	0	1	1	0	0
0	1	1	0	0	1
1	0	1	0	1	0

Furthermore, consider a 3×3 matrix like below:

1	0	1
0	1	0
1	0	1

The convolution of the input image matrix and the green matrix above is calculated by sliding this 3×3 image by 1 pixel through the input image matrix as such:

1*1	1*0	1*1	0	0	0
0*0	0*1	1*0	1	1	0
0*1	0*0	0*1	1	1	1
0	0	1	1	0	0
0	1	1	0	0	1
1	0	1	0	1	0

1	1*1	1*0	0*1	0	0
0	0*0	1*1	1*0	1	0
0	0*1	0*0	1*1	1	1
0	0	1	1	0	0
0	1	1	0	0	1
1	0	1	0	1	0

1	1	1*1	0*0	0*1	0
0	0	1*0	1*1	1*0	0
0	0	0*1	1*0	1*1	1
0	0	1	1	0	0
0	1	1	0	0	1
1	0	1	0	1	0

1	1	1	0*1	0*0	0*1
0	0	1	1*0	1*1	0*0
0	0	0	1*1	1*0	1*1
0	0	1	1	0	0
0	1	1	0	0	1
1	0	1	0	1	0

This sliding operation is called a stride and would lead to the convoluted matrix

2	3	3	3
2	2	4	3
1	3	3	3
4	2	3	1

This is achieved by performing an elementwise multiplication, then adding the outputs from the multiplication to obtain the final integer. This makes up a single

component of the resultant matrix (yellow). Only a portion of the input matrix is seen by the green matrix with each stride (Simard et al. 2003). The green matrix is referred to as a *filter* in CNN terminology. It can also be referred to as a *kernel* or *feature detector.* The output from a convolution operation is referred to as the *Convolved Feature* or *Feature Map* (Simard et al. 2003). These filters are used as feature detectors to extract features from the input image. It can be deduced from these matrices that diverse values will yield distinct feature maps for the same input. For instance, consider the filter below, on the same 6×6 input image given above

0	1	0
1	0	1
0	1	0

This would lead to the feature map

2	2	3	2
0	3	3	3
2	2	2	2
1	3	2	2

With different filters, one can perform computations like blur, sharpen and edge detection. This is achieved by simply altering the values of the filter matrix prior to performing convolution operations. What this implies is disparate filters can extract disparate features from an image such as curves, edges and so on. It is worth mentioning that the convolution operation records the local relationships in the original image. Practically, a convolutional neural network assimilates the filter values by itself during the training phase of operations. However, it is still important that certain criteria be specified like the architecture of the network, the number of filters and the filter size before training can be performed. The more filters there are, the more diverse the features are that get extracted from the image. This will also make the network better at generalizing.

Three parameters control the feature map size that must be decided upon before the convolution step. These are the depth, the stride and zero-padding. The depth defines the number of filters to be used for the convolution procedure which leads to the same number of feature maps. The stride defines the number of pixels by which one slides the filter matrix across the original input matrix when performing the convolution procedure. A value of one mean moves the filter one pixel at a time while a value of two says jump two pixels at a time. Larger stride values result in smaller feature maps. For example, using the same 6×6 input matrix above, if the filter is a 4×4 matrix:

1	0	0	1
0	1	1	0
1	0	1	0
0	1	0	1

Then the corresponding feature map will be:

3	5	5
3	4	5
3	6	3

It can be seen that the dimension of this feature map is smaller than that of when a 3×3 filter matrix is used (3×3 vs. 4×4). Zero-padding entails adding zeros around the border of the input matrix to make applying the filter to the border elements of the input matrix possible. One beneficial aspect of zero-padding is that it leaves room for control of the size of the feature maps. This procedure is also called *wide convolution* with the converse termed a *narrow convolution*.

2.5.2 Non-linearity (ReLU)

The next procedure after convolution is ReLU which stands for Rectified Linear Unit and it is a non-linear procedure. The outputs of this operation can be obtained by (Hu et al. 2014; Krizhevsky et al. 2012):

$$O = \max(0, x).$$

It is an elementwise operation that is applied on individual pixels of an image, replacing the negative pixels by zero in the feature map. The main objective of the ReLU operation is to incorporate non-linearity into the CNN, adding on to convolution which comprises elementwise matrix multiplication and addition, which are linear operations, as most of the data in the real world are non-linear. Other non-linear functions such as hyperbolic tangent and sigmoid can be applied in the place of the ReLU operation, but the ReLU operation has been observed to perform more efficiently (Hu et al. 2014; Krizhevsky et al. 2012). The output feature map of a ReLU operation is called a rectified feature map.

2.5.3 *Pooling*

Spatial pooling, also referred to as subsampling or down-sampling is an operation meant to reduce the dimensionality of the feature maps, however, it possesses the advantage of retaining the vital information. Pooling could be max pooling, average pooling or sum pooling to name a few (Hu et al. 2014; Krizhevsky et al. 2012). Considering max pooling, a spatial neighbourhood is defined, from which the largest element from the output from the ReLU operation in the window is chosen. An alternative to selecting the largest element in the window is to compute the average of the elements (average Pooling). Another option is to calculate the sum of the elements (sum pooling). Max pooling has been proven to perform better (Hu et al. 2014; Krizhevsky et al. 2012).

Below, we depict an example of the max ooling computation on a rectified feature map that is obtained after convolution and the ReLU operation, using a 2×2 window. Consider a 4×4 matrix

0	0	1	2
2	3	3	4
1	1	0	0
5	1	1	2

The Max Pooling operation leads to

3	4
5	2

The above matrix is obtained by

$$\text{Max}(0, 0, 2, 3)$$
$$\text{Max}(1, 2, 3, 4)$$
$$\text{Max}(1, 1, 5, 1)$$
$$\text{Max}(0, 0, 1, 2)$$

This is done by sliding the 2×2 window by two cells and taking the maximum value in the region. This operation leads to a reduction in the feature map dimensions as can be seen from the two matrices above. The pooling operation is applied to each feature map, so if there are three feature maps, the pooling operation will be performed on these three maps, resulting in three feature maps with reduced dimensions. The primary goal of the pooling operation is to reduce the spatial dimension of the input matrix in a progressive manner. More specifically, pooling:

- reduces the dimension of the input representation feature map making it more manageable,
- controls the issue of overfitting by lowering the number of computations and parameters in the network,
- renders the network insensitive to minute distortions, transformations and translations in the input image, and,
- helps attain almost scale-invariant representation of the input image termed *equivariant*. This is an important and powerful aspect as it will be possible to detect objects in an image regardless of the location.

2.5.4 Fully Connected Layer

This layer is a conventional multilayer perceptron that makes use of a softmax output layer activation function. The connotation *fully connected* refers to the fact that every neuron is connected to every other neuron in the subsequent layer. The convolutional layer and pooling layer outputs are high-level feature representations of the input image. The aim of this layer is to use these to perform classifications of input images into discrete classes based upon the training set of data. Besides classification, adding this layer is a cheap way of learning non-linear combinations of features. A lot of the features from the convolutional layer and pooling layer could be used to perform classifications, however, combining the features might lead to better classification results. The sum of probabilities at the output from this layer is always 1. This is guaranteed by making use of the softmax function as the output layer activation function. The softmax function takes as input a vector of real-valued scores. It then converts it to a vector of values in the range [0, 1] that sum up to one.

2.6 Recurrent Neural Networks (RNNs)

A recurrent neural network is a type of neural network in which the links between units form a graph with directed edges along a sequence (Grossberg 2013; Mikolov et al. 2010). The principal objective behind these networks is to use the sequential information. In conventional neural networks, it is assumed that there is no interdependency between the inputs and the outputs. However, this is not advisable for certain exercises, for instance, if the aim is to determine what the next word in a sentence is, it is necessary to know what the previous words were. RNNs are referred to as being recurrent since the same operation is performed by the network for all the elements of a sequence with the output depending on previous calculations. One could also think about RNNs as having a memory aspect that records the details regarding what has occurred so far. Theoretically speaking, RNNs can use the knowledge obtained in promptly long sequences while in practical terms, these networks are restricted to using information from only a few steps back. An RNN is as follows:

In the above diagram, an RNN is depicted that gets unfolded into a full network. Unfolding the network implies writing out the network for the entire sequence. For instance, if the sequence under consideration has four words, unfolding the network would lead to a four-layer network. Each layer would represent a word. Below, we outline the equations that are used for the computations that take place in an RNN:

- i_t represents the time step t input.
- s_t represents time step t hidden state. This represents the network's memory. Using the previously hidden state and the input at the current step, s_t is obtained (Grossberg 2013; Mikolov et al. 2010):

$$s_t = f(Xi_t + Ws_{t-1}). \tag{2.17}$$

- f is usually a non-linear function one like the hyperbolic tangent, ReLU or sigmoid. s_{-1} is typically initialized to zeroes and it is needed to compute the first hidden state.
- o_t represents the output at time step t. For instance, if the aim was to determine what the next word in a sentence is, this output could be obtained by

$$o_t = \text{softmax}(Ys_t). \tag{2.18}$$

A few things worth mentioning are as follows:

- One could consider the state s_t as the memory of the network. s_t stores data regarding the computations performed in every one of the previous steps. o_t is computed using only the information stored at time t.
- Contrary to a conventional deep neural network, that makes use of disparate parameters at each layer, RNNs share similar parameters through all steps.
- Figure 2.2 shows that at each time step, there are outputs. However, depending on the exercise, it may not be necessary to do this. For instance, in determining the sentiment of a sentence, one may only be concerned about the final output. It would not be necessary to consider the sentiment after each word. In addition, it may not be necessary to have inputs at each step. The hidden states of an RNN are the main feature of the network and they capture some details about a sequence.

2.7 Deep Autoencoders (DAE)

Autoencoder networks are defined as networks that try to remember the inputs as outputs in the output layer (Mistry et al. 2008), and this guarantees that the network can predict new input values as the outputs when presented with new inputs. These autoencoders are made up of one input layer and one output layer both having the same number of neurons, resulting in the term auto-associative (Lu and Hsu 2002).

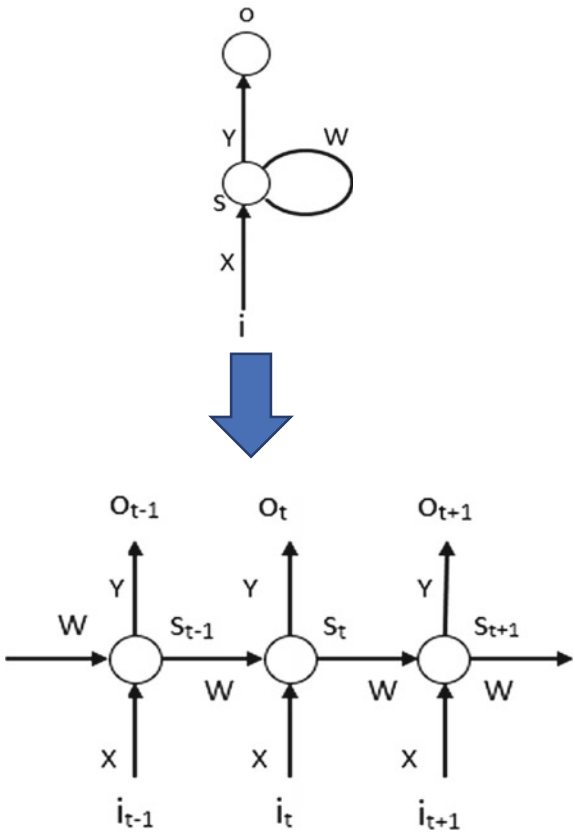

Fig. 2.2 A recurrent neural network

Besides these two layers, a narrow-hidden layer exists, and it is important that this layer contains fewer neurons than there are in the input and output layers with the aim of this being to apply encoding and decoding procedures when trying to find a solution to a given task (Mistry et al. 2008). These networks have been used in a variety of applications (Hines et al. 1998; Atalla and Inman 1998; Smauoi and Al-Yakoob 2003; Marwala 2001). The main concept that defines the operation of autoencoder networks is the notion that the mapping from the input to the output, $x^{(i)} \rightarrow y^{(i)}$ stores very important information and the key inherent architecture present in the input, $x^{(i)}$, that is contrarily hypothetical (Hinton 2002; Hines et al. 1998). An autoencoder takes x as the input and transcribes it into y, which is a hidden representation of the input, by making use of a mapping function, f_θ, that is deterministic. This function is expressed as Leke and Marwala (2016), Isaacs (2014):

$$f_\theta(x) = s(Wx + b). \tag{2.19}$$

The parameter, θ, is made up of the weights W and biases b. s represents the sigmoid activation function which is given by

$$s = \frac{1}{1 + e^{-x}}, \tag{2.20}$$

where y is then mapped to a vector, z, representing the reconstruction of the inputs from this hidden representation. This reconstructed vector is obtained by using the following equations (Leke and Marwala 2016):

$$z = g_{\theta'}(y) = s\left(W'y + b'\right), \tag{2.21}$$

or

$$z = g_{\theta'}(y) = W'y + b'. \tag{2.22}$$

In the above equations, θ' is made up of the transpose of the weights matrix and the vector of biases from Eq. (2.19). Equation (2.21) is the autoencoder output function with a sigmoid transfer function Eq. (2.20), while Eq. (2.22) is the linear output equation. After these operations, the network can then be fine-tuned by applying a supervised learning approach (Isaacs 2014). The network is said to have tied weights when the weights matrix is transposed. The vector z is not described as an exacting transformation of x, but rather as the criteria of a distribution $p(X|Z = z)$ in probabilistic terms. The hope is that these criteria will result in x with high probability (Isaacs 2014; Marwala 2013). The resulting equation is as follows (Leke and Marwala 2016):

$$p(X|Y = y) = p(X|Z = g_{\theta'}(y)). \tag{2.23}$$

This leads to an associated regeneration disparity that forms the basis for the objective function to be used by the optimization algorithm. This disparity is usually represented by (Leke and Maewala 2016)

$$L(x, z) \propto -\log p(x|z), \tag{2.24}$$

where α indicates proportionality. The equation above could be represented by (Sartori et al. 2005):

$$\delta_{AE}(\theta) = \sum_{t} L\left(x^{(t)}, g_\theta\left(f_\theta\left(x^{(t)}\right)\right)\right) \tag{2.25}$$

Autoencoder networks have been used in a variety of application areas, by several researchers, with the focus being on the problem of missing data (Abdella and Marwala 2005; Baek and Cho 2003; Tim et al. 2004; Brain et al. 2006). A Deep autoencoder adopts the same principles of a narrow autoencoder, with the exception

being that there are more hidden layers than the one conventional hidden layer in a normal autoencoder. It still possesses a butterfly-like structure with a linear middle hidden layer. Each layer of the network is an RBM, pretrained using the CD algorithm, and stacked together to form the encoder part of the network. This encoder is then transposed to form the decoder part of the network, and this final network is fine-tuned using a supervised learning method. The creation of each layer of the network is done in an unsupervised learning manner. The fine-tuning is conventionally done using the Stochastic Gradient Descent (SGD) method. The hidden layers, besides the middle one which uses a linear output activation function, use non-linear sigmoid activation functions.

2.8 Conclusions

This chapter begins by presenting some introductory information on deep learning followed by an overview of the building blocks of conventional deep learning frameworks being restricted Boltzmann machines and contrastive divergence. We also discussed some of the existing deep learning frameworks including deep belief networks, convolutional neural networks, recurrent neural networks and deep autoencoders.

References

Abdella, M., & Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. *24*, 577–589.

Alex, K., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25, pp. 1097–1105). Curran Associates, Inc. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. Accessed May 2016.

Atalla, M. J., & Inman, D. J. (1998). On model updating using neural networks. *Mechanical Systems and Signal Processing, 12,* 135–161.

Baek, K., & Cho, S. (2003). Bankruptcy prediction for credit risk using an auto-associative neural network in Korean firms. In *IEEE Conference on Computational Intelligence for Financial Engineering, Hong Kong, China* (pp. 25–29).

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems* (pp. 153–160).

Brain, L. B., Marwala, T., & Tettey, T. (2006). Autoencoder networks for HIV classification. *Current Science, 91*(11), 1467–1473.

Carreira-Perpin, M., & Hinton, G. E. (2005). On contrastive divergence learning. In *Artificial Intelligence and Statistics* (pp. 1–7). http://learning.cs.toronto.edu/_hinton/absps/cdmiguel.pdf. Accessed March 15, 2015.

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing, 7*(3–4), 197–387.

Deng, L., et al. (2013). Recent advances in deep learning for speech research at Microsoft. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 8604–8608).
Grossberg, S. (2013). Recurrent neural networks. *Scholarpedia, 8*(2), 1888.
Hines, J. W., Robert, E. U., & Wrest, D. J. (1998). Use of autoassociative neural networks for signal validation. *Journal of Intelligent and Robotic Systems, 21*(2), 143–154.
Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation, 14*(8), 1771–1800.
Hinton, G. E. (2009). Deep belief networks. *Scholarpedia, 4*(5), 5947.
Hinton, G. (2010). A practical guide to training restricted boltzmann machines. *Momentum, 9*(1), 926.
Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504–507.
Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 527–1554.
Hu, B., Lu, Z., Li, H., & Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems* (pp. 2042–2050).
Isaacs, J. C. (2014). Representational learning for sonar ATR. in SPIE Defense + Security. In *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XIX; 907203*. International Society for Optics and Photonics. https://doi.org/10.1117/12.2053057.
Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv: 1408.5882.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).
Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research, 1,* 1–40.
Le Roux, N., & Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation, 20*(6), 1631–1649.
LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks, 3361*(10), 1–14.
LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). Deep learning. *Nature, 521,* 436–444.
Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference in Swarm Intelligence* (pp. 259–270). Heidelberg: Springer International Publishing.
Lu, P. J., & Hsu, T. C. (2002). Application of autoassociative neural network on gas-path sensor data validation. *Journal of Propulsion and Power, 18*(4), 879–888.
Marwala, T. (2001). Probabilistic fault identification using a committee of neural networks and vibration data. *Journal of Aircraft, 38*(1), 138–146.
Marwala, T. (2013). *Economic modelling using artificial intelligence methods*. London: Springer.
Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of The International Speech Communication Association*.
Mistry, J., Nelwamondo, F., & Marwala, T. (2008). Estimating missing data and determining the confidence of the estimate data. In *Seventh International Conference on Machine Learning and Applications, San Diego, CA, USA* (pp. 752–755).
Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning, Series* (pp. 791–798). New York: ACM. http://doi.acm.org/10.1145/1273496.1273596. Accessed May 2016.
Salakhutdinov, R., & Hinton, G. (2009). Deep boltzmann machines. *Artificial Intelligence and Statistics, 1*(2), 448–455.

Sartori, N., Salvan, A., & Thomaseth, K. (2005). Multiple imputation of missing values in a cancer mortality analysis with estimated exposure dose. *Computational Statistics & Data Analysis, 49*(3), 937–953.

Sermanet, P., & LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *The 2011 International Joint Conference on Neural Networks (IJCNN)* (pp. 2809–2813). IEEE.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition* (p. 958). IEEE.

Smauoi, N., & Al-Yakoob, S. (2003). Analyzing the dynamics of cellular flames using karhunen-loeve decomposition and autoassociative neural networks. *Society for Industrial and Applied Mathematics, 24,* 1790–1808.

Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning, Series* (pp. 1064–1071). New York: ACM. http://doi.acm.org/10.1145/1390156.1390290. Accessed May 2016.

Tieleman, T., & Hinton, G. E. (2009). Using fast weights to improve persistent contrastive divergence. In *Proceedings of 26th International Conference on Machine Learning* (pp. 1033–1040).

Tim, T., Mutajogire, M., & Marwala, T. (2004). Stock market prediction using evolutionary neural networks. In *Fifteenth Annual Symposium of the Pattern Recognition, PRASA* (pp. 123–133).

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research, 11,* 3371–3408.

Chapter 3
Missing Data Estimation Using Bat Algorithm

3.1 Introduction

Missing data has always been a challenge in the real world as well as within the research community. Decision-making processes that rely on accurate knowledge are quite reliant upon the availability of data, from which information can be extracted. Such processes often require predictive models or other computational intelligence techniques that use the observed data as inputs. However, in some cases, data could be lost, corrupted or recorded incompletely, which affects the quality of the data negatively. Majority of the decision-making and machine learning frameworks such as artificial neural networks, support vector machines and principal component analysis cannot be used for decision-making and data analysis if the data is incomplete because these missing values can critically influence pattern recognition and classification tasks. Since the decision output should still be maintained despite the missing data, it is important to deal with the problem. Therefore, in case of incomplete or missing data, the initial step in processing the data is estimating the missing values.

The applications of missing data estimation methods are numerous and include, but are not limited to, structural equation modelling (Carter 2006), air quality data (Zainuri et al. 2015) and reconstruction of times series data (Sidekerskiene and Damasevicius 2016) observations. With that said, the existing methods depend on the nature of the data, the pattern of missingness and are predominantly implemented on low-dimensional datasets. Abdella and Marwala (2005) implemented a joint neural network-genetic algorithm framework to impute missing values in one dimension while Rana et al. (2015) used robust regression imputation in datasets with outliers and investigated its performance. Zhang et al. (2011) recommended that knowledge in relevant instances be used when estimating missing data. This resulted in a nonparametric iterative imputation algorithm (NIIA) being suggested. In papers such as Jerez et al. (2010), Leke and Marwala (2016), Liew et al. (2011), Van Buuren (2012) and Ju et al. (2015), new techniques to impute missing data and comparisons between these and existing methods are observed. The aforementioned

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_3

techniques mainly cater to low-dimensional datasets with missing values but are less effective when high-dimensional datasets are considered with missingness occurring in an uncontrolled manner, as will be depicted subsequently in the chapter. The main motivation behind this work is, therefore, to introduce high-dimensional missing data estimation approaches, with emphasis laid on image recognition datasets. These approaches will be used to reconstruct corrupted images by estimating image pixel values.

In an attempt to investigate new techniques to address the previously mentioned dimensionality drawback, we implement a deep autoencoder, with an autoencoder network being an unsupervised learning technique that endeavours to reconstruct the input data as the output by learning an approximation function which decreases the contrast between the original data, x, and the output reconstructed data, $\tilde{x}$, the same as with principal component analysis (PCA). By definition, a deep autoencoder is made up of two symmetrical deep belief networks with a couple of shallow layers representing the encoding section of the network, while the second set of shallow layers represent the decoding section, with each of the individual layers being restricted Boltzmann machines (RBMs). These networks are applicable in a variety of sectors, for example, deep autoencoders were used for visuomotor learning by Finn et al. (2016) while in Ju et al. (2015), they were used with support vector machines (SVMs) to learn sparse features and perform classification tasks. Feng et al. (2014) used a deep denoising autoencoder to generate robust speech features for a noisy reverberant speech recognition system, and in Krizhevsky and Hinton (2011), these networks were used to map small colour images to short binary codes. Autoencoders have several advantages such as being more flexible by introducing nonlinearity in the encoding part of the network contrary to the likes of PCA, which is a key property of the pretraining procedure, they require no prior knowledge of the data properties and correlations, and also, they are intuitive. The main drawback of this network is its need for a significant amount of samples for training and learning, which are not always readily available.

On the other hand, the bat algorithm, which is a metaheuristic swarm intelligence technique based on the echolocation trait of bats is also used. Bats possess incredible capabilities with their ability to hunt for prey in the dark using sonar and the *Doppler effect*. This trait was expressed mathematically in Yang (2010) in the form of an optimization algorithm which was tested against existing optimization methods on a set of benchmark functions. In Yang (2011), the bat algorithm was implemented to solve multi-objective optimization tasks which comprise of most engineering problems while in Yang et al. (2012) it was applied as an approach to solve topology optimization problems. Teodoro et al. (2012) used the bat algorithm to optimize mono/multi-objective tasks linked to brushless DC wheel motors. Some of the inherent advantages of the algorithm are that it implements a great balance between exploitation (by using the loudness and pulse emission rate of bats) and exploration (like in the standard Particle Swarm Optimization (PSO)), and also, Yang (2010) reveals that there is a guarantee to attain a global optimum solution in the search of an optimal point. These are some of the reasons why the algorithm was used in this research, in addition to the fact that it has not been used before in

the missing data literature. Furthermore, literature on the Bat algorithm reveals that it is more than capable in terms of performance in solving a variety of problems in different application domains, therefore, we thought it necessary to investigate how it will perform in the domain of missing data estimation. The main disadvantage of the algorithm, however, is that the convergence rate of the optimization process depends on the fine adjustment of the algorithm parameters.

In this chapter, we propose a new method for missing data estimation in high-dimensional datasets, namely, DL-BAT, based on the advantages of deep autoencoders and the bat algorithm. For this purpose, we begin by describing the proposed model for the imputation task which consists of a deep autoencoder and the bat algorithm. Subsequently, we present analyses of the performance of the proposed model and compare the results obtained to those of existing methods. Furthermore, we report on the results and discuss the findings from our experiments. Finally, brief conclusions are presented.

3.2 Model

In this section, we describe the proposed approach that combines the advantages of deep autoencoder networks and the bat algorithm. When applying a network of any nature to estimate missing data, it is very important that the network model be very accurate. The proposed methodology is summarized in Fig. 3.1, where I_{k} and I_{u} represent the known and unknown features, respectively, and make up the set of inputs.

First, the network is trained using the stochastic gradient descent (SGD) algorithm, with a complete set of records to identify a mapping function, f, that extracts and stores the interrelationships and correlations between features in the dataset. This leads to the output, $\vec{O}$, obtained by

$$\vec{O} = f\left(\begin{bmatrix} \overrightarrow{I_{\mathrm{k}}} \\ \overrightarrow{I_{\mathrm{u}}} \end{bmatrix}\right). \tag{3.1}$$

Due to the fact that the autoencoder framework tries to represent the input as the output of the network, the output is approximately equal to the input. This results in a small error which relies upon the accuracy of the autoencoder. The error is expressed as

$$\vec{e} = \begin{bmatrix} \overrightarrow{I_{\mathrm{k}}} \\ \overrightarrow{I_{\mathrm{u}}} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_{\mathrm{k}}} \\ \overrightarrow{I_{\mathrm{u}}} \end{bmatrix}, \vec{w}\right). \tag{3.2}$$

To guarantee that the error is always positive, Eq. (3.2) is squared, yielding the following equation:

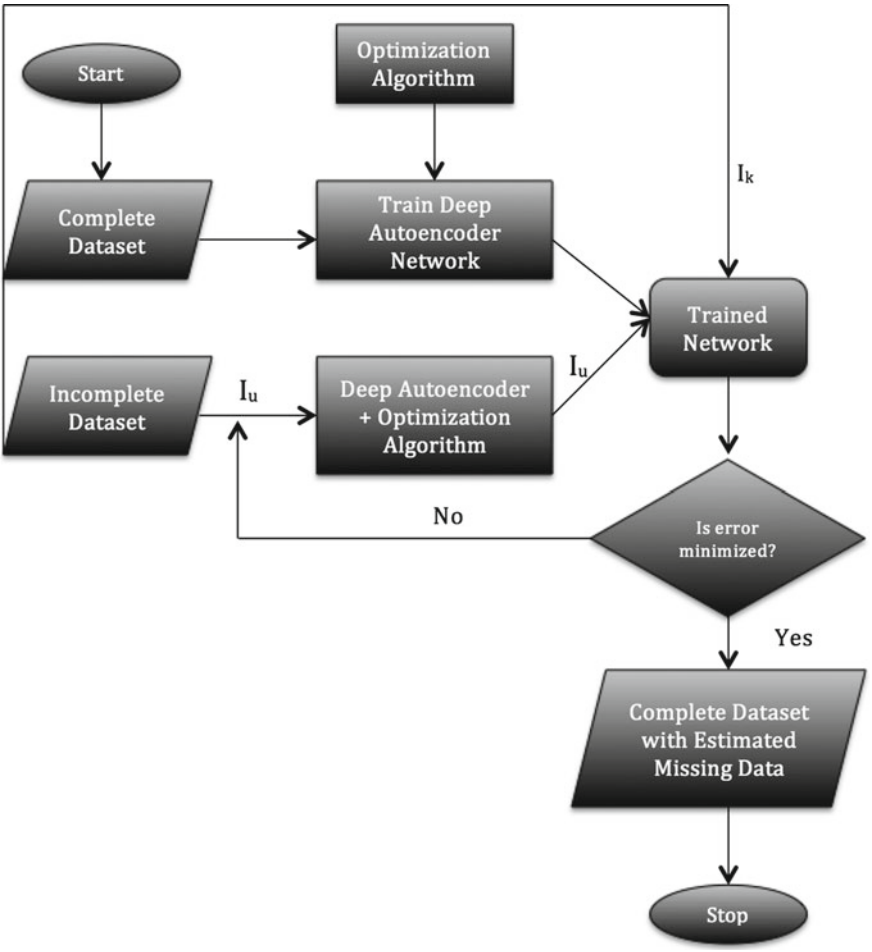

Fig. 3.1 Flow chart representation of estimation model. © 2017 IEEE. Reproduced with permission from Leke et al. (2017)

$$\vec{e} = \left(\begin{bmatrix} \overrightarrow{I_{\mathrm{k}}} \\ \overrightarrow{I_{\mathrm{u}}} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_{\mathrm{k}}} \\ \overrightarrow{I_{\mathrm{u}}} \end{bmatrix}, \vec{w} \right) \right)^2. \tag{3.3}$$

The error will be minimized when the output is as close as possible to the input. This predominantly occurs when the data presented to the framework has similar characteristics as those recorded during training. Therefore, during the estimation procedure, minimizing the error symbolizes the optimization of the missing data in such a way that the new data matrix matches the pattern yielded by the complete dataset during training. The bat algorithm is used to further optimize the network

weights, based on the fact that as mentioned before, it is guaranteed to attain a global optimum point, thereby making sure the network is as accurate as possible. The algorithm as an optimization technique tries to attain a global optimum point which in the landscape of the error function, being the mean squared error, is the lowest point or valley of the function. This procedure is expressed mathematically as

$$\underset{\vec{w}}{\text{minimize}} \left(\begin{bmatrix} \overrightarrow{I_{\text{k}}} \\ \overrightarrow{I_{\text{u}}} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_{\text{k}}} \\ \overrightarrow{I_{\text{u}}} \end{bmatrix}, \vec{w} \right) \right)^2. \tag{3.4}$$

To achieve this objective, each bat moves around the solution space with a specific velocity and at a given point/position. There always exists a bat towards which all other bats move and this constitutes the current best solution. In addition to these, the bats in using sonar emit sounds with a particular frequency, wavelength and loudness. These can be adjusted depending on the proximity of a prey or an obstacle. These properties are expressed mathematically in Eqs. (3.5), (3.6) and (3.7)

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{3.5}$$

$$v_i^{t+1} = v_i^t + \left(x_i^t - x^*\right) f_i, \tag{3.6}$$

and

$$f_i = f_{\min} + (f_{\max} - f_{\min})v, \tag{3.7}$$

where $v \in [0, 1]$ is a random vector drawn from U(a, b) with $-\infty < a < b < \infty$ and probability density function, $f(x) = \frac{1}{b-a}$, such that $a < x < b$. The bat algorithm implemented in this chapter adds an element of randomness in the motion of the bats in the form of a Lévy flight. This results in the position equation being

$$x_i^{t+1} = x_i^t + v_i^{t+1} + \text{Lévy}(\lambda). \tag{3.8}$$

In Fig. 3.2, we depict the pseudocode of the original bat algorithm which reveals the stopping criteria options and points, the stages at which solutions are updated and the parameter update steps.

There are three main rules based upon which the bat algorithm is designed, these being:

1. Each bat uses echolocation to sense distance and also tell the difference between food/prey and obstacles,
2. Bats fly randomly with a velocity and position and a given frequency, varying wavelength and also varying loudness to search for their prey, and,
3. Although there are several ways in which the loudness could vary, the assumption made is that the loudness varies from a large positive value to a low constant value.

The bat algorithm parameters used in the model implementation are given in Table 3.1 with the exception of the number of decision variables which depends on

Bat Algorithm

Objective function $f(x)$, $x = (x_1, ..., x_d)^T$

Initialize the bat population x_i $(i = 1, 2, ..., n)$ *and* v_i

Define pulse frequency f_i *at* x_i

Initialize pulse rates r and the loudness A

while *(t < Max number of iterations)*

Generate new solutions by adjusting frequency,

and updating velocities and locations/solutions [equations (5) to (7)]

if *(rand > r)*

Select a solution among the best solutions

Generate a local solution around the selected best solution

end if

Generate a new solution by flying randomly

If *(rand < A &* $f(x_i) < f(x_*)$*)*

Accept the new solutions

Increase r and reduce A

end if

Rank the bats and find the current best x_*

end while

Postprocess results and visualization

Fig. 3.2 Pseudocode of original bat algorithm

the number of missing values in a record. These parameters were chosen because they resulted in the more optimal outcomes with different combinations and permutations of values having been tested.

The incomplete dataset is passed as input to the bat algorithm which has, as part of the objective function, the optimized deep neural network. The missing data is estimated and together with the known values, run through the trained network to observe whether or not the network error has been minimized, without any changes for a few iterations. If it has, the dataset is returned with imputed values, otherwise, the imputation procedure is executed again. The stopping criteria for the missing data estimation procedure using the bat algorithm are either 40,000 function evaluations having been executed, or there is no change in the objective function value.

Table 3.1 Bat algorithm parameters

Parameter	Value
Population size	40
Loudness	0.25
Pulse rate	0.5
Number of generations	1000

From the above description of how the deep autoencoder and the bat algorithm are used, the equation below summarizes the hybrid function of the proposed approach, with f_{BAT} being the Bat algorithm data estimation operation and f_{DAE} being the function of the deep autoencoder.

$$y = f_{\text{DAE}}\left(W, f_{\text{BAT}}\left(\vec{I}\right)\right), \tag{3.9}$$

where $\vec{I} = \begin{bmatrix} \overrightarrow{I_{\text{k}}} \\ \overrightarrow{I_{\text{u}}} \end{bmatrix}$ represent the input space of known and unknown features.

3.3 Performance Analysis

To investigate and validate the proposed model against other existing approaches, the Mixed National Institute of Standards and Technology (MNIST) dataset of handwritten digits was used. It contains 60,000 training samples and 10,000 test samples, each with 784 features representing the individual pixel values of the image. The black pixels are represented by 0, while the white pixels have values of 255, as such, all the variables could be considered as being continuous. The data in its current form has been cleaned of outliers, therefore, each sample used in the dataset should contribute to generalizing the system. The data are normalized to being in the range [0, 1], and randomized to improve the network performance. The training set of data is further split into training and validation sets (50000 and 10000, respectively).

The neural network architecture is optimized using the validation set of data with the correctness in performance of the proposed system being tested using the test data. It is worth mentioning that the objective of this research is not to propose a state-of-the-art classification model which rivals existing state-of-the art methods, but rather to propose an approach to reconstruct an image in the event of missing data (missing pixels). The optimized deep network architecture used is 784-1000-500-250-30-250-500-1000-784 as suggested initially by Hinton et al. (2006) and further verified by performing cross-validations using the validation set of data, with 784 input and output nodes, and seven hidden layers with 1000, 500, 250, 30, 250,

500 and 1000 nodes, respectively. The objective function value (mean squared error) obtained after training was 1.96% which was close to the lowest value obtained during cross-validation (2.12%).

The multilayer perceptron (MLP) autoencoder used for comparison purposes has the same number of nodes in the input and output layer as the deep autoencoder, with a single hidden layer consisting of 400 nodes which is determined by varying the number of nodes in the hidden layer, and determining which architecture produces the lowest network error. It was experimentally observed that the 784-400-784 MLP autoencoder network architecture yielded a network error value of 2.5088%. The sigmoid and linear activation functions are used for the hidden and output layers of the MLP network, respectively. The sigmoid activation function is used in each layer of the deep autoencoder except for the bottleneck layer where the linear activation function is used. The training is done using the scaled conjugate gradient (SCG) algorithm for the MLP and the stochastic gradient descent (SGD) algorithm for the deep autoencoder network, for 500 epochs.

100 samples are selected at random from the test set of data and features are then removed respecting the missing at random (MAR) and missing completely at random (MCAR) mechanisms as well as the arbitrary missing data pattern. This is done by creating a binomial matrix of the same size as the test data with zeros and ones adhering to the stated mechanisms and pattern, and replacing every occurrence of one with NaN (implying missingness). The dataset obtained from this, with missing values is used to test the performance of the missing data estimation scheme, which is then compared against existing methods.

The effectiveness of the proposed approach is estimated using the mean squared error (MSE), the root mean squared logarithmic error (RMSLE), the correlation coefficient (r) and the relative prediction accuracy (RPA). Also used are the signal-to-noise ratio (SNR) and global deviation (GD). The mean squared and root mean squared logarithmic errors as well as the global deviation yield measures of the difference between the actual and predicted values, and provides an indication of the capability of the estimation.

$$\text{MSE} = \frac{\sum_{i=1}^{n}\left(I_i - \hat{I}_i\right)^2}{n}, \tag{3.10}$$

$$\text{RMSLE} = \sqrt{\frac{\sum_{i=1}^{n}\left(\log(I_i + 1) - \log\left(\hat{I}_i + 1\right)\right)^2}{n}}, \tag{3.11}$$

and

$$\text{GD} = \left(\frac{\sum_{i=1}^{n}\left(\hat{I}_i - I_i\right)}{n}\right)^2. \tag{3.12}$$

The correlation coefficient provides a measure of the similarity between the predicted and actual data. The output value of this measure lies in the range $[-1, 1]$ where the absolute value indicates the strength of the link, while the sign indicates the direction of said link. Therefore, a value close to 1 signifies a strong predictive capability while a value close to -1 signifies otherwise. In the equation below, $\bar{I}$ represents the mean of the data.

$$r = \frac{\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)\left(\hat{I}_i - \bar{\hat{I}}_i\right)}{\left[\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)^2 \sum_{i=1}^{n}\left(\hat{I}_i - \bar{\hat{I}}_i\right)^2\right]^{1/2}}. \tag{3.13}$$

The relative prediction accuracy (RPA), on the other hand, measures the number of estimates made within a specific tolerance, with the tolerance dependent on the sensitivity required by the application. The tolerance is set to 10% as it seems favorable for the application domain. This measure is given by

$$\text{RPA} = \frac{n_\tau}{n} * 100. \tag{3.14}$$

The squared error (SE) measures a quadratic scoring rule that records the average magnitude of the error. This can be obtained by calculating the variance square root, also referred to as the standard deviation. It reduces the variance in the error, contrary to the MSE, hence, its application in this work. SE can be obtained by using the formula

$$\text{SE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(I_i - \hat{I}_i\right)^2}. \tag{3.15}$$

The mean absolute error (MAE) measures the average magnitude of the errors in a dataset without considering direction. Under ideal scenarios, SE values are always greater than the MAE values, and in case of equality, the error values are said to have the same magnitude. This error can be calculated using the following equation:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|I_i - \hat{I}_i\right|. \tag{3.16}$$

The coefficient of determination is a metric regularly applied in statistical analysis tasks aimed at assessing the performance of a model in the explanation and prediction of future outputs. It is also referred to as the R-squared statistic, obtained by the following:

Table 3.2 DL-BAT additional metrics

Method	DL-BAT	MLP-PSO	MLP-SA	MLP-GA
COD	**91.89**	59.19	0.04	0.09
GD	**0.06**	0.8	12.1	12.36
MAE	**4.73**	14.3	47.74	48.15
MSE	**0.89**	5.11	30.92	31.4
SNR	**8.44**	53.47	228.91	230.52

The bold values indicate that the algorithm does yield the best (lowest) results

$$\mathrm{COD} = \left(\frac{\sum_{i=1}^{n} \left(I_i - \bar{I}_i\right)\left(\hat{I}_i - \bar{\hat{I}}_i\right)}{\left[\sum_{i=1}^{n} \left(I_i - \bar{I}_i\right)^2 \sum_{i=1}^{n} \left(\hat{I}_i - \bar{\hat{I}}_i\right)^2\right]^{1/2}} \right)^2. \tag{3.17}$$

The signal-to-noise ratio used in this chapter is obtained by

$$\mathrm{SNR} = \frac{\mathrm{var}\left(I - \hat{I}\right)}{\mathrm{var}\left(\hat{I}\right)}, \tag{3.18}$$

where in Eqs. (3.10)–(3.13) and Eqs. (3.15)–(3.17), n is the number of samples while I and $\hat{I}$ represent the real test set values and predicted missing output values, respectively. In Eq. (3.14), $\mathrm{n_r}$ represents the number of correctly predicted outputs.

3.4 Results and Discussion

Taking into consideration the above-mentioned metrics, the performance of the DL-BAT method was evaluated and compared against existing methods by estimating the missing attributes concurrently, wherever missing data may be ascertained. The scenarios investigated were such that any sample/record could have at least 62, and at most 97 missing attributes (dimensions) to be approximated.

Figures 3.3, 3.4, 3.5 and 3.6 show the performance and comparison of DL-BAT with MLP-PSO, MLP-SA and MLP-GA as well as Tables 3.2 and 3.3. The approach introduced as will be observed, differs significantly from existing methods which cater to cases whereby one or no more than 10 missing values are estimated per instance.

The results reveal that the DL-BAT approach outperforms the other approaches. The squared error is given in Fig. 3.3. It shows a 9.45% of error for DL-BAT while MLP-PSO, MLP-SA and MLP-GA obtain 22.61%, 55.61% and 56.04% error values, respectively.

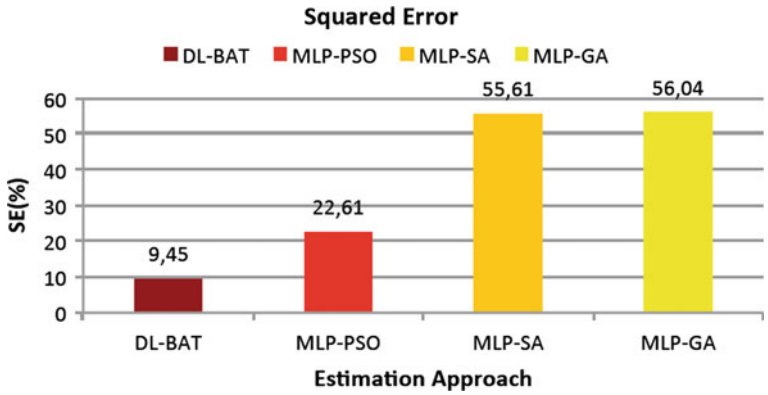

Fig. 3.3 Squared error versus estimation approach

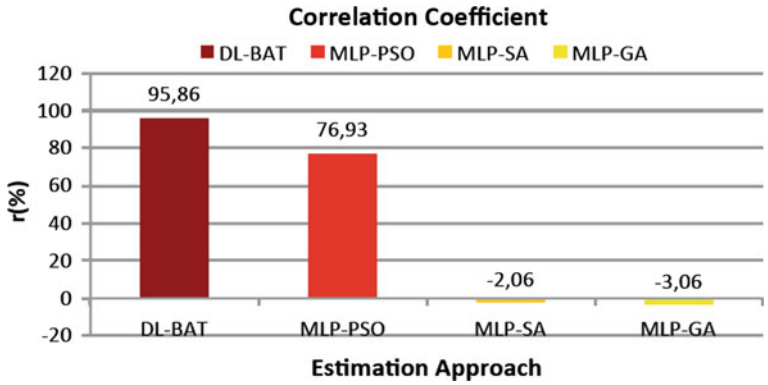

Fig. 3.4 Correlation coefficient versus estimation approach

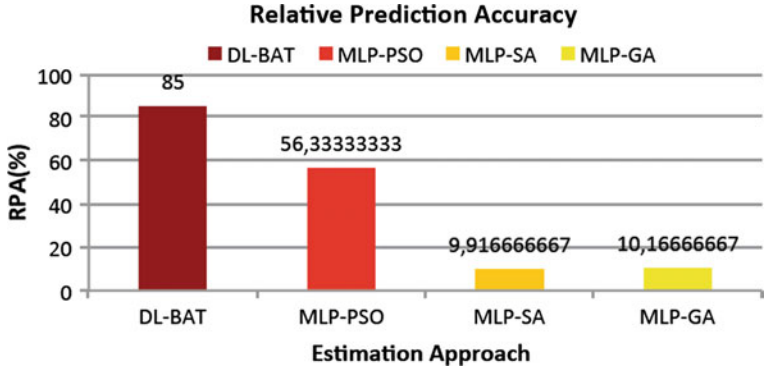

Fig. 3.5 Relative prediction accuracy versus estimation approach

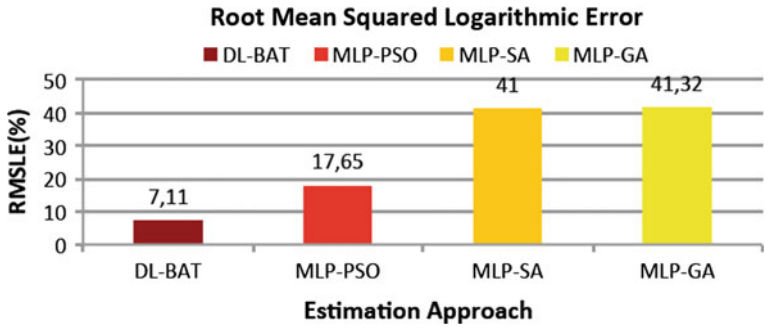

Fig. 3.6 Root mean squared logarithmic error versus estimation approach

Table 3.3 DL-BAT mean squared error objective value per sample

Sample	Dimensions	DL-BAT	MLP-PSO	MLP-SA	MLP-GA
1	66	**5.09**	9.05	9.02	9.02
2	69	**1.62**	5.26	8.60	8.60
3	73	**0.32**	3.69	4.74	4.74
4	85	**3.90**	9.22	18.54	18.54
5	83	**2.20**	7.28	15.83	15.83
6	92	**2.74**	8.59	8.79	8.79
7	77	**3.08**	7.56	13.44	13.44
8	82	**0.53**	6.07	4.46	4.46
9	84	**2.29**	6.51	17.09	17.09
10	63	**0.52**	3.21	1.82	1.82

The bold values indicate that the algorithm does yield the best (lowest) results

Figures 3.4 and 3.5 show the correlation coefficient and relative prediction accuracy of the four approaches analyzed, including the novel DL-BAT approach. They both confirm better performance of the DL-BAT approach when compared to the others. DL-BAT exhibits a 95.86% of correlation with 85% of RPA while we obtain 76.93% of correlation coefficient for MLP-PSO, and −2.06% and −3.06% of correlation for MLP-SA and MLP-GA, respectively. MLP-PSO depicts 56.33% of RPA, while MLP-SA and MLP-GA produce values of 9.92% and 10.17%, respectively.

In Fig. 3.6, we depict the root mean squared logarithmic error values obtained from analyzing the methods. It can be observed that the DL-BAT approach yields the lowest RMSLE value of 7.11% while the second best performer is the MLP-PSO which shows a value of 17.65%. The MLP-SA and MLP-GA approaches produce RMSLE values of 41% and 41.32%, respectively.

These findings are further backed by the values in Table 3.2 with the DL-BAT system yielding the lowest COD, GD, MAE, MSE and SNR values.

Table 3.4 Statistical analysis of DL-BAT results

Pairs compared	*P*-values (95% confidence level)
DL-BAT:MLP-PSO	$1.2 \times 10e{-}07$
DL-BAT:MLP-SA	$2.0 \times 10e{-}134$
DL-BAT:MLP-GA	$9.0 \times 10e{-}137$

Fig. 3.7 Top row: corrupted images—bottom row: DL-BAT reconstructed images

Taking into account Table 3.3, it is observed that the proposed DL-BAT approach results in the best objective function value per record during the estimation of all missing values within that record.

In Table 3.4, we present results obtained from statistically analyzing the estimates obtained by the DL-BAT approach when compared against the MLP-PSO, MLPSA and MLP-GA approach using the t-test. The t-test null hypothesis (H_0) assumes that there is no significant difference in the means of the missing data estimates obtained by the DL-BAT, MLP-PSO, MLP-SA and MLP-GA methods. The alternative hypothesis (H_A), however, indicates that there is a significant difference in the means of the missing data estimates obtained by the four methods.

Table 3.4 reveals that there is a significant difference at a 95% confidence level in the means of the estimates obtained by DL-BAT when compared to MLP-PSO, MLP-SA and MLP-GA, yielding p-values of 1.2*10e-07, 2.0*10e-134 and 9.0*10e-137, respectively, when all three pairs are compared. This, therefore, indicates that the null hypothesis (H_0), which assumes that there is no significant difference in the means between DL-BAT and the other three methods can be rejected in favour of the alternative hypothesis (H_A) at a 95% confidence level.

In the top row of Fig. 3.7, we depict 10 images with missing pixel values which are to be estimated prior to classification tasks being performed by statistical methods. In the bottom row of the same figure, we show the reconstructed images from using the DL-BAT approach, while in the top and bottom rows of Fig. 3.8, we observe the reconstructed images when the MLP-PSO and MLP-GA approaches are used, respectively. It can be seen that the reconstructed images using MLPPSO and MLP-GA introduce a lot of noise, more so in the bottom row than in the top row, as opposed to when the DL-BAT approach is applied. Furthermore, closer inspection reveals that the images are not fully reconstructed as not all pixel values within the images are estimated correctly.

With regards to the duration of the execution, DL-BAT, which exhibits better percentage of error, highest correlation coefficient and highest relative prediction accuracy as well as better performance in the other metrics, it is slower than MLP-PSO, MLP-SA and MLP-GA which are observed to be faster than any other approach as seen in Fig. 3.9.

Fig. 3.8 Top row: MLP-PSO reconstructed images—bottom row: MLP-GA reconstructed images

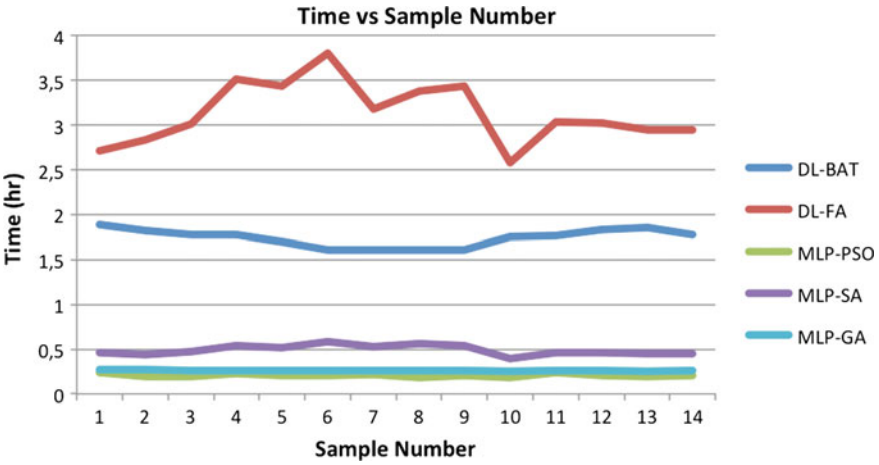

Fig. 3.9 Time versus sample number. © 2017 IEEE. Reproduced with permission from Leke et al. (2017)

3.5 Conclusion

This chapter investigated the estimation of missing data via a novel approach. The estimation method comprises of a deep autoencoder network to replicate the input data, in combination with the bat algorithm to estimate the missing data. The performance of the hybrid model is investigated and compared against existing methods including an MLP autoencoder with genetic algorithm, simulated annealing and particle swarm optimization. The proposed method is also compared to a recent technique suggested to estimate missing data in high-dimensional datasets which makes use of a deep autoencoder network in combination with the firefly algorithm. The results obtained reveal that the proposed system yields more accurate estimates not only when compared against the MLP hybrid systems, but also when compared to the recent deep autoencoder-firefly algorithm system. This is made evident when the standard error, correlation coefficient, relative prediction accuracies, signal-to-noise ratio and global deviation are taken into account, with the proposed approach yielding the best values of these. Also, when the objective function value is considered during the estimation process, it is observed that the proposed approach results in the lowest values for this for each record. An obstacle faced was the computational time required to estimate the missing data which can be addressed by parallelizing the estimation

procedure to observe whether this approach does speed up the process while maintaining efficiency and accuracy. Also, the experiments conducted in this article due to time constraints could only consider a scenario whereby 10% of the data is missing. It would be worth it to experiment further with different percentages of missingness. In addition, it will be worth it to test the proposed scheme on other missing data mechanisms such as missing completely at random (MCAR) and missing by natural design(MBND), as well as other patterns such as the Monotone missing data pattern.

References

Abdella, M., & Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. In *3rd International Conference on Computational Cybernetics, ICCC* (pp. 207–212). IEEE.

Carter, R. L. (2006). Solutions for missing data in structural equation modeling. *Research & Practice in Assessment, 1*(1), 1–6.

Feng, X., Zhang, Y., & Glass, J. R. (2014). Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *International Conference on Acoustic, Speech and Signal Processing (ICASSP)* (pp. 1759–1763). IEEE.

Finn C., Tan, X., Duan, Y., Darrell, T., Levine, S., & Abbeel, P. (2016). Deep spatial autoencoders for visuomotor learning. In *International Conference on Robotics and Automation (ICRA)* (pp. 512–519). IEEE.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 15271554.

Jerez, J. M., et al. (2010). Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine*, *50*(2), 105–115 (Elsevier).

Ju, Y., Guo, J., & Liu, S. (2015). A deep learning method combined sparse autoencoder with SVM. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 257–260). IEEE.

Krizhevsky, A., & Hinton, G. E. (2011). Using very deep autoencoders for content based image retrieval. In *19th European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 27–29 April 2011.

Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference in Swarm Intelligence* (pp. 259–270). Springer International Publishing.

Leke, C., Ndjiongue, A. R., Twala, B., & Marwala, T. (2017). Deep learning-bat high-dimensional missing data estimator. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 483–488) IEEE

Liew, A. W.-C., Law, N.-F., & Yan, H. (2011). Missing value imputation for gene expression data: Computational techniques to recover missing data from available information. *Briefings in Bioinformatics, 12*(5), 498–513 (Oxford University Press).

Rana, S., John, A. H., Midi, H., & Imon, A. (2015). Robust regression imputation for missing data in the presence of outliers. *Far East Journal of Mathematical Sciences*, *97*(2), 183 (Pushpa Publishing House).

Sidekerskiene, T., & Damasevicius, R. (2016). *Reconstruction of missing data in synthetic time series using EMD*.

Teodoro C. B., Leandro d-. S. C., & Luiz, L. (2012). Bat-inspired optimization approach for the brushless DC wheel motor problem. *Transactions on Magnetics*, *48*(2), 947–950 (IEEE).

Van Buuren, S. (2012). *Flexible imputation of missing data*. Boca Raton: CRC Press.

Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. *Nature inspired cooperative strategies for optimization (NISCO)* (pp. 65–74), Studies in Computational Intelligence. Heidelberg: Springer.

Yang, X. S. (2011) Bat algorithm for multi-objective optimization. *International Journal of Bio-Inspired Computation*, *3*(5), 267–274.

Yang, X. S., Karamanoglu, M., & Fong, S. (2012). Bat algorithm for topology optimization in microelectronic applications. In *First International Conference on Future Generation Communication Technologies (FGST)* (pp. 150–155), London, United Kingdom, 12–14 December 2012.

Zainuri, N. A., Jemain, A. A., & Muda, N. (2015). A comparison of various imputation methods for missing values in air quality data. *Sains Malaysiana, 44*(3), 449–456.

Zhang, S., Jin, Z., & Zhu, X. (2011). Missing data imputation by utilizing information within incomplete instances. *Journal of Systems and Software, 84*(3), 452459 (Elsevier).

Chapter 4
Missing Data Estimation Using Cuckoo Search Algorithm

4.1 Introduction

Datasets nowadays such as those that record production, manufacturing and medical data may suffer from the problem of missing data at different phases of the data collection and storage processes. Faults in measuring instruments or data transmission lines are predominant causes of missing data. The occurrence of missing data results in difficulties in decision-making and analysis tasks, which rely on access to complete and accurate data, resulting in data estimation techniques which are not only accurate but also efficient. Several methods exist as a means to alleviate the problems presented by missing data ranging from deleting records with missing attributes (listwise and pairwise data deletion) to approaches that employ statistical and artificial intelligence methods such as those presented in the next paragraph. The problem though is, some of the statistical and naive approaches more often than not produce biased approximations, or they make false assumptions about the data and correlations within the data. These have an adverse effect on the decision-making processes, which are data dependent.

The applications of missing data estimation techniques are vast and with that said, the existing methods depend on the nature of the data, the pattern of missingness and are predominantly implemented on low-dimensional datasets. Abdella and Marwala (2005) implemented a joint neural network–genetic algorithm framework to impute missing values in one dimension, while Leke et al. (2014) used a combination of particle swarm optimization, simulated annealing and genetic algorithm with a multilayer perceptron (MLP) autoencoder network yielding good results in one dimension. Vukosi et al. (2007) used MLP autoencoder networks, principal component analysis and support vector machines in combination with the genetic algorithm to impute missing variables. In Ssali and Marwala (2007), a decision tree is combined with an MLP autoencoder network to estimate missing data in datasets yielding increased accuracy. In papers such as Jerez et al. (2010), Liew et al. (2011), Myers (2011), and Schafer (2002), new techniques to impute missing data and comparisons between

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_4

these and existing methods are observed. Recently, in Leke and Marwala (2016), a technique involving a deep network framework and a swarm intelligence algorithm was proposed to handle missing data in high-dimensional datasets, a first of its nature, with promising outcomes. It is because of this work that we conduct the research in this chapter to investigate other high-dimensional missing data estimation approaches to observe whether the accuracy obtained can be improved upon.

Investigating new techniques to address the previously mentioned drawbacks led to the implementation of a deep autoencoder, which is by definition an unsupervised learning technique that tries to recall the input space by learning an approximation function which diminishes the disparity between the original data, x, and the reconstructed data, $\hat{x}$, the same as with principal component analysis (PCA). A deep autoencoder comprises of two symmetrical deep belief networks with a couple of shallow layers representing the encoding section of the network, while the second set of shallow layers represent the decoding section, with each of the individual layers being a restricted Boltzmann machine (RBM). The RBMs are stacked together to form the overall deep autoencoder network, which is subsequently fine-tuned using the back-propagation algorithm. These networks are applicable in a variety of sectors, for example deep autoencoders were used for visuomotor learning in Finn et al. (2016) while in Ju et al. (2015), they were used with support vector machines (SVMs) to learn sparse features and perform classification tasks. Brain et al. (2006) used an autoencoder for HIV classification, and in Krizhevsky and Hinton (2011), these networks were used to map small colour images to short binary codes. Autoencoders have several advantages such as being more flexible by introducing nonlinearity in the encoding part of the network contrary to the likes of PCA, which is a key property of the pretraining procedure, they require no prior knowledge of the data properties and correlations, and also, they are intuitive. The main drawback of this network is its need for a significant number of samples for training and learning, which are not always readily available.

In combination with a deep autoencoder network, the Cuckoo Search (CS) algorithm is used. It is a population-based stochastic optimization technique inspired by the brood parasitism of cuckoo birds (Yang and Debb 2014). The algorithm is further enhanced by the use of Lévy flights to introduce more randomness in the motion of the birds. It has been utilized to minimize the generation and emission costs of a microgrid while satisfying system hourly demands and constraints (Vasanthakumar et al. 2015), while in Wang et al. (2016), it was used to establish the parameters of chaotic systems via an improved cuckoo search algorithm. Ali and Mohamed (2016) presented a new hybrid algorithm comprised of the cuckoo search algorithm and the Nelder–Mead method with the aim to solve the integer and minimax optimization problems.

In this chapter, a deep autoencoder network is trained and in combination with the CS algorithm (DL-CS), its performance is compared to that of an ordinary Multilayer Perceptron autoencoder and other schemes. The optimization algorithm used is presented. A description of the methodology is presented ensued by a presentation of the model performance and analyses. Subsequently, results and the findings from the experiments are presented, followed by concluding remarks.

4.2 Cuckoo Search (CS)

The CS algorithm is a population-based meta-heuristic technique based on the brood parasitism trait of certain species of Cuckoo birds (Yang and Debb 2014). Cuckoos are very interesting birds, not only courtesy of the serene sounds they make but also due to their aggressive reproduction strategies. The design of the CS algorithm is simplified by the assumption of three main rules being:

(i) Each cuckoo lays just one egg at a time and it dumps this egg in a nest randomly,
(ii) The best nests with a high quality of eggs will carry over to the next generation, and,
(iii) The number of nests in which cuckoos can dump their eggs is fixed, and the eggs laid by cuckoos in these nests can be discovered by the host bird with a probability, $p_a \in [0, 1]$ (Yang and Debb 2009).

When the cuckoo egg is discovered, the egg is either thrown out of the nest, or the host bird abandons the nest and builds a new one. The last rule is further simplified by approximating p_a and replacing a fraction of the existing nests with new nests, which have new random solutions (Yang and Debb 2009). In solving maximization problems, the fitness of a solution can simply be proportional to the value of the objective function. To further simplify the implementation of the algorithm, it is assumed that each egg in a nest represents a solution, and a cuckoo egg represents a new solution. The objective is to use the new and potentially better solutions/cuckoos to replace the not so good solutions in the nests. These new solutions are given by (Yang and Debb 2009):

$$x_i^{(t+1)} = x_i^{(t)} + \propto \oplus L\acute{e}vy(\lambda), \tag{4.1}$$

where $\propto > 0$ is the step size which should be related to the scales of the problem of interest. More often than not, $\propto = 1$. Equation (4.1) essentially represents the stochastic equation for a random walk process. In a general sense, a random walk is a Markov chain whose next location is solely reliant upon the current location [the first term in Eq. (4.1)] and the transition probability [the second term in Eq. (4.1)]. The $\oplus$ symbol represents entrywise multiplications. The Lévy flight basically provides a random walk with the random step drawn from a Lévy distribution like so:

$$\text{Lévy} \sim u = t^{-\lambda}, \tag{4.2}$$

where t is the step-length and λ is a heavy-tailed continuous probability distribution generated from having a probability density function that follows the condition $1 < \lambda \leq 3$. The Lévy distribution has an infinite variance with an infinite mean.

The reason for applying the CS algorithm in this work, although it has been used in several domains, is courtesy of the fact that it has not been investigated in missing data estimation tasks. Also, the randomization of the motion equation is more efficient as the step-length is heavy-tailed, in addition to there being a low number of parameters

Table 4.1 Cuckoo search parameters

Parameter	Value
Number of nests	40
Discovery rate of eggs (p_a)	0.25
Maximum number of iterations	1000

which need tuning, thereby making the algorithm more generic to adapt to a wider range of optimization problems.

The CS algorithm has been utilized to minimize the generation and emission costs of a microgrid while satisfying system hourly demands and constraints (Vasanthakumar et al. 2015). In Wang et al. (2016), it was used to establish the parameters of chaotic systems via an improved cuckoo search algorithm. Ali and Mohamed (2016) presented a new hybrid algorithm comprised of the cuckoo search algorithm and the Nelder–Mead method with the aim being to solve the integer and minimax optimization problems.

The CS parameters used in the model implementation are given in Table 4.1 with the exception of the number of decision variables, which depend on the number of missing values in a record. The data was normalized to being in the range [0, 1], meaning the lower and upper bounds of the decision variables are 0 and 1, respectively. These parameters were chosen because they resulted in the more optimal outcomes with different combinations and permutations of values having been tested.

4.3 Proposed Approach

In this section, we present information on the proposed approach which uses a combination of a deep learning regression model, a deep autoencoder network, and an optimization technique, the CS algorithm, to approximate the missing data. Figure 4.1 illustrates how the regression model and the optimization method will be used.

Two predominant features of an autoencoder are; (i) its auto-associative nature, and (ii) the butterfly-like structure of the network resulting from a bottleneck trait in the hidden layers. Autoencoders are also ideal courtesy of their ability to replicate the input data by learning certain linear and non-linear correlations and covariances present in the input space, by projecting the input data into lower dimensions. The only condition required is that the hidden layer(s) have fewer nodes than the input layer, though it is dependent on the application. Prior to optimizing the regression model parameters, it is necessary to identify the network structure where the structure depends on the number of layers, the number of hidden units per hidden layer, the activation functions used and the number of input and output variables. After this, the parameters can then be approximated using the training set of data. The parameter approximation procedure is done for a given number of training cycles, with the optimal number of this being obtained by analyzing the validation error. The aim of

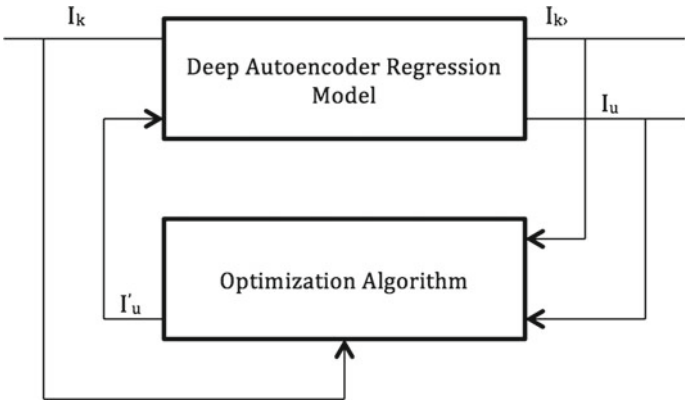

Fig. 4.1 Data imputation configuration. © 2017 Springer. Reproduced with permission from Leke et al. (2017)

this was to avoid overtraining the network and also to use the fastest training approach without compromising on the accuracy. The optimal number of training cycles was found to be 500. The training procedure estimates weight parameters such that the network output is as close as possible to the target output.

The CS algorithm is used to estimate the missing values by optimizing an objective function, which has as part of it the trained network. It will use values from the population as part of the input to the network, and the network will recall these values and they will form part of the output. The complete data matrix containing the estimated values and observed values will be fed into the autoencoder as input. Some inputs are known, with others unknown and they will be estimated using the regression method and the CS algorithm as described at the beginning of the paragraph. The symbols I_k and I_u as used in Figs. 4.1 and 4.2 represent the known and unknown/missing values, respectively.

Considering that the approach makes use of a deep autoencoder, it is imperative that the autoencoder architecture matches the output to the input. This trait is expected when a dataset with familiar correlations recorded in the network is used. The error used is the disparity between the target output and the network output, expressed as

$$\delta = \vec{I} - f\left(\vec{W}, \vec{I}\right), \tag{4.3}$$

where $\vec{I}$ and $\vec{W}$ represent the inputs and the weights, respectively.

The square of Eq. (4.3) is used to always guarantee that the error is positive. This results in the following equation:

$$\delta = \left(\vec{I} - f\left(\vec{W}, \vec{I}\right)\right)^2. \tag{4.4}$$

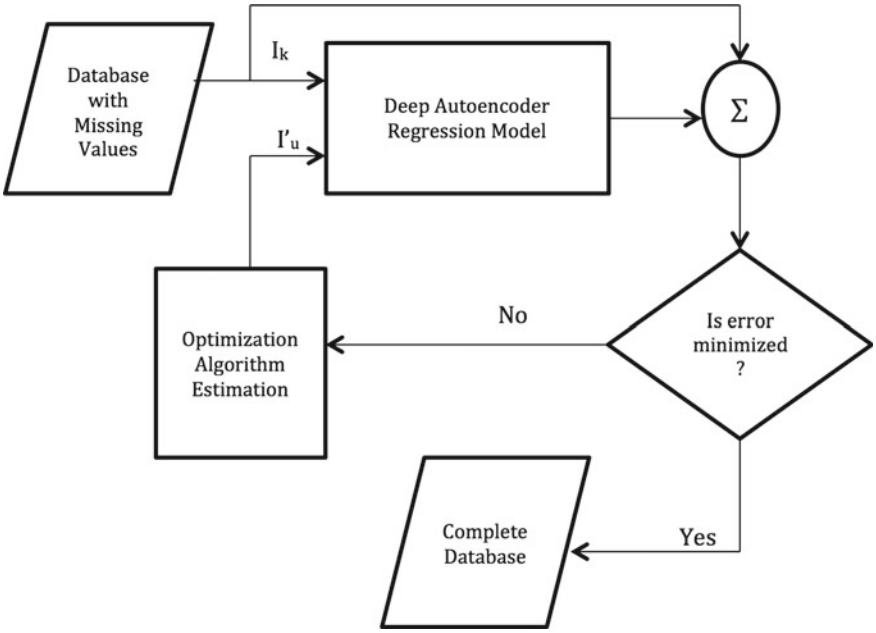

Fig. 4.2 Deep autoencoder and cuckoo search optimization missing data estimator structure. © 2017 Springer. Reproduced with permission from Leke et al. (2017)

Courtesy of the fact that the input and output vectors contain both I_k and I_u, the error function is rewritten as

$$\delta = \left(\begin{bmatrix} I_k \\ I_u \end{bmatrix} - f\left(\left\{ \begin{matrix} I_k \\ I_u \end{matrix} \right\}, w \right) \right)^2. \tag{4.5}$$

Equation (4.5) is the objective function used and minimized by the CS algorithm in order to estimate I_u, with f being the regression model function. From the above descriptions of how the deep autoencoder and the CS algorithm operate, the equation below summarizes the function of the proposed approach, with f_{CS} being the CS algorithm estimation operation and f_{DAE} being the function of the deep autoencoder.

$$y = f_{DAE}\left(W, f_{CS}\left(\vec{I}\right)\right), \tag{4.6}$$

where $\vec{I} = \begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}$ represents the input space of known and unknown features.

4.4 Experimental Analysis

To investigate and validate the proposed model against other existing approaches, the Mixed National Institute of Standards and Technology (MNIST) dataset of handwritten digits was used. It consists of 60,000 training samples and 10,000 test samples, each with 784 features representing the individual pixel values of the image. The black pixels are represented by 0, while the white pixels have values of 255, as such, all the variables could be considered to be continuous. The data in its current form has been cleaned of outliers, therefore, each image used in the dataset should contribute to generalizing the system. The data are normalized to be in the range [0, 1], and randomized to improve the network performance. The training set of data is further split into training and validation sets (50,000 and 10,000, respectively).

The neural network architecture is optimized using the validation set of data with the correctness in the performance of the proposed system being tested using the test data. It is worth mentioning that the objective of this research is not to propose a state-of-the-art classification model, which rivals existing state-of-the-art methods, but rather to propose an approach to reconstruct an image in the event of missing data (missing pixels). The optimized deep network architecture used is 784-1000-500-250-30-250-500-1000-784 as suggested initially by Hinton et al. (2006) and further verified by performing cross-validations using the validation set of data, with 784 input and output nodes, and 7 hidden layers with 1000, 500, 250, 30, 250, 500 and 1000 nodes, respectively. The objective function value (mean squared error) obtained after training was 1.96% which was close to the lowest value obtained during cross-validation (2.12%). The multilayer perceptron (MLP) autoencoder used for comparison purposes has the same number of nodes in the input and output layer as the deep autoencoder, with a single hidden layer consisting of 400 nodes which is determined by varying the number of nodes in the hidden layer, and determining which architecture produces the lowest network error. It was experimentally observed that the 784-400-784 MLP autoencoder network architecture yielded a network error value of 2.5088%. The sigmoid and linear activation functions are used for the hidden and output layers of the MLP network, respectively. The sigmoid activation function is used in each layer of the deep autoencoder except for the bottleneck layer, where the linear activation function is used. The training is done using the scaled conjugate gradient (SCG) algorithm for the MLP and the stochastic gradient descent (SGD) algorithm for the deep autoencoder network, for 500 epochs. 100 samples are selected at random from the test set of data and features are then removed respecting the missing at random (MAR) and missing completely at random (MCAR) mechanisms as well as the arbitrary missing data pattern. This is done by creating a binomial matrix of the same size as the test data with zeros and ones adhering to the stated mechanisms and pattern, and replacing every occurrence of one with NaN (implying missingness). The dataset obtained from this, with missing values is used to test the performance of the missing data estimation scheme, which is then compared against existing methods.

The effectiveness of the proposed approach is estimated using the mean squared error (MSE), the root mean squared logarithmic error (RMSLE), the correlation coefficient (r) and the relative prediction accuracy (RPA). Also used are the signal-to-noise ratio (SNR) and global deviation (GD). The mean squared and root mean squared logarithmic errors as well as the global deviation yield measures of the difference between the actual and predicted values, and provides an indication of the capability of the estimation.

$$\text{MSE} = \frac{\sum_{i=1}^{n}\left(I_i - \hat{I}_i\right)^2}{n}, \tag{4.7}$$

$$\text{RMSLE} = \sqrt{\frac{\sum_{i=1}^{n}\left(\log(I_i + 1) - \log\left(\hat{I}_i + 1\right)\right)^2}{n}}, \tag{4.8}$$

and

$$\text{GD} = \left(\frac{\sum_{i=1}^{n}\left(\hat{I}_i - I_i\right)}{n}\right)^2. \tag{4.9}$$

The correlation coefficient provides a measure of the similarity between the predicted and actual data. The output value of this measure lies in the range $[-1, 1]$ where the absolute value indicates the strength of the link, while the sign indicates the direction of the said link. Therefore, a value close to 1 signifies a strong predictive capability while a value close to -1 signifies otherwise. In the equation below, $\bar{I}$ represents the mean of the data.

$$r = \frac{\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)\left(\hat{I}_i - \bar{\hat{I}}_i\right)}{\left[\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)^2 \sum_{i=1}^{n}\left(\hat{I}_i - \bar{\hat{I}}_i\right)^2\right]^{1/2}}. \tag{4.10}$$

The relative prediction accuracy (RPA) on the other hand measures the number of estimates made within a specific tolerance, with the tolerance dependent on the sensitivity required by the application. The tolerance is set to 10% as it seems favorable for the application domain. This measure is given by

$$\text{RPA} = \frac{n_\tau}{n} * 100. \tag{4.11}$$

The squared error (SE) measures a quadratic scoring rule that records the average magnitude of the error. This can be obtained by calculating the variance square root, also referred to as the standard deviation. It reduces the variance in the error, contrary to the MSE, hence, its application in this work. SE can be obtained by using the following formula:

$$\mathrm{SE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(I_i - \hat{I}_i\right)^2}. \tag{4.12}$$

The mean absolute error (MAE) measures the average magnitude of the errors in a dataset without considering direction. Under ideal scenarios, SE values are always greater than the MAE values, and in case of equality, the error values are said to have the same magnitude. This error can be calculated using the following equation:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|I_i - \hat{I}_i\right|. \tag{4.13}$$

The coefficient of determination is a metric regularly applied in statistical analysis tasks aimed at assessing the performance of a model in the explanation and prediction of future outputs. It is also referred to as the R-squared statistic obtained by the following:

$$\mathrm{COD} = \left(\frac{\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)\left(\hat{I}_i - \bar{\hat{I}}_i\right)}{\left[\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)^2 \sum_{i=1}^{n}\left(\hat{I}_i - \bar{\hat{I}}_i\right)^2\right]^{1/2}}\right)^2. \tag{4.14}$$

The signal-to-noise ratio used in this chapter is obtained by

$$\mathrm{SNR} = \frac{\mathrm{var}\left(I - \hat{I}\right)}{\mathrm{var}\left(\hat{I}\right)}, \tag{4.15}$$

where in Eqs. (4.7)–(4.10) and Eqs. (4.12)–(4.14), n is the number of samples while I and $\hat{I}$ represent the real test set values and predicted missing output values, respectively. In Eq. (4.11), n_r represents the number of correctly predicted outputs.

Taking into consideration the performance metrics, the performance of the DL-CS method was evaluated and compared against existing methods by estimating the missing attributes concurrently, wherever missing data may be ascertained. The scenarios investigated were such that any sample/record could have at least 62, and at most 97 missing attributes (dimensions) to be approximated. Figures 4.3, 4.4, 4.5 and 4.6 show the performance and comparison of DL-CS with MLP-PSO, MLP-SA and MLP-GA. The approach introduced as will be observed, differs significantly from existing methods which cater to cases whereby one or no more than 10 missing values are estimated per instance.

Due to the execution times, the results presented are averaged over only 5 runs of the models. Figures 4.3, 4.4, 4.5, 4.6 and 4.9 show the performance and comparison of DL-CS with MLP-PSO, MLP-SA and MLP-GA. Before heading to the analysis of the results, note that the mean squared error is a deviation that measures the average

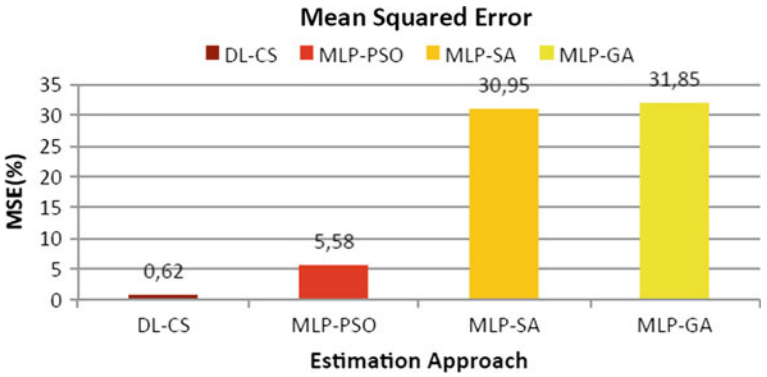

Fig. 4.3 Mean squared error versus estimation approach

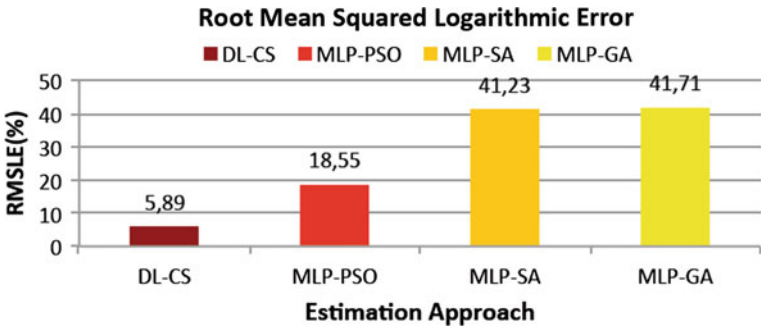

Fig. 4.4 Root mean squared logarithmic error versus estimation approach

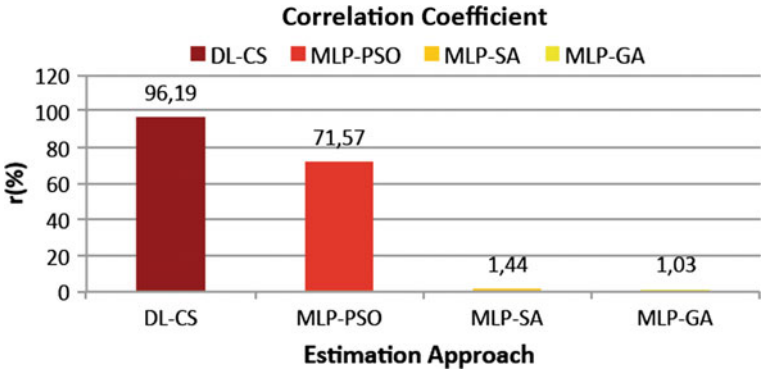

Fig. 4.5 Correlation coefficient versus estimation approach

of the squares of the errors. The optimum performance of an estimator corresponds to a minimum value of MSE. Another measure is the root mean squared logarithmic error, which measures the difference between samples predicted by the models. As

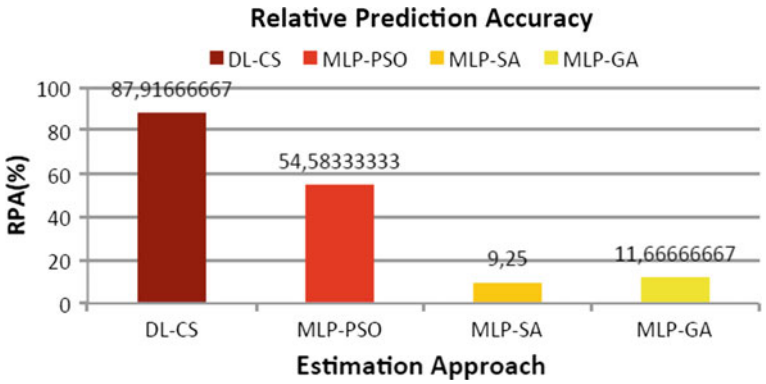

Fig. 4.6 Relative prediction accuracy versus estimation approach

in the case of MSE, the proposed approach gives better performance yielding the minimum value of the RMSLE. In evaluating data mining approaches, we also look at the correlation coefficient. It quantifies the type of dependence between samples. An estimator performs better when the correlation coefficient is higher. As with the correlation coefficient, the prediction accuracy is also given in percentage and represents the mean absolute deviation with the optimum accuracy corresponding to a 100%. Figures 4.3 and 4.4 are bar charts that show the MSE and RMSLE values for DL-CS when compared to MLP-PSO, MLP-SA and MLP-GA.

We observed 0.62, 5.58, 30.95 and 31.85% of MSE and, 5.89, 18.55, 41.23 and 41.71% of RMSLE for DL-CS, MLP-PSO, MLP-SA and MLP-GA, respectively. DL-CS yielded the lowest MSE value when compared to the others. These results are validated by the correlation coefficient whose bar chart is given in Fig. 4.5.

DL-CS and MLP-PSO yielded 96.19 and 71.57% correlation values, respectively, while MLP-SA and MLP-GA showed correlations of 1.44 and 1.03%, respectively.

MLP-SA and MLP-GA yielded 9.25 and 11.67% of RPA respectively, while DL-CS and MLP-PSO, respectively, yielded 87.92 and 54.58%, as shown in Fig. 4.6. As observed, the DL-CS approach obtained the best figures for all four metrics presented graphically.

In the top row of Fig. 4.7, we depict 10 images with missing pixel values which are to be estimated prior to classification tasks being performed by statistical methods. In the bottom row of the same figure, we show the reconstructed images from using the DL-CS approach, while in the top and bottom rows of Fig. 4.8, we observe the reconstructed images when the MLP-PSO and MLP-GA approaches are used, respectively. It can be seen that the reconstructed images using MLP-PSO and MLP-GA introduce a lot of noise, more so in the bottom row than in the top row, as opposed to when the DL-CS approach is applied. Furthermore, closer inspection reveals that the images are not fully reconstructed as not all pixel values within the images are estimated correctly when the MPL-PSO and MLP-GA approaches are used.

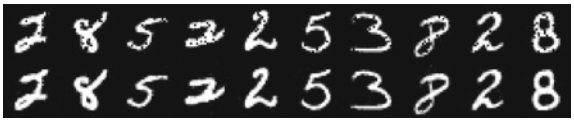

Fig. 4.7 Top row: corrupted images—Bottom row: DL-CS-reconstructed images

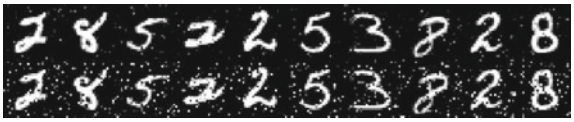

Fig. 4.8 Top row: MLP-PSO reconstructed images—Bottom row: MLP-GA-reconstructed images

Table 4.2 DL-CS mean squared error objective value per sample

Sample	Dimensions	DL-CS	MLP-PSO	MLP-SA	MLP-GA
1	83	**2.89**	5.72	9.26	9.26
2	75	**2.84**	8.94	14.22	14.22
3	85	**1.29**	5.73	6.77	6.77
4	74	**3.45**	7.72	16.06	16.06
5	66	**1.78**	6.79	10.33	10.33
6	74	**1.10**	5.37	9.12	9.12
7	82	**3.19**	9.31	11.79	11.79
8	77	**2.97**	10.38	14.64	14.64
9	74	**3.51**	8.35	8.49	8.49
10	81	**1.25**	5.67	15.36	15.36

The bold values indicate that the algorithm did yield the best (lowest) results

Table 4.3 DL-CS additional metrics

	DL-CS	MLP-PSO	MLP-SA	MLP-GA
COD	**92.52**	51.22	0.02	0.01
GD	**0.01**	1.23	14.93	15.22
MAE	**3.57**	15.17	47.63	48.07
SE	**7.85**	23.61	55.63	56.44
SNR	**8.36**	60.35	194.62	189.37

The bold values indicate that the algorithm did yield the best (lowest) results

In Table 4.2, the dimensions column refers to the number of missing values in a sample/record. Tables 4.2 and 4.3 further back the findings from Figs. 4.3, 4.4, 4.5 and 4.6 showing that the proposed DL-CS approach yielded the lowest objective function value in the estimation of missing values in each sample, as well as the best COD, GD, MAE, SE and SNR values.

In Table 4.4, we present results obtained from statistically analysing the estimates obtained by the DL-CS approach when compared to the MLP-PSO, MLP-SA and

Table 4.4 Statistical analysis of DL-CS results

Pairs compared	*P*-values (95% confidence level)
DL-CS:MLP-PSO	3.7 * 10e−19
DL-CS:MLP-SA	4.6 * 10e−50
DL-CS:MLP-GA	4.6 * 10e−50

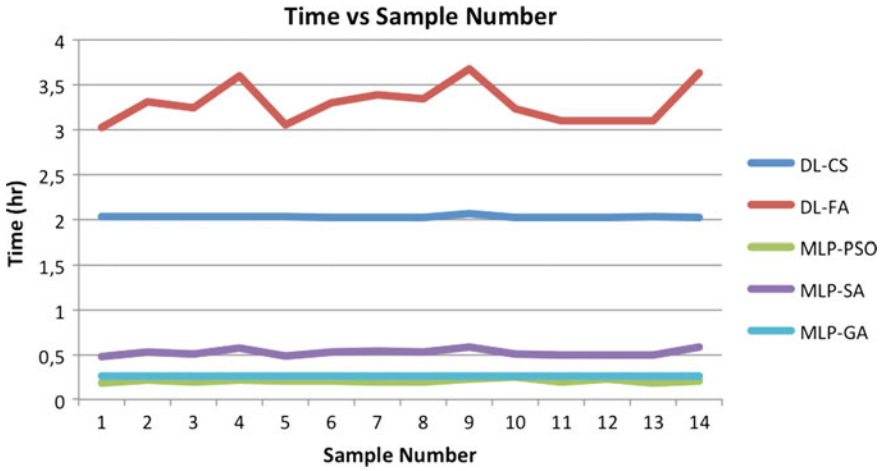

Fig. 4.9 Time versus sample number. © 2017 Springer. Reproduced with permission from Leke et al. (2017)

MLP-GA approaches using the t-test. The t-test null hypothesis (H_0) assumes that there is no significant difference in the means of the missing data estimates obtained by the DL-CS, MLP-PSO, MLP-SA and MLP-GA methods. The alternative hypothesis (H_A), however, indicates that there is a significant difference in the means of the missing data estimates obtained by the four methods.

Table 4.4 reveals that there is a significant difference at a 95% confidence level in the means of the estimates obtained by DL-CS when compared to MLP-PSO, MLP-SA and MLP-GA, yielding *p*-values of 3.7 × 10e−19, 4.6 × 10e−50 and 4.6 * 10e−50, respectively, when all three pairs are compared. This, therefore, indicates that the null hypothesis (H_0), which assumes that there is no significant difference in the means between DL-CS and the other three methods can be rejected in favour of the alternative hypothesis (H_A) at a 95% confidence level.

The execution time is the last metric used in this analysis. MLP-PSO, MLPSA and MLP-GA have always been outperformed by DL-CS. This is justified by their execution time which are relatively short, less than 30 min, when compared to DL-CS as shown in Fig. 4.9

4.5 Conclusion

This chapter presented an investigation of the estimation of missing data via a novel approach. The estimation method comprises of a deep autoencoder network to replicate the input data, in combination with the cuckoo search algorithm to estimate the missing data. The performance of the model is investigated and compared to existing methods including an MLP autoencoder with Genetic Algorithm, Simulated Annealing and Particle Swarm Optimization. The results obtained reveal that the proposed system yields more accurate estimates when compared against the MLP hybrid systems. This is made evident when the mean squared error, root mean squared logarithmic error, correlation coefficient, relative prediction accuracies, signal-to-noise ratio and global deviation are taken into account, with the proposed approach yielding the best values of these. Also, when the objective function value is considered during the estimation process, it is observed that the proposed approach results in the lowest values for this for each instance.

References

Abdella, M., & Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. In *3rd International Conference on Computational Cybernetics, (ICCC)* (pp. 207–212). IEEE.

Ali, F. A., & Mohamed, A. T. (2016). A hybrid cuckoo search algorithm with NelderMead method for solving global optimization problems. *SpringerPlus*, *5*(1), 473. https://doi.org/10.1186/s40064-016-2064-1. Springer International Publishing.

Brain, L. B., Marwala, T., & Tettey, T. (2006). Autoencoder networks for HIV classification. *Current Science, 91*(11), 1467–1473.

Finn C., Tan, X., Duan, Y., Darrell, T., Levine, S., & Abbeel, P. (2016). Deep spatial autoencoders for visuomotor learning. In *International Conference on Robotics and Automation (ICRA)* (pp. 512–519).

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.

Jerez, J. M., Molina, I., García-Laencina, P. J., Alba, E., Ribelles, N., Martín, M., et al. (2010). Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine, 50*(2), 105–115. Elsevier.

Ju, Y., Guo, J., & Liu, S. (2015). A deep learning method combined sparse autoencoder with SVM. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (Cyber C)* (pp. 257–260).

Krizhevsky, A., & Hinton, G. E. (2011). Using very deep autoencoders for content based image retrieval. In *19th European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium. 27–29 April 2011.

Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference in Swarm Intelligence* (pp. 259–270). Springer International Publishing.

Leke, C., Twala, B., & Marwala, T. (2014). Modeling of missing data prediction: Computational intelligence and optimization algorithms. In *International Conference on Systems, Man and Cybernetics (SMC)* (pp. 1400–1404). IEEE.

Leke, C., Ndjiongue, A. R., Twala, B., & Marwala, T. (2017). A deep learning-cuckoo search method for missing data estimation in high-dimensional datasets. *International Conference in Swarm Intelligence* (pp. 561–572). Cham: Springer.

Liew, A. W.-C., Law, N.-F., & Yan, H. (2011). Missing value imputation for gene expression data: computational techniques to recover missing data from available information. *Briefings in Bioinformatics, 12*(5), 498–513. Oxford University Press.

Myers, T. A. (2011). Goodbye, listwise deletion: Presenting hot deck imputation as an easy and effective tool for handling missing data. *Communication Methods and Measures*, *5*(4), 297–310. Taylor & Francis.

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods, 7*(2), 147. American Psychological Association.

Ssali, G., & Marwala, T. (2007). Estimation of missing data using computational intelligence and decision trees. arXiv preprint arXiv:0709.1640.

Vasanthakumar, S., Kumarappan, N., Arulraj, R., & Vigneysh, T. (2015). Cuckoo search algorithm based environmental economic dispatch of microgrid system with distributed generation. In *International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)* (pp. 575–580). IEEE.

Vukosi, M. N., Nelwamondo, F. V., & Marwala, T. (2007). Autoencoder, principal component analysis and support vector regression for data imputation. arXiv preprint arXiv:0709.2506.

Wang, J., Zhou, B., & Zhou, S. (2016). An improved cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation. *Computational Intelligence and Neuroscience*, 8. https://doi.org/10.1155/2016/2959370.

Yang, X. S., & Debb, S. (2014). Cuckoo search: Recent advances and applications. *Neural Computing and Applications, 24*(1), 169–174.

Yang, X. S., & Debb, S. (2009). Cuckoo search via levy flights. *World Congress on Nature and Biologically Inspired Computing (NaBIC), 48*(2), 210–214.

Chapter 5
Missing Data Estimation Using Firefly Algorithm

5.1 Introduction

Previous research across a wide range of academic fields suggests that decision-making and data analysis tasks are made non-trivial by the presence of missing data. As such, it can be assumed that decisions are likely to be more accurate and reliable when complete/representative datasets are used instead of incomplete datasets. This assumption has led to a great deal of research in the data mining domain, with novel techniques developed to perform this task accurately (Abdella and Marwala 2005; Aydilek and Arslan 2012; Koko and Mohamed 2015; Leke et al. 2014; Mistry et al. 2009; Nelwamondo et al. 2007; Rana et al. 2015; Zhang et al. 2011; Zhang 2011). Research suggests that applications in various professional fields such as in medicine, manufacturing or energy that use sensors in instruments to report vital information and enable decision-making processes may fail and lead to incorrect outcomes due to the presence of missing data. In such cases, it is very important to have a system capable of imputing the missing data from the failed sensors with high accuracy. The imputation procedure will require the approximation of missing values taking into account the interrelationships that exist between the data from sensors in the system. Another instance where the presence of missing data poses a threat in decision-making is in image recognition systems, whereby the absence of pixel values renders the image prediction or classification task difficult and as such, systems capable of imputing the missing values with high accuracy are needed to make the task more feasible.

Consider a high-dimensional dataset such as the Mixed National Institute of Standards and Technology (MNIST) dataset with 784 feature variables being the pixel values as shown in Fig. 5.1. Assuming that pixel values are missing at random as observed in the bottom row and a statistical analysis is required to classify the above dataset, the questions of interest would be: (i) Can we impute with some degree of certainty the missing data in high-dimensional datasets with high accuracy? (ii) Can new techniques be introduced for the approximation of the missing data when cor-

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_5

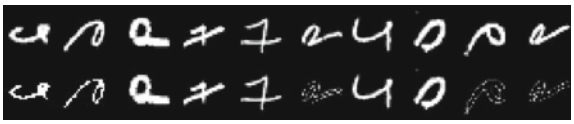

Fig. 5.1 Sample of MNIST dataset. Top row—real data: Bottom row—data with missing pixel values. © 2016 Springer. Reproduced with permission from Leke and Marwala (2016)

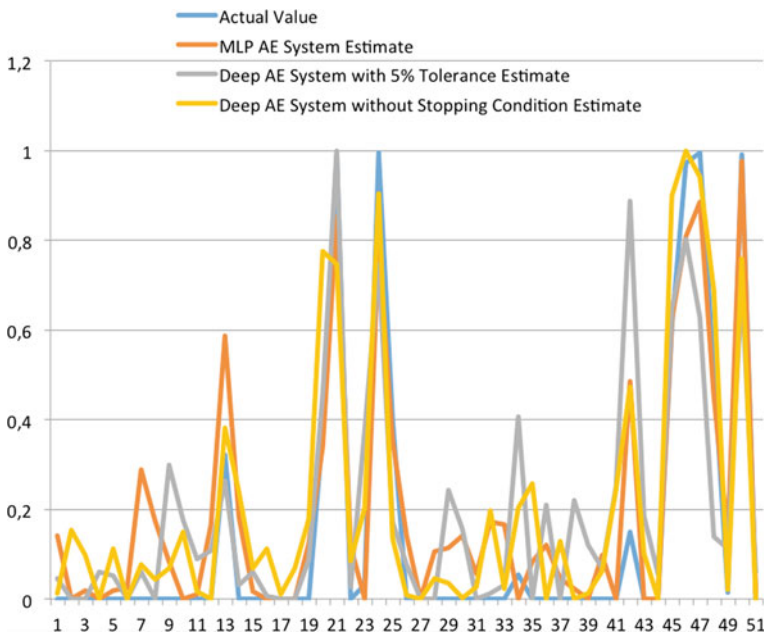

Fig. 5.2 Actual vs estimated values. © 2016 Springer. Reproduced with permission from Leke and Marwala (2016)

relation and interrelationships between the variables are considered? This chapter, therefore, aims to use a deep learning (DL) technique built with restricted Boltzmann machines stacked together to form an autoencoder in tandem with the firefly algorithm (FA) to estimate the missing data with the model created which would cater to the mechanisms of interest and the arbitrary pattern. The dataset used is the MNIST database of handwritten digits by LeCun (2016). It has a training set of 60,000 sample images and a test set of 10,000 sample images with 784 features. These images show handwritten digits from 0 to 9. Due to the fact that the research discussed in this chapter was conducted at a time when there was little or no interest in the DL-FA missing data predictors on high-dimensional data, this chapter seeks to exploit the use of this technique on the MNIST dataset.

The remainder of this chapter is structured as follows, Sect. 5.2 introduces missing data, the deep learning techniques used as well as the swarm intelligence algorithm

implemented. Section 5.3 presents the experimental design and procedures used, while Sect. 5.4 focuses on the results and key findings from the experiments conducted in this chapter. Discussions, concluding remarks and suggestions for future research are further presented in Sect. 5.5.

5.2 Review of Literature

This chapter implements a deep learning technique referred to as a stacked autoencoder built using restricted Boltzmann machines, all of which have been individually trained using the contrastive divergence algorithm and stacked together in a bottom-up manner. The estimation of missing values is performed by using the firefly algorithm, which is the swarm intelligence method. However, we will first briefly discuss the methods used and the problem they aim to solve.

5.2.1 Missing Data

Missing data is a situation whereby some features within a dataset are lacking components (Rubin 1978). With this ensues problems in application domains that rely on the access to complete and quality data which can affect every academic/professional fields and sectors. Techniques aimed at rectifying the problem have been an area of research in several disciplines (Rubin 1978; Allison 1999; Little and Rubin 2014). Missing data may occur because of one or all of the following reasons, (i) non-responses to questions during a survey or data entry process, (ii) failures in sensors or instruments in the data recording process, (iii) improper data collection procedures by researchers.

5.2.1.1 Missing Data Mechanisms

The manner in which data points go missing in a dataset determines the approach to be used in estimating these values. As per Little and Rubin (2014), there exist three missing data mechanisms. These are: (i) missing completely at random (MCAR), (ii) missing at random (MAR), and (iii) missing not at random or non-ignorable case (MNAR or NI). This chapter focuses on investigating the MCAR and MAR mechanisms. Previous research suggests that MCAR scenario arises when the chances of there being a missing data entry for a feature is not dependent on the feature itself or on any of the other features in the dataset (Leke et al. 2014). This implies a lack of correlation or cross-correlation between features including the feature of interest (Rubin 1978). MAR on the other hand arises when missingness in a specific feature is reliant upon the other features within the dataset, but not the feature of interest itself (Leke et al. 2014).

5.2.1.2 Missing Data Patterns

According to Little and Rubin (2014), there are two main missing data patterns. These are the arbitrary and monotone missing data patterns. In addition to these two patterns, there is the univariate missing data pattern. In the arbitrary pattern, missing observations may occur anywhere, and the ordering of the variables is of no importance. In monotone missing patterns, the ordering of the variables is of importance and occurrence is not random. As such in a dataset with several feature variables say, $X_1, X_2,\ldots, X_n$, a monotone missing pattern is present if a variable X_j which is observed for a particular instance, implies that all the previous variables X_k, where $k < j$, are also observed for that instance (Little and Rubin 2014). The univariate pattern defines a scenario whereby only one feature variable has missing data entries. Based on this realization, this chapter will go on to focus on the arbitrary missing pattern.

5.2.2 Deep Learning (DL)

Deep learning comprises of several algorithms in machine learning that make use of a cataract of non-linear processing units organized into a number of layers that extract and transform features from the input to the output data (Deng et al. 2013; Deng and Yu 2014). Each of the layers uses the output from the previous layer as input and a supervised or unsupervised algorithm could be used in the training phase. With these come applications in supervised and unsupervised problems like classification and pattern analysis, respectively. It is also based on the unsupervised learning of multiple levels of features or representations of the input data whereby higher level features are obtained from lower level features to yield a hierarchical representation of the data (Deng and Yu 2014). By learning multiple levels of representations that depict different levels of abstraction of the data, we obtain a hierarchy of concepts. In this article, the deep learning technique used is the stacked autoencoder.

5.2.3 Restricted Boltzmann Machine (RBM)

First, a Boltzmann machine (BM) is a bidirectionally connected network of stochastic processing units, which can be interpreted as a neural network. It can be used to learn important aspects of an unknown probability distribution based on samples from the distribution, which is generally a difficult process. This learning process can be simplified by imposing restrictions on the network topology leading to restricted Boltzmann machines (RBMs). An RBM can be described as an undirected, probabilistic, parameterized graphical model also known as a Markov random field (MRF). RBMs have received a lot of interest after being suggested as building blocks of multilayer architectures called deep networks (Fischer and Igel 2012). The idea is that

hidden neurons extract relevant features from the observations, which subsequently serve as input to another RBM. Stacking these RBMs together has as objective, obtaining high-level representations of data by learning features from features. An RBM which is also an MRF associated with a bipartite undirected graph consists of m visible units, $V = (V_1, V_2, \ldots, V_m)$ to represent observable data, and n hidden units, $H = (H_1, H_2, \ldots, H_n)$ that capture interdependencies between features in the input layer (Fischer and Igel 2012). In this chapter, the variables V take on values, v $[0,1]^{m+n}$, while H take on values, h $\{0,1\}^{m+n}$. The joint probability distribution given by the Gibbs distribution has as energy function (Fischer and Igel 2012):

$$E(v, h) = -h^T W v - b^T v - c^T h. \tag{5.1}$$

In scalar form, (5.1) is expressed as Fischer and Igel (2012):

$$E(v, h) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i. \tag{5.2}$$

In Eq. (5.2), w_{ij}, which is the most important part of an RBM model is a real-valued weight between units V_j and H_i, while b_j and c_i are real-valued bias terms associated with the jth visible and ith hidden variable, respectively. If w_{ij} is negative, and v_j and h_i are equal to one, the probability decreases leading to a high energy. On the contrary, if w_{ij} is positive, and v_j and h_i are equal to zero, the probability increases leading to a lower energy. If b_j is negative and $v_j = 1$, E increases leading to a low probability. Therefore, there is a preference for $v_j = 0$ instead of $v_j = 1$. However, if b_j is positive and $v_j = 0$, E decreases leading to a high probability, and a preference for $v_j = 1$ instead of $v_j = 0$. A negative b_j value decreases the second term in Eq. (5.2), while a positive value for b_j increases this second term. The same applies for c_i and the third term in Eq. (5.2). The Gibbs distributions or probabilities from Eqs. (5.1) or (5.2) are then obtained by Fischer and Igel (2012):

$$\begin{aligned} p(v, h) &= \frac{e^{-E(v,h)}}{Z} \\ &= \frac{e^{\left(h^T W v + b^T v + c^T h\right)}}{Z} \\ &= \frac{e^{\left(h^T W v\right)} e^{\left(b^T v\right)} e^{\left(c^T h\right)}}{Z}. \end{aligned} \tag{5.3}$$

Here, the exponential terms are factors of a Markov network with vector nodes, while Z is the intractable partition function. It is intractable courtesy of the exponential number of values it can take. For an RBM,

$$Z = \sum_{v,h} e^{-E(v,h)}. \tag{5.4}$$

Another key aspect of RBMs is that h is conditionally independent of v and vice versa, due to the fact that there are no connections between nodes in the same layer. This property is expressed mathematically as Fischer and Igel (2012):

$$p(h|v) = \prod_{i=1}^{n} p(h_i|v)$$

and

$$p(v|h) = \prod_{i=1}^{m} p(v_i|h). \tag{5.5}$$

5.2.4 Contrastive Divergence (CD)

The objective in training an RBM is to minimize the average negative log-likelihood (loss) without regularization using a stochastic gradient descent algorithm as it scales well with high-dimensional datasets. This loss function could be expressed as

$$Loss = \frac{1}{T} \sum_{t} -\log p\left(v^{(t)}\right). \tag{5.6}$$

Achieving this objective requires the partial derivative of any parameter, θ, of the loss function as per the following equation:

$$\frac{\partial\left(-\log p\left(v^{(t)}\right)\right)}{\partial\theta} = E_h\left[\frac{\partial E\left(v^{(t)}, h\right)}{\partial\theta}|v^{(t)}\right] - E_{v,h}\left[\frac{\partial E(v,h)}{\partial\theta}\right]. \tag{5.7}$$

The first term in Eq. (5.5) is the expectation over the data distribution and is referred to as the positive phase, while v and h represent the same variables as in Eqs. (5.1)–(5.4). The second term, which is the expectation over the model distribution is termed the negative phase. This phase is hard to compute and also intractable because an exponential sum is required over both h and v. Furthermore, obtaining unbiased estimates of the log-likelihood gradient typically requires many sampling steps. However, it has been shown recently that estimates obtained after running a Markov chain for just a few steps can be sufficient for training a model. This approach has led to the contrastive divergence (CD) algorithm. CD is a training method for undirected probabilistic graphical models with the idea being to do away with the double expectations in the negative phase in Eq. (5.5) and instead focus on estimation. It basically implements a Monte Carlo estimate of the expectation over a single input data point. The idea of k-step CD learning (CD-k) is that instead of approximating the second term in Eq. (5.5) by a sample from the model distribution, a Gibbs chain is run for only k steps, with k usually set to 1. The Gibbs chain is initialized with

a training example $v^{(0)}$ of the training set and yields the sample $v^{(k)}$ after k steps. Each step, t, consists of sampling $h^{(t)}$ from $p(h|v^{(t)})$ and sampling $v^{(t+1)}$ from $p(v|h^{(t)})$ subsequently. The gradient of the log-likelihood with respect to θ for one training pattern, $v^{(0)}$, is approximated by

$$CD_k\left(\theta, v^{(0)}\right) = -\sum_h p\left(h|v^{(0)}\right)\frac{\partial E\left(v^{(0)}, h\right)}{\partial \theta} + \sum_h p\left(h|v^{(k)}\right)\frac{\partial E\left(v^{(k)}, h\right)}{\partial \theta}. \tag{5.8}$$

Due to the fact that $v^{(k)}$ is not a sample from the stationary model distribution, the approximation Eq. (5.6) is biased. The bias in effect vanishes as $k \to \infty$. Another aspect that points to CD being biased is that it maximizes the difference between two Kullback–Liebler (KL) divergences:

$$KL(q|p) - KL(p_k|p). \tag{5.9}$$

Here, q is the empirical distribution and p_k is the distribution of the visible variables after k steps of the Markov chain (Fischer and Igel 2012). If the chain already reached stationarity, it holds that $p_k = p$, and thus $KL(p_k|p) = 0$, and the approximation error of CD vanishes (Fischer and Igel 2012).

5.2.5 *Autoencoder (AE)*

An autoencoder is an artificial neural network that attempts to reproduce its input at the output layer. The basic idea behind autoencoders is that the mapping from the input to the output, $x^{(i)} \to y^{(i)}$ reveals vital information and the essential structure in the input vector that is otherwise abstract. An autoencoder takes an input vector x and maps it to a hidden representation y via a deterministic mapping function f_θ of the form (Isaacs 2014; Leke and Marwala 2016):

$$f_\theta(x) = s(Wx + b). \tag{5.10}$$

The θ parameter comprises of the matrix of weights W and the vector of offsets/biases b. s is the sigmoid activation function expressed as

$$s = \frac{1}{1 + e^{-x}}. \tag{5.11}$$

The hidden representation y is then mapped to a reconstructed vector z, which is obtained by the functions (Leke and Marwala 2016):

$$z = g_{\theta'}(y) = s\left(W'y + b'\right)$$

or

$$z = g_{\theta'}(y) = W'y + b'. \tag{5.12}$$

Here, the parameter set θ' comprises of the transpose of the matrix of weights and vector of biases from the encoder prior to the fine-tuning phase Leke and Marwala (2016). When the aforementioned transposition of weights is done, the autoencoder is said to have tied weights. The parameter z is not explained as a rigorous regeneration of x but rather as the parameters of a distribution $p(X|Z=z)$ in probabilistic terms, that may yield x with high probability (Isaacs 2014). This, thus, leads to

$$p(X|Y = y) = p(X|Z = g_{\theta'}(y)). \tag{5.13}$$

From this, we obtain an associated reconstruction error which is to be optimized by the optimization technique and is of the form $L(x, z) \propto -\log p(x|z)$. This equation as per Bengio et al. (2013) could also be expressed as

$$\delta_{AE}(\theta) = \sum_t L\left(x^{(t)}, g_\theta\left(f_\theta\left(x^{(t)}\right)\right)\right) \tag{5.14}$$

5.2.6 *Firefly Algorithm (FA)*

FA is a nature-inspired meta-heuristic algorithm based on the flashing patterns and behaviour of fireflies (Yang 2010). It is based on three main rules being:

(i) Fireflies are unisex, so all fireflies are attracted to all other fireflies,
(ii) Attractiveness is proportional to the brightness and they both decrease as the distance increases. The idea is the less bright firefly will move towards the brighter one. If there is no obvious brighter firefly, they move randomly, and,
(iii) Brightness of a firefly is determined by the landscape of the objective function (Yang 2010).

Considering that attractiveness is proportional to light intensity, the variation of attractiveness can be defined with respect to the distance as

$$\beta = \beta_0 e^{-\gamma r^2}. \tag{5.15}$$

In Eq. (5.15), β is the attractiveness of a firefly, β_0 is the initial attractiveness of a firefly and r is the distance between two fireflies. The movement of a firefly towards a brighter one is determined by

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j^t - x_i^t\right) + \alpha_i \varepsilon_i^t. \tag{5.16}$$

Here, x_i and x_j are the positions of two fireflies, and the second term is due to the attraction between the fireflies. t and $t+1$ represent different time steps, α is the randomization parameter controlling the step size in the third term, while ϵ is a vector

of random numbers drawn from a Gaussian or uniform distribution. If $\beta_0 = 0$, the movement becomes a simple random walk (Yang 2010). If $\gamma = 0$, the movement reduces to a variant of the particle swarm optimization algorithm (Yang 2010). The parameters used in this research are: (i) 25 fireflies, (ii) 1000 iterations, (iii) $\alpha = 0.5$, (iv) $\beta = 0.2$ and (v) $\gamma = 1$. The parameters were selected as they yielded more optimal results after experimentation with different permutations and combinations of values. The FA algorithm is used because although it has been applied in a number of domains such as digital image compression, eigenvalue optimization, feature selection and fault detection, scheduling and TSP, etc., its efficiency has not been investigated in missing data estimation tasks.

5.3 Experimental Design and Procedure

In the design of the experiment, MATLAB® R2014a software was used on a Dell Desktop computer with Intel® Core ™ i3-2120 CPU @ 3.30 GHz processor, 4.00 GB RAM, 32 GB virtual RAM, 64-bit Operating System running Windows 8.1 Pro. Additionally, the MNIST database was used and it contains 60,000 training images and 10,000 test images. Each of these images is of size $28 \times 28 = 784$ pixels. This results in a training set of size $60{,}000 \times 784$ and a test of size $10{,}000 \times 784$. Data preprocessing was performed by normalizing all pixel values in the range [0, 1]. The individual network layers of the Deep AE were pretrained using RBMs and CD to initialize the weights and biases in a good solution space. The individual layers pre-trained were of size 784-1000, 1000-500, 500-250 and 250-30. These are stacked and subsequently transposed to obtain the encoder and decoder parts of the autoencoder network, respectively. The resulting network architecture is of size, 784-1000-500-250-30-250-500-1000-784, with an input and output layer having the same number of nodes, and seven hidden layers with varying number of nodes. The network is then fine-tuned using back-propagation, minimizing the mean-squared network error. The error value obtained after training is 0.0025. The training is done using the entire training set of data that are divided into 600 balanced mini-batches each containing 10 examples of each digit class. The weights and biases are updated after each mini-batch. Training higher layers of weights are achieved by having the real-valued activities of the visible units in the RBM being transcribed as the activation probabilities of the hidden units in the lower level RBM. The multilayer perceptron (MLP) AE has an input and output layer, both consisting of 784 nodes, and one hidden layer consisting of 400 nodes obtained by experimenting with different numbers of nodes in the hidden layer, and observing which architecture yields the lowest mean squared network error. A 784-400-784 network architecture led to the lowest mean squared network error value of 0.0032. The hidden and output layer activation function used is the sigmoid function. The training is done using the scaled conjugate gradient descent algorithm for 1000 epochs. Missingness in the test

set of data is then created at random according to the MAR and MCAR mechanisms, as well as the arbitrary pattern, and these missing values are approximated using the firefly algorithm which has as objective function minimizing the loss function of the fine-tuned network. The tolerance error is initially set to 0.05 (5%) in one of the networks and is considered reasonable for a first-time investigation of the proposed method. A matrix of the same size as the test set of data is created with values obtained from a binomial distribution with the required percentage of missingness (10%), which is then superimposed on the test set to incorporate the intended missing data. The overall approach consists of four consecutive steps listed below:

1. Train the individual RBMs on a training set of data with complete records using the greedy layer-by-layer pretraining algorithm described in Hinton and Salakhutdinov (2006) starting from the bottom layer. Each layer is trained for 50 epochs.
2. Stack the RBMs to form the encoder and decoder phases of a deep autoencoder with tied weights.
3. Fine-tune the deep autoencoder using back-propagation via means of the stochastic gradient descent technique.
4. Estimate the missing data with the fine-tuned deep network as part of the objective function in the firefly algorithm parsing the known variable values to the objective function, while first estimating the unknown values before parsing these estimates to the objective function. The estimation procedure is terminated when a stopping criterion is achieved, which is either an error tolerance of 5% (0.05), or the maximum number of function evaluations being attained.

5.4 Experimental Results

In the investigation of the imputation technique, we used the test set of data which contained missing data entries accounting for approximately 10% of the data. We present in Tables 5.1 and 5.2, actual, estimate and squared error values from the proposed deep autoencoder system without tolerance (Table 5.1) for some missing data entries, and from MLP Autoencoder system (Table 5.2). The distance, ϵ, from the estimate to the actual value, added to the squared error are parameters that determine the performance of the method. In all cases presented in both tables, the Deep Autoencoder system shows $\epsilon_d = 0$, 0.0608, 0, 0.0275, 0, 0.0922, 0.0009, 0.0283, while for the same entries (actual values), the MLP autoencoder shows that $\epsilon_m =$ 0.0246, 0.2646, 0.0149, 0.1643, 0, 0.1982, 0.0509, 0.0473, respectively. They show better performance of the proposed technique without a set error tolerance when compared to the existing MLP autoencoder. This knowledge is validated by the squared error which is always smaller for the proposed technique, for all cases presented in Tables 5.1 and 5.2.

Table 5.1 Actual, estimated and squared error values from deep autoencoder system without set tolerance

Actual	Estimate	Squared error
0	0	0
0.3216	0.3824	0.0037
0	0	0
0.9725	1	0.0008
0	0	0
0.9961	0.9039	0.0085
0.0509	0.0500	8.38e-07
0.5765	0.6048	0.0008

Table 5.2 Actual, estimated and squared error values from MLP autoencoder system

Actual	Estimate	Squared error
0	0.0246	0.0006
0.3216	0.5862	0.0700
0	0.0149	0.0002
0.9725	0.8082	0.0270
0	0	0
0.9961	0.7979	0.0393
0.0509	0	0.0026
0.5765	0.5292	0.0022

We could consider this enough to conclude of on the performance of both compared techniques, but we need to analyse the processing time, which seems to be better for the existing method when compared to the proposed deep autoencoder system. This is demonstrated by Fig. 5.3, where we compare processing times for both techniques. It is evident that setting an error tolerance value makes the estimation process faster as observed in Fig. 5.3. However, this is at the expense of accuracy which is the main aspect of such a task as seen in Fig. 5.2. The bigger the error tolerance value, the faster the estimation of the missing data.

The proposed approach was further used to reconstruct corrupted images, still by way of estimating missing pixels. Its effectiveness was determined using the SE, MSE, RMSLE, MAE, r and the RPA metrics. Also used were the SNR, GD and COD performance measures as defined in Chap. 3. Taking into consideration these evaluation metrics, the novel DL-FA approach is compared against existing approaches in the literature (MLP-PSO, MLP SA and MLP-GA). The results are grouped in Figs. 5.4, 5.5, 5.6 and 5.7, Tables 5.3 and 5.4.

The results reveal that the DL-FA approach outperforms the other approaches. The global deviation is given in Fig. 5.4. It shows a 0.27% GD value for DL-FA

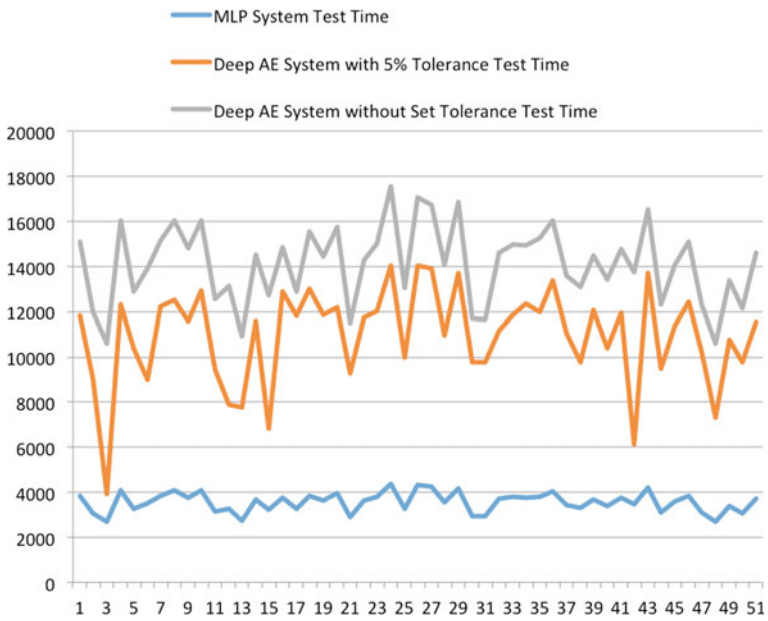

Fig. 5.3 Test time per sample

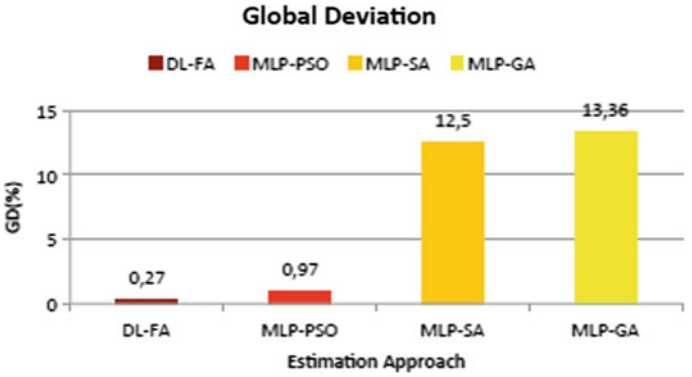

Fig. 5.4 Global deviation versus estimation approach

while MLP-PSO, MLP-SA and MLP-GA obtain 0.97, 12.5 and 13.36% GD values, respectively.

Figures 5.5 and 5.6 show the mean squared error and root mean squared logarithmic error values of the four approaches analysed, including the novel DL-FA approach. They both confirm better performance of the DL-FA approach when compared to the others. DL-FA exhibits a 2.24% of MSE with 11.79% of RMSLE while we obtain 5.83% of MSE for MLP-PSO, and, 30.81 and 33.27% of MSE for MLP-SA

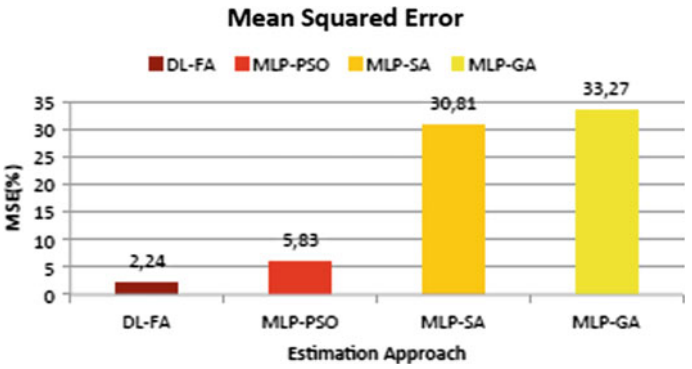

Fig. 5.5 Mean squared error versus estimated approach

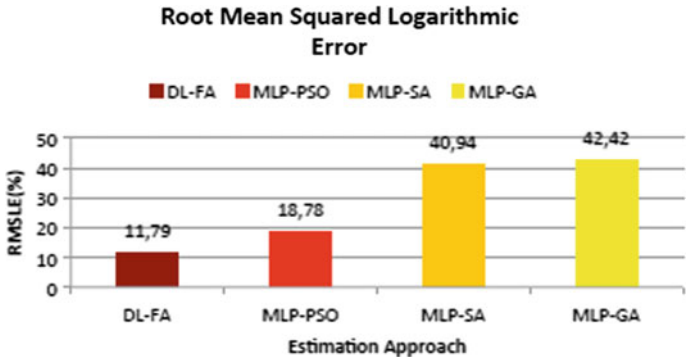

Fig. 5.6 Root mean squared logarithmic error versus estimation approach

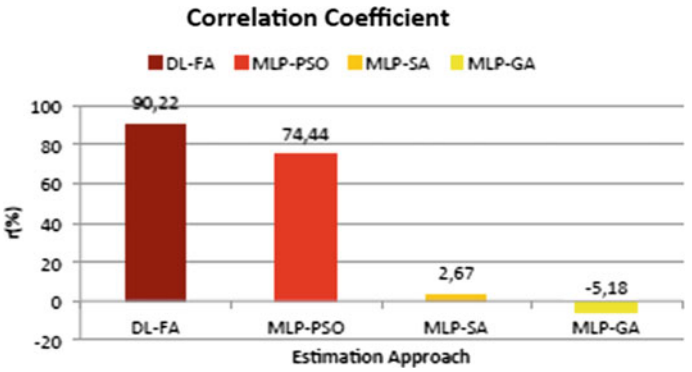

Fig. 5.7 Correlation coefficient versus estimation approach

Table 5.3 DL-FA additional metrics

Method	DL-FA	MLP-PSO	MLP-SA	MLP-GA
COD	**81.41**	55.41	0.07	0.27
MAE	**10.05**	15.83	47.57	49.8
RPA	**56.75**	51.75	10.75	8.25
SE	**14.98**	24.15	55.51	57.68
SNR	**22.42**	61.1	221.36	236.98

The bold values indicate that the algorithm does yield the best (lowest) results

Table 5.4 DL-FA mean squared error objective value per sample

Sample	Dimensions	DL-FA	MLP-PSO	MLP-SA	MLP-GA
1	74	**1.27**	2.44	4.11	4.11
2	74	**2.93**	7.42	8.56	8.56
3	72	**12.34**	17.90	20.69	20.69
4	72	**1.36**	5.58	4.28	4.28
5	73	**2.97**	4.99	12.23	12.23
6	75	**2.86**	5.90	13.84	13.84
7	78	**5.46**	8.76	13.59	13.59
8	78	**3.43**	11.34	11.34	11.34
9	84	**1.74**	6.00	6.72	6.72
10	97	**5.87**	17.65	19.58	19.58

The bold values indicate that the algorithm does yield the best (lowest) results

and MLP-GA, respectively. MLP-PSO depicts 18.78% of RMSLE, while MLP-SA and MLP-GA produce values of 40.94 and 42.42%, respectively.

Considering Fig. 5.7, we observe that the DL-FA approach produces the highest correlation coefficient value of 90.22%, with the second best value of 74.44% obtained by the MLP-PSO method. MLP-SA and MLP-GA yield correlation coefficient values of 2.67 and −5.18%, respectively.

These findings are further backed by the values in Table 5.3 with the DL-FA system yielding the best COD, MAE, RPA, SE and SNR values.

Taking into account Table 5.4, it is observed that the proposed DL-FA approach results in the best objective function value per record during the estimation of all missing values within that record.

In Table 5.5, we present results obtained from statistically analysing the estimates obtained by the DL-FA approach when compared against the MLP-PSO, MLP-SA and MLP-GA approaches using the t-test. The t-test null hypothesis (H_0) assumes that there is no significant difference in the means of the missing data estimates obtained by the DL-FA, MLP-PSO, MLP-SA and MLP-GA methods. The alternative hypothesis (H_A), however, indicates that there is a significant difference in the means of the missing data estimates obtained by the four methods.

Table 5.5 Statistical analysis of DL-FA results

Pairs compared	P-values (95% confidence level)
DL-FA:MLP-PSO	4.09 × 10e−05
DL-FA:MLP-SA	2.0 × 10e−132
DL-FA:MLP-GA	1.0 × 10e−140

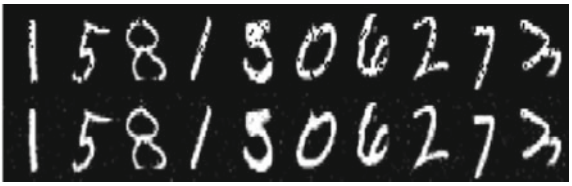

Fig. 5.8 Top row: corrupted images—Bottom row: DL-FA reconstructed images

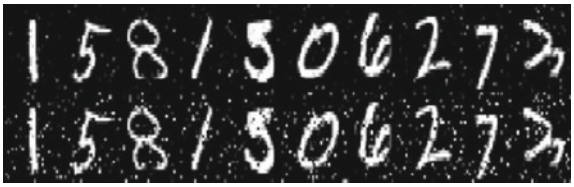

Fig. 5.9 Top row: MLP-PSO reconstructed images—Bottom row: MLP-GA reconstructed images.

Table 5.5 reveals that there is a significant difference at a 95% confidence level in the means of the estimates obtained by DL-FA when compared to MLP-PSO, MLP-SA and MLP-GA, yielding p-values of 4.09 × 10e−05, 2.0 × 10e−132 and 1.0 × 10e−140, respectively, when all three pairs are compared. This, therefore, indicates that the null hypothesis (H_0), which assumes that there is no significant difference in the means between DL-FA and the other three methods can be rejected in favour of the alternative hypothesis (H_A) at a 95% confidence level.

In the top row of Fig. 5.8, we depict 10 images with missing pixel values which are to be estimated prior to classification tasks being performed by statistical methods. In the bottom row of the same figure, we show the reconstructed images from using the DL-FA approach, while in the top and bottom rows of Fig. 5.9, we observe the reconstructed images when the MLP-PSO and MLP-GA approaches are used, respectively. It can be seen that the reconstructed images using MLP-PSO and MLP-GA introduce a large amount of noise, more so in the bottom row than in the top row, as opposed to when the DL-FA approach is applied. Furthermore, closer inspection reveals that the images are not fully reconstructed as not all pixel values within the images are estimated correctly.

5.5 Conclusions

This chapter investigates the use of a deep neural network with a swarm intelligence algorithm to impute missing data in a high-dimensional dataset. According to the arbitrary missing data pattern and MCAR mechanism, missing data could occur anywhere in the dataset. The experiment in this chapter considers a scenario in which 10% of the test set of data is missing. These values are to be estimated with a set error tolerance of 10%. It is observed from analysing the outputs from the system that some of the estimated values do fall within this tolerance value, but others do not. Also, the 1.29% error value obtained on average could be deemed misleading as individual comparisons between actual and imputed values in some cases resulted in high error values as observed in the first part of the results section. It can be assumed that the reason for these discrepancies could be as a result of inadequate parameters being used in the estimation procedure or some of the known values being re-approximated during the estimation phase. This, therefore, alters the learned interrelationships and correlations between the dataset variables stored in the network weights. Based on the findings in this chapter, it will be worth it performing in-depth parameter analysis in any future research in order to observe which parameters are optimal for the task and subsequently generalize this aspect using several datasets. Another obstacle faced in this research was the computation time to estimate the missing values and to address this, it will be a good idea to look into parallelizing the process on a multi-core system to observe whether parallelizing the task does indeed speed up the process and maintain efficiency and accuracy.

References

Abdella, M., & Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. In *3rd International Conference on Computational Cybernetics, ICCC 2005* (pp. 207–212). IEEE.

Allison, P. D. (1999). Multiple imputation for missing data: A cautionary tale. Philadelphia.

Aydilek, I. B., & Arslan, A. (2012). A novel hybrid approach to estimating missing values in databases using *k*-nearest neighbors and neural networks. *International Journal of Innovative Computing, Information and Control, 7*(8), 4705–4717.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1798–1828. IEEE.

Deng, L., et al. (2013). Recent advances in deep learning for speech research at Microsoft. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 8604–8608). IEEE.

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing, 7*(3–4), 197–387. Now Publishers Inc.

Fischer, A., & Igel, C. (2012). An introduction to restricted Boltzmann machines. In *17th Iberoamerican Congress, CIARP*, *Proceedings Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (pp. 14–36). Heidelberg: Springer. ISBN: 978–3-642-33275-3.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *American Association for the Advancement of Science, 313*(5786), 504–507.

Isaacs, J. C. (2014). Representational learning for sonar ATR. In *Proceedings SPIE 9072, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XIX* (p. 907203). http://dx.doi.org/10.1117/12.2053057.

Koko, E. E. M., & Mohamed, A. I. A. (2015). Missing data treatment method on cluster analysis. *International Journal of Advanced Statistics and Probability, 3*(2), 191–209.

LeCun, Y. (2016). *The MNIST database of handwritten digits*. http://yann.lecun.com/exdb/mnist/. Accessed 1 Jan 2016.

Leke, C., Twala, B., & Marwala, T. (2014). Modeling of missing data prediction: Computational intelligence and optimization algorithms. In *International Conference on Systems, Man and Cybernetics (SMC)* (pp. 1400–1404). IEEE.

Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference in Swarm Intelligence*. Springer International Publishing, pp. 259–270.

Little, R. J., & Rubin, D. B. (2014). *Statistical analysis with missing data*. New York: Wiley.

Mistry, F. J., Nelwamondo, F. V., & Marwala, T. (2009). Missing data estimation using principle component analysis and autoassociative neural networks. *Journal of Systemics, Cybernatics and Informatics, 7*(3), 72–79.

Nelwamondo, F. V., Mohamed, S., & Marwala, T. (2007). Missing data: A comparison of neural network and expectation maximisation techniques. ArXiv preprint arXiv:0704.3474.

Rana, S., John, A. H., Midi, H., & Imon, A. (2015). Robust regression imputation for missing data in the presence of outliers. *Far East Journal of Mathematical Sciences*, *97*(2), 183. Pushpa Publishing House.

Rubin, D. B. (1978). Multiple imputations in sample surveys-a phenomenological Bayesian approach to nonresponse. In *Proceedings of the Survey Research Methods Section of the American Statistical* (vol. 1, pp. 20–34). Association. American Statistical Association.

Yang, X.-S. (2010). Firefly algorithm, Levy flights and global optimization. In M. Bramer, R. Ellis, & M. Petridis (Eds.), *Research and development in intelligent systems XXVI* (pp. 209–218). London: Springer.

Zhang, S., Jin, Z., & Zhu, X. (2011). Missing data imputation by utilizing information within incomplete instances. *Journal of Systems and Software*, *84*(3), 452–459. Elsevier.

Zhang, S. (2011). Shell-neighbor method and its application in missing data imputation. *Applied Intelligence*, *35*(1), 123–133. Springer.

Chapter 6
Missing Data Estimation Using Ant Colony Optimization Algorithm

6.1 Introduction

The presence of missing data affects the quality of the dataset, which impacts the analysis and interpretation of the data. There are several reasons that could lead to data being missing in a dataset with some being more predominant than others. The first well-known reason is that participants deny revealing some personal and sensitive information, for example, monthly income or monthly expenditure. The second main reason is the failure of systems that are meant to capture and store the data in databases. Another main reason is interoperability whereby information exchanged between systems may be subjected to missing data.

Missing data remains an ever-present issue in the real world as well as within the academic community. Making decisions which entail processes that rely on accurate knowledge relies predominantly upon the availability of data, from which information can be extracted and these decisions made. Such processes often require predictive models or other computational intelligence techniques that use the observed data as inputs. However, in some cases, data could be lost, corrupted or recorded incompletely, which affects the quality of the data negatively. Majority of the decision-making and machine learning frameworks such as artificial neural networks, support vector machines and principal component analysis cannot be used for decision-making and data analysis if the data is incomplete because these missing values can critically influence pattern recognition and classification tasks. Since the decision output should still be maintained despite the missing data, it is important to deal with the problem. Therefore, in case of incomplete or missing data, the initial step in processing the data is estimating the missing values.

The applications of missing data estimation methods are numerous and include, but are not limited to, structural equation modelling (Carter 2006), air quality data (Zainuri et al. 2015) and reconstruction of times series data (Sidekerskiene and Damasevicius 2016) observations. With that said, the existing methods depend on the nature of the data, the pattern of missingness and are predominantly implemented

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_6

on low-dimensional datasets. Abdella and Marwala (2005) implemented a joint neural network-genetic algorithm framework to impute missing values in one dimension while Rana et al. (2015) used robust regression imputation in datasets with outliers and investigated its performance. In Zhang et al. (2011), it was recommended that knowledge in relevant instances be used when estimating missing data. This led to the non-parametric iterative imputation algorithm (NIIA) being introduced. In papers such as Jerez et al. (2010), Leke et al. (2014), Liew et al. (2011), Myers (2011), and Van Buuren (2012), new techniques to impute missing data and comparisons between these and existing methods are observed. The techniques mainly cater to low-dimensional datasets with missing values but are less effective when high-dimensional datasets are considered with missingness occurring in an uncontrolled manner as will be depicted subsequently in the chapter. The main motivation behind this work is therefore to introduce high-dimensional missing data estimation approaches, with emphasis laid on image recognition datasets. These approaches will be used to reconstruct corrupted images by estimating image pixel values.

To investigate new techniques to address the previously mentioned dimensionality drawback, we implement a deep autoencoder, with an autoencoder network being an unsupervised learning technique that endeavours to reconstruct the input data as the output by learning an approximation function which decreases the contrast between the original data, x, and the output reconstructed data, $\tilde{x}$, the same as with principal component analysis (PCA). A deep autoencoder is made up of two symmetrical deep belief networks with a couple of shallow layers representing the encoding section of the network, while the second set of shallow layers represents the decoding section, with each of the individual layers being restricted Boltzmann machines (RBMs). These networks are applicable in a variety of sectors, for example, deep autoencoders were used for visuomotor learning in Finn et al. (2016) while in Ju et al. (2015), they were used with support vector machines (SVMs) to learn sparse features and perform classification tasks. Feng et al. (2014) used a deep denoising autoencoder to generate robust speech features for a noisy reverberant speech recognition system, and in Krizhevsky and Hinton (2011), these networks were used to map small colour images to short binary codes. Autoencoders have several advantages such as they are more flexible by introducing non-linearity in the encoding part of the network contrary to the likes of PCA, which is a key property of the pretraining procedure, they require no prior knowledge of the data properties and correlations, and they are intuitive. The main drawback of this network is its need for a significant number of samples for training and learning, which are not always readily available.

In this chapter, we propose a new method for missing data estimation in high-dimensional datasets, namely, DL-ACO, based on the advantages of deep autoencoders and the ant colony optimiz ation algorithm. For this purpose, we begin by giving background information on the optimization algorithm, being the ACO algorithm. Subsequently, we present a description of the experimental model used. Next, we present analyses of the performance of the proposed model and compare the results obtained to those of existing methods. Furthermore, we report on the results and discuss the findings from our experiments. Finally, brief conclusions are presented.

6.2 Ant Colony Optimization (ACO)

ACO is an algorithm that mimics the innate behaviour of ants. In the everyday lives of ants, the ants have to explore the neighbourhood of their nests in the search for food for sustenance purposes (Monteiro et al. 2012). When ants move, they leave a substance termed pheromone on their trail. There are two main objectives behind this deposit of pheromones. One reason is to allow ants to navigate their way back to the nests, and the second is that it allows other ants to trace back the direction used by other ants, so it can be followed (Monteiro et al. 2012). ACO has a collection of characteristic traits that can be regarded as building blocks. These traits are essential and are required to be specified in every implementation. These characteristics include (Dorigo et al. 1991):

i. The method selected to build the solutions,
ii. The examining knowledge,
iii. The rule to update the pheromones,
iv. The probability function and transition rules,
v. The values of the parameters, and,
vi. The stopping criteria (Monteiro et al. 2012).

The algorithm considers a unique colony of ants that has m artificial ants collaborating with one another. Prior to the start of the execution of the algorithm, each of the links between the solutions is given a certain number of pheromones, τ_0. This value is usually very small, small enough that the probability of the path to each solution being chosen is not zero. At each iteration, each of the m ants that has constructed a solution in it updates the values of the pheromone. The pheromone, τ_{ij}, that is related to the link between solutions i and j is revised using the following equation (Dorigo et al. 1991, 1996):

$$\tau_{ij} \leftarrow (1-\rho)^{*}\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{6.1}$$

where ρ represents the evaporation rate of the pheromone, m depicts the number of ants, and $\Delta\tau_{ij}^{k}$ is the number of pheromones deposited in the link between solutions i and j, which is updated by ant k such that

$$\Delta\tau_{ij}^{k} = \frac{Q}{L_k}, \tag{6.2}$$

if ant k used the link between solutions i and j, while $\Delta\tau_{ij}^{k} = 0$ otherwise (Dorigo et al. 1996). Q is a constant, with L_k representing the radius of the link created by ant k. In the construction of a new solution, ants choose the next solution via a stochastic approach. When ant k is at solution i and has constructed a partial solution, a^{P}, the probability of then moving to solution j is such that (Dorigo and Di Caro 1999)

Table 6.1 ACO parameters

Parameters	Value
Maximum Number of Iterations	1000
Population Size	10
Intensification Factor	0.5
Deviation Distance Ratio	1
Sample Size	40

$$p_{ij}^{k} = \frac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} * \eta_{il}^{\beta}} \, if \; c_{ij} \in N\left(s^p\right),$$

or

$$p_{ij}^{k} = 0 if \, c_{ij} \notin N\left(s^p\right). \tag{6.3}$$

$N(s^p)$ represents a collection of appropriate items that are links between solutions i and l, whereby l is a solution that has not yet been tested for its fitness towards the task at hand by ant k. The α and β parameters govern the related relevance of the heuristic information versus pheromone, η_{ij}, obtained by (Dorigo et al. 1991, 2006)

$$\eta_{ij} = \frac{1}{d_{ij}}, \tag{6.4}$$

where *dij* is the distance between solutions i and j. This algorithm has been applied in several papers such as in Zecchina et al. (2006), in which it was used to solve problems in water distribution systems, while in Liu et al. (2013), the ACO algorithm was employed to solve a mathematical model which was constructed to represent a process planning problem. In Nkaya et al. (2015), the ACO algorithm was used in problems where no a priori information was considered in spatial clustering problems and compared against a novel algorithm which was proposed.

The ACO parameters used in the model implementation are given in Table 6.1 except for the number of decision variables which depend on the number of missing values in a record. The data was normalized to being in the range [0, 1], meaning the lower and upper bounds of the decision variables were 0 and 1, respectively. These parameters were selected because they led to the more optimal outcomes. This conclusion was arrived at after different combinations and permutations of values were tested.

6.3 Experimental Model

In this section, we present information on the proposed approach which uses a combination of a deep learning regression model, a deep autoencoder network, and an

optimization technique, the ACO algorithm, to approximate the missing data. Two predominant features of an autoencoder are: (i) its auto-associative nature and (ii) the butterfly-like structure of the network resulting from a bottleneck trait in the hidden layers, which were the reasons behind the network being used. Autoencoders are also ideal courtesy of their ability to replicate the input data by learning certain linear and non-linear correlations and covariances present in the input space, by projecting the input data into lower dimensions. The only condition required is that the hidden layer(s) have fewer nodes than the input layer, though it is dependent on the application. Prior to optimizing the regression model parameters, it is necessary to identify the network structure where the structure depends on the number of layers, the number of hidden units per hidden layer, the activation functions used and the number of input and output variables. After this, the parameters can be approximated using the training set of data. The parameter approximation procedure is done for a given number of training cycles, with the optimal number of this being obtained by analysing the validation error. The aim of this was to avoid overtraining the network and to use the fastest training approach without compromising on accuracy. The optimal number of training cycles was found to be 500. The training procedure estimates weight parameters such that the network output is as close as possible to the target output.

The ACO algorithm is used to estimate the missing values by optimizing an objective function which has the trained network as a part of it. It will use values from the population as part of the input to the network, and the network will recall these values and they will form part of the output. The complete data matrix containing the estimated values and observed values will be fed into the autoencoder as input. Some inputs are known, with others unknown and they will be estimated using the regression method and the ACO algorithm as described at the beginning of the paragraph. The symbols I_k and I_u as used in Fig. 6.1 represent the known and unknown/missing values, respectively.

Considering that the approach makes use of a deep autoencoder, it is imperative that the autoencoder architecture matches the output to the input. This trait is expected when a dataset with familiar correlations recorded in the network is used. The error used is the disparity between the target output and the network output, which is expressed as

$$\delta = \vec{I} - f\left(\vec{W}, \vec{I}\right), \tag{6.5}$$

where $\vec{I}$ and $\vec{W}$ represent the inputs and the weights, respectively.

The square of Eq. (6.5) is used to always guarantee that the error is positive. This results in the following equation:

$$\delta = \left(\vec{I} - f\left(\vec{W}, \vec{I}\right)\right)^2. \tag{6.6}$$

Courtesy of the fact that the input and output vectors contain both I_k and I_u, the error function is rewritten as

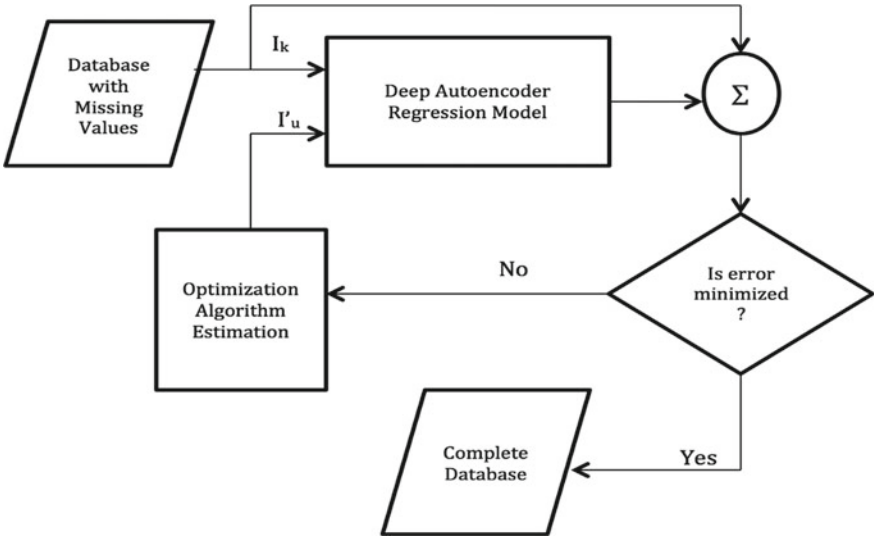

Fig. 6.1 Deep autoencoder and ant colony optimization missing data estimator structure. © 2017 Springer. Reproduced with permission from Leke et al. (2017)

$$\delta = \left(\begin{bmatrix} I_k \\ I_u \end{bmatrix} - f\left(\begin{Bmatrix} I_k \\ I_u \end{Bmatrix}, w \right) \right)^2. \tag{6.7}$$

Equation (6.7) is the objective function used and minimized by the ACO algorithm to estimate I_u, with f being the regression model function. From the above descriptions of how the deep autoencoder and the ACO algorithm operate, the equation below summarizes the function of the proposed approach, with f_{ACO} being the ACO algorithm estimation operation and f_{DAE} being the function of the deep autoencoder.

$$y = f_{\text{DAE}}\left(W, f_{\text{ACO}}\left(\vec{I}\right)\right), \tag{6.8}$$

where $\vec{I} = \begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}$ represents the input space of known and unknown features. This equation represents the model design whereby the complete input vector with known and estimated missing data entries obtained by executing the missing data estimation procedure (f_{ACO}) is presented to the deep regression model (f_{DAE}) to observe whether the network error has been minimized. If it is found that the error has been minimized, the output, y, will contain the known input vector values and the optimal missing data estimated values.

6.4 Experimental Results

Considering the evaluation metrics presented in Sect. 4.4, the performance of the DL-ACO method is evaluated and compared against the existing methods (MLP-GA, MLP-SA and MLP-PSO) by estimating the missing attributes concurrently, wherever missing data may be ascertained. The scenarios investigated were such that any sample/record could have at least 62, and at most 97 missing attributes (dimensions) are to be approximated. The MLP network used has a structure of 784-400-784, with 784 input and output nodes in the input and output layers, respectively, and, 400 nodes in the one hidden layer. This number is obtained by testing the network with different number of nodes in the hidden layer and observing the network structure that produces the lowest possible network error.

Figures 6.2, 6.3, 6.4 and 6.5 show the performance and comparison between the DL-ACO, MLP-PSO, MLP-SA and MLP-GA approaches. Figures 6.2 and 6.3 are bar charts that show the MSE and RMSLE values for DL-ACO when compared to MLP-PSO, MLP-SA and MLP-GA.

We observe 0.66, 5.19, 31.02 and 30.98% of MSE and 6.06, 17.98, 41.21 and 41.25% of RMSLE for DL-ACO, MLP-PSO, MLP-SA and MLP-GA, respectively. DL-ACO yielded the lowest MSE value when compared to the other approaches. These results are validated by the correlation coefficient whose bar chart is given in Fig. 6.4.

DL-ACO and MLP-PSO yielded 96.29 and 74.52% correlation values, respectively, while MLP-SA and MLP-GA showed correlations of −1.88 and −0.56%, respectively.

The MLP-SA and MLP-GA yielded 13.25 and 13.67% of global deviation, respectively, while DL-ACO and MLP-PSO, respectively, yielded 0.0078 and 0.98%, as shown in Fig. 6.5. As observed, the DL-ACO approach obtained the best values for all four metrics presented diagrammatically.

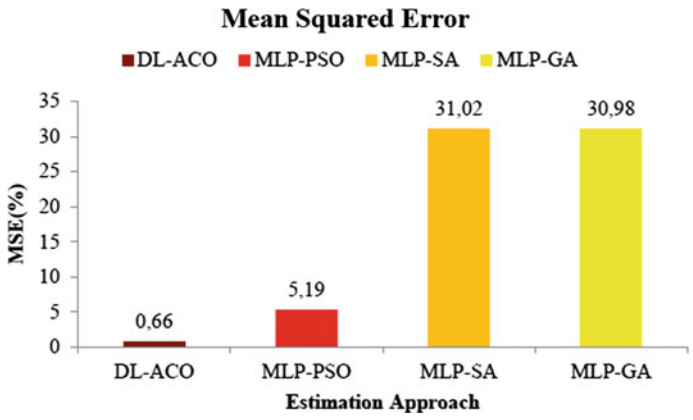

Fig. 6.2 Mean squared error versus estimation approach

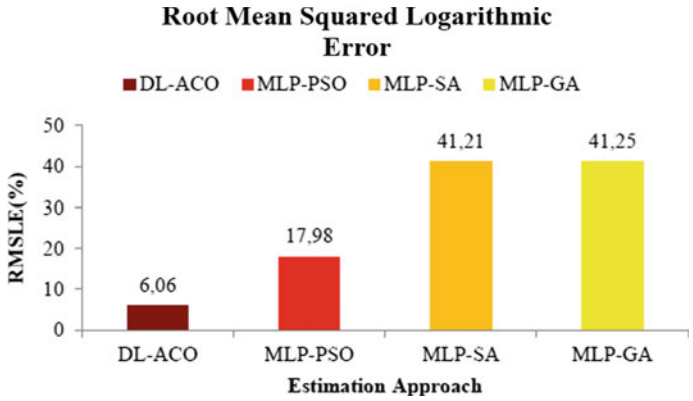

Fig. 6.3 Root mean squared logarithmic error versus estimation approach

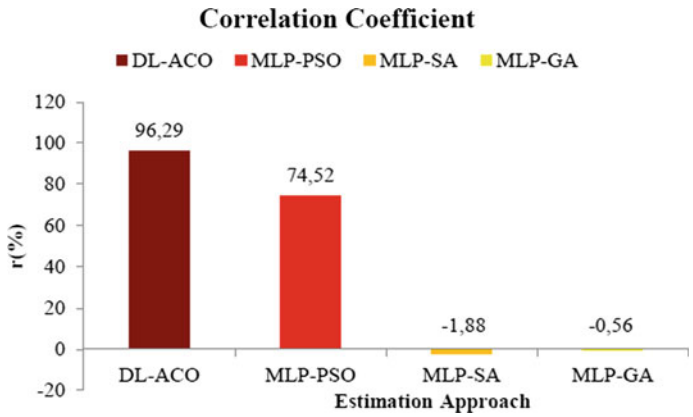

Fig. 6.4 Correlation coefficient versus estimation approach

In Table 6.2, the dimensions column refers to the number of missing values in a sample/record. Tables 6.2 and 6.3 further back the findings from Figs. 6.4, 6.5, 6.6 and 6.7 showing that the proposed DL-ACO approach yielded the lowest objective function value in the estimation of missing values in each sample, as well as the best COD, MAE, SE and SNR values. Considering the RPA metric, the MLP-PSO approach yielded a better value than the proposed approach.

In Table 6.4, we present results obtained from statistically analysing the estimates obtained by the DL-ACO approach when compared against the MLP-PSO, MLP-SA and MLP-GA approaches using the t-test. The t-test null hypothesis (H_0) assumes that there is no significant difference in the means of the missing data estimates obtained by the DL-ACO, MLP-PSO, MLP-SA and MLP-GA methods. The alternative hypothesis (H_A), however, indicates that there is a significant difference in the means of the missing data estimates obtained by the four methods.

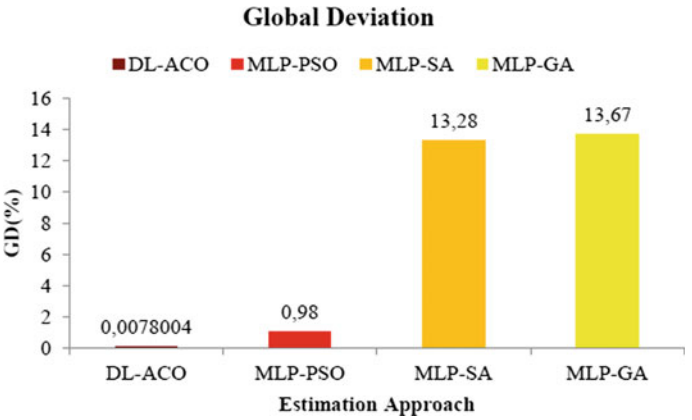

Fig. 6.5 Global deviation versus estimation approach

Table 6.2 DL-ACO mean squared error objective value per sample

Sample	Dimensions	DL-ACO	MLP-PSO	MLP-SA	MLP-GA
1	79	**2.52**	13.62	15.59	15.59
2	88	**2.38**	7.34	8.78	8.78
3	75	**1.27**	5.59	6.76	6.76
4	81	**0.26**	3.69	5.91	5.91
5	83	**0.46**	6.57	8.02	8.02
6	82	**1.27**	4.81	9.76	9.76
7	90	**1.91**	5.66	15.05	15.05
8	79	**1.18**	7.58	9.54	9.54
9	76	**2.59**	7.96	9.48	9.48
10	76	**2.86**	6.52	12.60	12.60

The bold values indicate that the algorithm does yield the best (lowest) results

Table 6.3 DL-ACO additional metrics

Method	DL-ACO	MLP-PSO	MLP-SA	MLP-GA
COD	**92.71**	55.53	0.0353	0.0032
MAE	**3.37**	14.82	47.5	47.7
RPA	53.25	**53.58**	10.75	10.08
SE	**8.12**	22.78	55.7	55.66
SNR	**7.7**	57.16	209.04	208.83

The bold values indicate that the algorithm does yield the best (lowest) results

Table 6.4 reveals that there is a significant difference at a 95% confidence level in the means of the estimates obtained by the DL-ACO when compared to the MLP-PSO, MLP-SA and MLP-GA, yielding p-values of 5.31×10^{-15}, 2×10^{-167} and 3×10^{-173}, respectively, when all three pairs are compared. This, therefore,

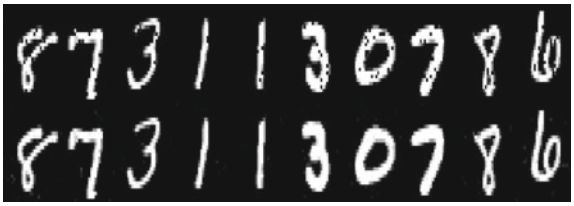

Fig. 6.6 Top row: Corrupted images—Bottom row: DL-ACO reconstructed images

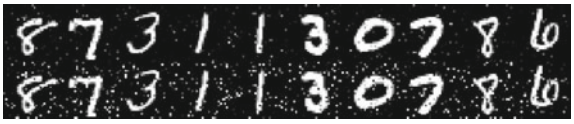

Fig. 6.7 Top row: MLP-PSO reconstructed images—Bottom row: MLP-GA reconstructed images

Table 6.4 Statistical analysis of DL-ACO results

Pairs compared	p-Values (95% Confidence Level)
DL-ACO:MLP-PSO	5.31×10^{-15}
DL-ACO:MLP-SA	2×10^{-167}
DL-ACO:MLP-GA	3×10^{-173}

indicates that the null hypothesis (H_0), which assumes that there is no significant difference in the means between DL-ACO estimates and those of the other three methods can be rejected in favour of the alternative hypothesis (H_A) at a 95% confidence level.

In the top row of Fig. 6.6, we depict 10 images with missing pixel values which are to be estimated prior to classification tasks being performed by statistical methods. In the bottom row of the same figure, we show the reconstructed images from using the DL-ACO approach, while in the top and bottom rows of Fig. 6.7, we observe the reconstructed images when the MLP-PSO and MLP-GA approaches are used, respectively. It can be seen that the reconstructed images using the MLP-PSO and the MLP-GA introduce much noise, more so in the bottom row than in the top row compared to when the DL-ACO approach is applied. Furthermore, closer inspection reveals that the images are not fully reconstructed as not all pixel values within the images are estimated correctly.

6.5 Conclusion

This chapter presented an investigation of the estimation of missing data via a novel approach. The estimation method comprises of a deep autoencoder network to replicate the input data, in combination with the cuckoo search algorithm to estimate the

missing data. The performance of the model is investigated and compared against existing methods including an MLP autoencoder with genetic algorithm, simulated annealing and particle swarm optimization. The results obtained reveal that the proposed system yields more accurate estimates when compared against the MLP hybrid systems. This is made evident when the mean squared error, root mean squared logarithmic error, correlation coefficient, relative prediction accuracies, signal-to-noise ratio and global deviation are considered, with the proposed approach yielding the best values of these. Also, when the objective function value is considered during the estimation process, it is observed that the proposed approach results in the lowest values for this for each instance.

References

Abdella, M., & Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. In *3rd International Conference on Computational Cybernetics, ICCC* (pp. 207–212). IEEE.

Carter, R. L. (2006). Solutions for missing data in structural equation modeling. *Research & Practice in Assessment, 1*(1), 1–6.

Dorigo, M., & Di Caro, G. (1999). New ideas in optimization. In *The Ant Colony Optimization Meta-heuristic* (pp. 11–32). Maidenhead, UK: McGraw-Hill Ltd. Retrieved May 20, 2016, from http://dl.acm.org/citation.cfm?id=329055.329062.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). *Positive feedback as a search strategy*. Technical Report.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26*(1), 29–41.

Dorigo, M., Birattari, M., & Sttzle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine, 1,* 28–39.

Feng, X., Zhang, Y., & Glass, J. R. (2014). Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *International Conference on Acoustic, Speech and Signal Processing (ICASSP)* (pp. 1759–1763). IEEE.

Jerez, J. M., Molina, I., Garc´ıa-Laencina, P. J., Alba, E., Ribelles, N., Martín, M., & Franco, L. (2010). Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine, 50*(2), 105–115. Elsevier.

Ju, Y., Guo, J., & Liu, S. (2015). A deep learning method combined sparse autoencoder with SVM. In *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 257–260). IEEE.

Krizhevsky, A., & Hinton, G. E. (2011). Using very deep autoencoders for content based image retrieval. In *19th European Symposium on Artificial Neural Networks (ESANN)*, 27–29 April 2011. Bruges, Belgium.

Leke, C., Twala, B., & Marwala, T. (2014). Modeling of missing data prediction: Computational intelligence and optimization algorithms. In *International Conference on Systems, Man and Cybernetics (SMC)* (pp. 1400–1404). IEEE.

Leke, C., Ndjiongue, A. R., Twala, B., & Marwala, T. (2017). A deep learning-cuckoo search method for missing data estimation in high-dimensional datasets. In *International Conference in Swarm Intelligence* (pp. 561–572). Springer, Cham.

Liew, A. W. -C., Law, N. -F., & Yan, H. (2011). Missing value imputation for gene expression data: Computational techniques to recover missing data from available information. *Briefings in Bioinformatics, 12*(5), 498–513. Oxford University Press.

Liu, X. J., Yi, H., & Ni, Z.-H. (2013). Application of ant colony optimization algorithm in process planning optimization. *Journal of Intelligent Manufacturing, 24*(1), 1–13.

Monteiro, M. S. R., Fontes, D. B. M. M., & Fontes, F. A. C. C. (2012). Ant colony optimization: A literature survey. Universidade do Porto, Faculdade de Economia do Porto, FEP Working Papers 474. Retrieved January 2016, from http://EconPapers.repec.org/RePEc:por:fepwps:474.

Myers, T. A. (2011). Goodbye, listwise deletion: Presenting hot deck imputation as an easy and effective tool for handling missing data. *Communication Methods and Measures, 5*(4), 297–310. Taylor & Francis.

Nkaya, T., Kayalgil, S., & Zdemirel, N. E. (2015). Ant colony optimization-based clustering methodology. *Applied Soft Computing, 28,* 301–311.

Rana, S., John, A. H., Midi, H., & Imon, A. (2015). Robust regression imputation for missing data in the presence of outliers. *Far East Journal of Mathematical Sciences. Pushpa Publishing House, 97*(2), 183–195.

Sidekerskiene, T., & Damasevicius, R. (2016). Reconstruction of missing data in synthetic time series using EMD. In *CEUR Workshop Proceedings* (Vol. 1712, pp. 7–12).

Finn C., Tan, X., Duan, Y., Darrell, T., Levine, S., & Abbeel, P. (2016). Deep Spatial Autoencoders for Visuomotor Learning. In *International Conference on Robotics and Automation (ICRA)* (pp. 512–519). IEEE.

Van Buuren, S. (2012). *Flexible imputation of missing data*. CRC press.

Zainuri, N. A., Jemain, A. A., & Muda, N. (2015). A comparison of various imputation methods for missing values in air quality data. *Sains Malaysiana, 44*(3), 449–456.

Zecchina, A. C., Simpsona, A. R., Maiera, H. R., Leonarda, M., Roberts, A. J., & Berrisforda, M. J. (2006). Application of two ant colony optimisation algorithms to water distribution system optimisation. *Mathematical and Computer Modelling, 44*(5–6), 451–468.

Zhang, S., Jin, Z., & Zhu, X. (2011). Missing data imputation by utilizing information within incomplete instances. *Journal of Systems and Software. 84*(3), 452–459. Elsevier

Chapter 7
Missing Data Estimation Using Ant-Lion Optimizer Algorithm

7.1 Introduction

The presence of missing data in datasets as per previous research in a variety of academic domains alludes to the fact that data analysis tasks and decision-making processes are rendered non-trivial. Due to this observation, one can assume that reliable and accurate decisions are more likely to be made when complete records are used as opposed to incomplete datasets. The result from this presumption has been more research being conducted in the data mining domain with the introduction of novel methods that accurately perform the task of filling in missing data. Research shows that processes in different professional sectors that use sensors in instruments to report very important information, which are subsequently used to make decisions, for example in medicine, manufacturing and energy, may come across instances whereby these sensors fail, leading to there being missing data entries in the dataset, thereby influencing the nature of the decisions made. In scenarios such as these, it is of great importance that there be a system that can impute the missing data from these faulty sensors with high accuracy, the missing data from these faulty sensors. This imputation framework will need to take into consideration the existing correlations between the information obtained from the sensors in the system in order to accurately estimate the missing data. Another example of the missing data problem is in image recognition tasks, whereby missing pixel values renders the task of predicting or classifying an image difficult. As such, it is paramount that there be a system capable of estimating these missing pixel values with high accuracy in order to make these tasks easier and more feasible.

Datasets nowadays such as those that record production, manufacturing and medical data may suffer from the problem of missing data at different phases of the data collection and storage processes. Faults in measuring instruments or data transmission lines are predominant causes of missing data. The occurrence of missing data results in difficulties in decision-making and analysis tasks which rely on access to complete and accurate data. This results in data estimation techniques which are

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_7

not only accurate but also efficient. Several methods exist as a means to alleviate the problems presented by missing data ranging from deleting records with missing attributes (listwise and pairwise data deletion) to approaches that employ statistical and artificial intelligence methods such as hybrid neural network and evolutionary algorithm approaches. The problem though is some of the statistical and naive approaches more often than not produce biased approximations, or they make false assumptions about the data and correlations within the data. These have adverse effects on the decision-making processes which are data dependent.

Furthermore, missing data has always been a challenge in the real world as well as within the research community. Decision-making processes that rely on accurate knowledge are quite reliant upon the availability of data from which information can be extracted. Such processes often require predictive models or other computational intelligence techniques that use the observed data as inputs. However, in some cases due to alternate reasons, data could be lost, corrupted or recorded incompletely, which affects the quality of the data negatively. The majority of the decision-making and machine learning frameworks such as artificial neural networks (ANNs), support vector machines (SVMs), principal component analysis (PCA) and others cannot be used for decision-making and data analysis if the data is incomplete. Missing values can critically influence pattern recognition and classification tasks. Since the decision output should still be maintained despite the missing data, it is important to deal with the problem. Therefore, in case of incomplete or missing data, the initial step in processing the data is estimating the missing values.

7.2 Rationale

The applications of missing data estimation techniques are vast and with that said, the existing methods depend on the nature of the data, the pattern of missingness and are predominantly implemented on low-dimensional datasets. These methods include, but are not limited to, structural equation modelling (Carter 2006), environmental (Zainuri et al. 2015) and time series (Sidekerskiene and Damasevicius 2016) observations. Marivate et al. (2007) used the MLP autoencoder networks, principal component analysis and support vector machines in combination with the genetic algorithm to impute missing data, and in Rana et al. (2015), they used robust regression imputation in datasets with outliers and investigated its performance. Missing data imputation via a multi-objective genetic algorithm technique is presented in Lobato et al. (2015). The results obtained point to the fact that the approach proposed outperforms certain popular missing data imputation techniques yielding accuracy values in the 90 percentiles.

Abdella and Marwala (2005) implemented a hybrid system comprising of the genetic algorithm and a neural network with the aim of imputing missing data values within a single feature variable at a time, in scenarios whereby the number of missing values varied within this variable. Aydilek and Arslan (2012) proposed a novel system that hybridizes the k-nearest neighbour with a neural network with

the aim of imputing missing values within a single feature variable. In Leke et al. (2014), hybrid systems made up of an auto-associative neural network (AANN) and the particle swarm optimization (PSO), simulated annealing (SA) and genetic algorithm (GA) optimization techniques were created and applied to estimate missing values, yielding high accuracy values in scenarios where a single feature variable was affected by the problem of missing data. Some researchers used neural networks with principal component analysis (PCA) and GA to solve the missing data problem (Mistry et al. 2009; Nelwamondo et al. 2007). Information within records with missing values was suggested to be used in the missing data estimation task in (Zhang et al. 2011). This resulted in a non-parametric iterative imputation algorithm (NIIA) being introduced which yielded a classification accuracy of at most 87.3% on the imputation of discrete values, and a root mean squared error value of at least 0.5 on the imputation of continuous values when the missing data ratios were varied. In Zhang (2011), a shell-neighbour imputation (SNI) approach is proposed and applied to the missing data imputation problem, and it makes use of the shell-neighbour method. The results obtained indicated that the proposed method performed better than the k-nearest neighbour imputation when imputation and classification accuracy were considered. This was because of the method considering the right and left nearest neighbours of the missing data in addition to using different numbers of nearest neighbours as opposed to the k-nearest neighbour method which uses a fixed number of k-nearest neighbours. New techniques aimed at solving the missing data problem and comparisons between these and existing methods can be found in the literature (Baraldi and Enders 2010; Van Buuren 2012; Jerez et al. 2010).

7.3 Ant-Lion Optimizer (ALO)

The ALO is a meta-heuristic algorithm that mimics the interaction between ants, prey and the ant-lion species (Mirjalili 2015). The ALO implements five main steps of hunting, these being

1. The random motion of ants,
2. The construction of traps by the ant-lions,
3. The capturing of ants in the traps,
4. The catching of prey, and,
5. The rebuilding of traps.

Also, it is a gradient-free algorithm which has the property of providing greater exploration and exploitation of the solution space. Exploration is assured by the selection of ant-lions at random, as well as the motion of ants around them which is also random. Exploitation, on the other hand, is assured by the flexible declining of the boundaries of ant-lion traps. The algorithm is based on three tuples being ALO (A_1, A_2, A_3), which estimate the global optimum of an optimization problem. These three tuples are defined respectively as (Mirjalili 2015; Gupta and Saxena 2016)

$$\Phi \rightarrow_{A_1} \{G_{\text{Ant}}, G_{\text{OA}}, G_{\text{Antlions}}, G_{\text{OAL}}\}, \tag{7.1}$$

$$\{G_{\text{Ant}}, G_{\text{Antlion}}\} \rightarrow_{A_2} \{G_{\text{Ant}}, G_{\text{Antlion}}\}, \tag{7.2}$$

and

$$\{G_{\text{Ant}}, G_{\text{Antlion}}\} \rightarrow_{A_3} \{\text{true}, \text{ false}\}, \tag{7.3}$$

where G_{Ant} represents the ants' position matrix, G_{Antlion} is comprised of the ant-lions' position, G_{OA} depicts the fitness of the ants, and finally, G_{OAL} contains the fitness values of the ant-lions. The algorithm operates in such a way that the ant-lion and ant matrices are initialized in a random manner by applying Eq. (7.1). The roulette wheel operator is used to select the location of each ant relative to the ant-lion. Equation (7.2) is used to update the elite in each iteration. The update of the perimeter location is primarily described in relation to the iteration number at that instance. The location is subsequently refined by using two random steps near the selected elite and ant-lion. The fitness function is used to estimate the points where every ant randomly walks. In case any of the ants become more capable than any of the ant-lions, their location will be used in the next iteration as the new location for the ant-lions. There will then be a comparison between the best ant-lion and the best ant-lion obtained during the optimization procedure, and they are substituted in one of the key operations in the implementation of the algorithm. These steps are executed until the function in Eq. (7.3) returns false.

In the implementation of the algorithm, ants walk randomly according to (Mirjalili 2015; Gupta and Saxena 2016):

$$X^a(t) = [0, \text{ cummsum}(2l(t_1) - 1), \text{ cummsum}(2l(t_2) - 1), \ldots, \text{ cummsum}(2l(T_n) - 1)], \tag{7.4}$$

where n is the maximum number of iterations, cummsum represents the cumulative sum, and t indicates the random step walk. $l(t)$ is a stochastic equation defined by the relations:

$$l(t) = 1 \text{ if rand} > 0.5, \tag{7.5}$$

and

$$l(t) = 0 \text{ if rand} \leq 0.5. \tag{7.6}$$

where *rand* represents a random number obtained from a Gaussian distribution in the range [0, 1]. In order to restrict the random movement of ants, with the boundaries of the solution space, they are normalized according to (Mirjalili 2015; Gupta and Saxena 2016)

$$X_i^t = \frac{(X_i^t - a_i) * (d_i - c_i^t)}{(d_i^t - a_i)} + c_i, \tag{7.7}$$

where a_i represents the minimum random walk of the ith variable, b_i indicates the maximum random walk of the ith variable, c_i^t represents the minimum of the ith variable at the tth iteration and finally, d_i^t indicates the maximum of the ith variable at the tth iteration (Mirjalili 2015).

Modelling of the trapping of ants by ant-lion pits can be obtained by (Mirjalili 2015; Gupta and Saxena 2016)

$$c_i^t = \text{Antlion}_j^t + c^t, \tag{7.8}$$

and

$$d_i^t = \text{Antlion}_j^t + d^t, \tag{7.9}$$

where c^t represents the lower bound of all features at the tth step, d^t depicts the maximum of all features at the tth step and Antlion_j^t represents the location of the chosen jth ant-lion at the tth step.

The hunting capability of an ant-lion is described by the fitness proportional roulette wheel selection. The mathematical equation describing the way the ants that are trapped, slide down the trap and towards the ant-lion, is obtained by (Mirjalili 2015; Gupta and Saxena 2016):

$$u^t = \frac{u^t}{Z}, \tag{7.10}$$

and

$$v^t = \frac{v^t}{Z}, \tag{7.11}$$

whereby Z is a ratio calculated by

$$Z = 10^w \frac{t}{T}. \tag{7.12}$$

In the equation above, t is th current step, T represents the upper bound on the number of steps to be taken and w depicts a constant which relies on the current step according to the following relations (Mirjalili 2015):

$$w = \begin{cases} 2 \text{ if } t > ; 0.1T, \\ 3 \text{ if } t > ; 0.5T, \\ 4 \text{ if } t > ; 0.75T, \\ 5 \text{ if } t > ; 0.9T, \\ 6 \text{ if } t > ; 0.95T. \end{cases} \tag{7.13}$$

The last part of the algorithm is elitism which is done such that the fittest ant-lion at each step is said to be the elite. This implies that every ant randomly walks around a selected ant-lion with a location that respects the following equation (Mirjalili 2015; Gupta and Saxena 2016):

$$\mathrm{Ant}_i^t = \frac{R_A^t + R_E^t}{2}. \tag{7.14}$$

In Eq. (7.14), R_A^t represents the random motion around an ant-lion selected using the roulette wheel method at the tth step while R_E^t depicts the random motion around the elite ant-lion at the tth step.

In Gupta and Saxena (2016), the ALO algorithm was used to find the parameters of a primary governor loop of thermal generators for successful automatic generation control (AGC) of a two-area interconnected power system. It has been used by Satheeshkumar and Shivakumar (2016) to investigate a three-area interconnected power system while in Yamany et al. (2015), it was used to train a multilayer perceptron neural network. Zawbaa et al. (2016) used a chaotic ALO algorithm for feature selection purposes from large datasets meanwhile in Petrovi et al. (2015), the ALO algorithm was used to solve the NP-hard combinatorial optimization problem of obtaining an optimal process plan according to all alternative manufacturing resources.

The ALO parameters used in the model implementation are: the number of search agents which is set to 40, and, the maximum number of iterations, given a value of 1000. The number of decision variables depends on the number of missing values in a record. The data was normalized to being in the range [0, 1], meaning the lower and upper bounds of the decision variables are 0 and 1, respectively. These parameters were chosen because they resulted in the more optimal outcomes with different combinations and permutations of values having been tested.

7.4 Missing Data Estimation Model

The methodology implemented in this chapter comprises primarily preprocessing the data from the dataset. This procedure constitutes the normalization of the data which reduces the variation in the values between feature variables, in addition to ensuring that the network generates representative outputs. The ant-lion optimizer algorithm is applied to reduce an error function derived from training a deep learning linear regression model using the stochastic gradient descent method. A portion of the normalized data from the training set is presented to the deep learning network architecture for training. An error function is then derived which is defined mathematically by calculating the square of the disparity between the real outputs and the estimated model outputs. In this chapter, data entries from the test set from any of the feature variables could be missing simultaneously, and therefore, the error function is reformulated to incorporate the unknown and known input values. The

normal routine is to create missing values in a specific feature, and then estimate these. The uncontrolled nature of the missing data within the test set of data to the best of our knowledge is an aspect which has not been investigated and reported as well. Restricted Boltzmann machines (RBMs) are used to train the individual layers of the network in an unsupervised manner, which are subsequently joined together to form the encoding part of the network, and then transposed to make up the decoding part. The stochastic gradient descent (SGD) algorithm is applied to train the network using the training set of data in a supervised learning manner. The optimal network structure is constructed and consists of an input layer, seven hidden layers and an output layer. The number of nodes in the hidden layers is obtained from an initial suggestion made by Hinton et al. (2006) and by performing cross-validations on a held out set of the training data, known as the validation data. With the optimal network structure obtained via training, the swarm algorithm implemented is used to identify the optimal network and algorithm parameter combinations. The missing data estimation procedure is then performed with the parameters identified in the previous step. The expected outputs are compared against the estimated outputs to yield understanding into how the method performs.

To assess the performances of the methods as high-dimensional estimators of the missing values, an image recognition dataset is used. Entries from the test set of data were removed and approximated using the models. These models were all coded in MATLAB®. To measure the accuracies of the methods as estimators, eight error metrics were used, these being: Squared error (SE), mean square error (MSE), mean absolute error (MAE), root mean squared logarithmic error (RMSLE), global deviation (GD), relative prediction accuracy (RPA), signal-to-noise ratio (SNR) and coefficient of determination (COD). These error metrics were selected since they been applied in a variety of research reports (Marivate et al. 2007; Abdella and Marwala 2005; Leke et al. 2014; Mohamed et al. 2007) as performance measures for missing data estimation problems in addition to them being convenient. The correlation coefficient (r) between the estimated and expected output values is also used to provide more insight into the relationship between the estimated values and the real values. Statistical t-tests are also performed to back the findings from the metrics and provide information as to the statistical significance of the results obtained.

7.5 Experimental Results

Figures 7.1, 7.2, 7.3 and 7.4 show the performance and comparison of DL-ALO against the MLP-PSO, MLP-SA and MLP-GA approaches. The MLP network used has a structure of 784-400-784, with 784 input and output nodes in the input and output layers, respectively, and, 400 nodes in the one hidden layer. This number is obtained by testing the network with different numbers of nodes in the hidden layer and observing the network structure which leads to the lowest possible network error.

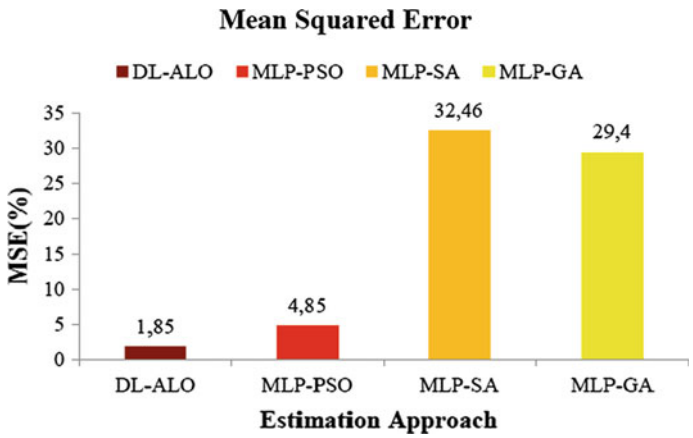

Fig. 7.1 Mean squared error versus estimation approach

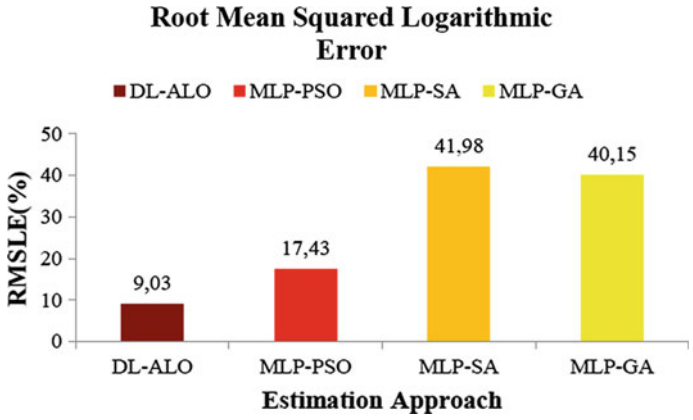

Fig. 7.2 Root mean squared logarithmic error versus estimation approach

Figures 7.1 and 7.2 are bar charts that show the MSE and RMSLE values for DL-ALO when compared to the MLP-PSO, MLP-SA and MLP-GA.

We observed 1.85, 4.85, 32.46 and 29.4% of MSE, and 9.03, 17.43, 41.98 and 40.15% of RMSLE for DL-ALO, MLP-PSO, MLP-SA and MLP-GA, respectively. The DL-ALO yielded the lowest MSE value when compared to the other approaches. These results were validated by the correlation coefficient whose bar chart is given in Fig. 7.3.

The DL-ALO and MLP-PSO yielded 92.49 and 78.62% correlation values, respectively, while the MLP-SA and MLP-GA showed correlations of 4.03 and 8.29%, respectively.

The MLP-SA and the MLP-GA yielded 10.92 and 11.42% of RPA, respectively, while the DL-ALO and the MLP-PSO, respectively, yielded 81.33 and 54.83%, as

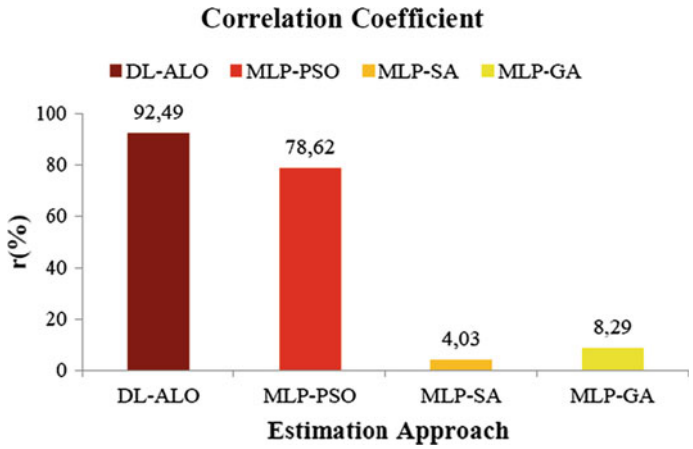

Fig. 7.3 Correlation coefficient versus estimation approach

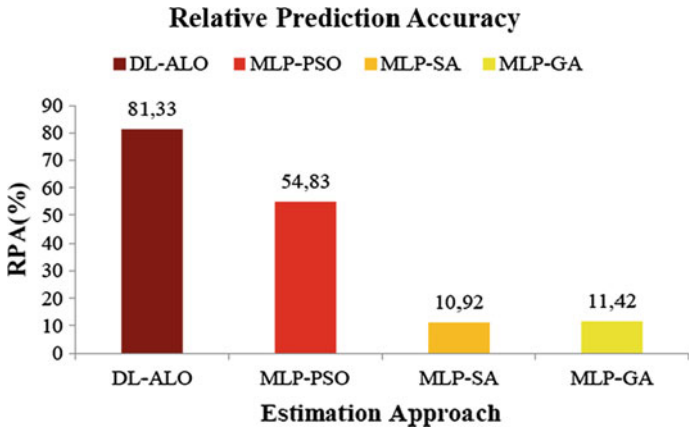

Fig. 7.4 Relative prediction accuracy versus estimation approach

shown in Fig. 7.4. As observed, the DL-ALO approach obtained the best values for all four metrics presented graphically.

In Table 7.1, the dimensions column refers to the number of missing values in a sample/record. Tables 7.1 and 7.2 further back the findings from Figs. 7.1, 7.2, 7.3 and 7.4 revealing that the proposed DL-ALO approach yielded the lowest objective function values in the estimation of missing values per sample, as well as the best COD, GD, MAE, SE and SNR values.

Table 7.2 indicates that there is no significant difference in the means of the estimates obtained by the DL-ALO when compared to the MLP-PSO, MLP-SA and MLP-GA, yielding p-values of 0.46, 0.34 and 0.16 when DL-ALO is compared

Table 7.1 DL-ALO mean squared error objective value per sample

Sample	Dimensions	DL-ALO	MLP-PSO	MLP-SA	MLP-GA
1	81	**1.29**	9.46	6.57	6.57
2	72	**1.63**	7.39	7.65	7.65
3	85	**3.65**	7.66	10.57	10.57
4	88	**1.13**	7.75	6.03	6.03
5	77	**2.21**	6.28	9.09	9.09
6	89	**1.45**	6.49	13.55	13.55
7	84	**2.70**	5.79	6.77	6.77
8	75	**1.14**	5.30	9.11	9.11
9	71	**1.22**	5.67	6.31	6.31
10	85	**1.67**	5.63	16.55	16.55

The bold values indicate the proposed method yields the best results

Table 7.2 Statistical analysis of DL-ALO results

Pairs compared	*P*-Values (95% confidence level)
DL-ALO:MLP-PSO	0.46
DL-ALO:MLP-SA	0.34
DL-ALO:MLP-GA	0.16

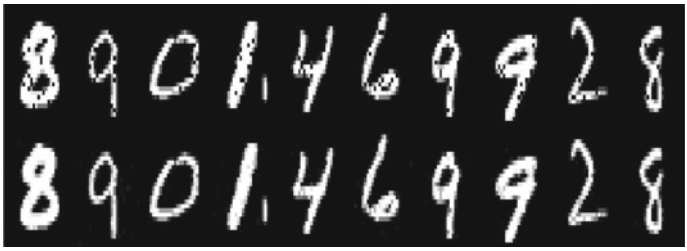

Fig. 7.5 Top row: Corrupted images—Bottom row: DL-ALO reconstructed images

against the MLP-PSO, MLP-SA and MLP-GA, respectively, at a 95% confidence level. Therefore, the null hypothesis can be accepted.

In the top row of Fig. 7.5, we depict 10 images with missing pixel values which are to be estimated prior to the classification tasks being performed by statistical methods. In the bottom row of the same figure, we show the reconstructed images from using the DL-ALO approach. In the top and bottom rows of Fig. 7.6, we observe the reconstructed images when the MLP-PSO and MLP-GA approaches are used, respectively. It can be seen that the reconstructed images using the MLP-PSO and the MLP-GA introduce a lot of noise, more so in the bottom row than in the top row, in contrast to when the DL-ALO approach is applied. Furthermore, closer inspection reveals that the images are not fully reconstructed as not all pixel values within the images are estimated correctly.

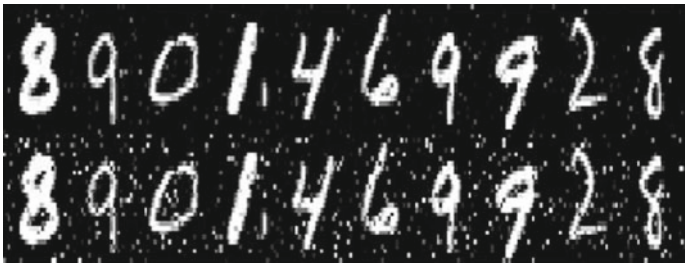

Fig. 7.6 Top row: MLP-PSO reconstructed images—Bottom row: MLP-GA reconstructed images

7.6 Conclusion

This chapter presented an investigation of the estimation of missing data via a novel approach. The estimation method comprises of a deep autoencoder network to replicate the input data, in combination with the ant-lion optimizer algorithm to estimate the missing data. The performance of the model is investigated and compared against existing methods including an MLP autoencoder with genetic algorithm, simulated annealing and particle swarm optimization. The results obtained reveal that the proposed system yields more accurate estimates when compared against the MLP hybrid systems. This is made evident when the mean squared error, root mean squared logarithmic error, correlation coefficient, relative prediction accuracies, signal-to-noise ratio and global deviation are considered, with the proposed approach yielding the best values of these. Also, when the objective function value is considered during the estimation process, it is observed that the proposed approach results in the lowest values for each instance.

References

Abdella, M., & Marwala, T. (2005). *The use of genetic algorithms and neural networks to approximate missing data in database* (vol. 24, pp. 577–589).

Aydilek, I., & Arslan, A. (2012). A novel hybrid approach to estimating missing values in databases using k-nearest neighbors and neural networks. *International Journal of Innovative Computing, Information and Control, 7*(8), 4705–4717.

Baraldi, A., & Enders, C. (2010). An introduction to modern missing data analyses. *Journal of School Psychology, 48*(1), 5–37.

Carter, R. L. (2006). Solutions for missing data in structural equation modelling. *Research & Practice in Assessment, 1*(1), 1–6.

Gupta, E., & Saxena, A. (2016). Performance evaluation of antlion optimizer based regulator in automatic generation control of interconnected power system. *Journal of Engineering, 2016,* 1–14.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.

Jerez, J. M., Molina, I., Garcia-Laencina, P. J., Alba, E., Ribelles, N., Martin, M., et al. (2010). Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine, 50*(2), 105–115.

Leke, C., Twala, B., & Marwala, T. (2014). Modeling of missing data prediction: Computational intelligence and optimization algorithms. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)* (pp. 1400–1404). San Diego, CA, USA.

Lobato, F., Sales, C., Araujo, I., Tadaiesky, V., Dias, L., Ramos, L., & Santana, A. (2015). Multi-objective genetic algorithm for missing data imputation. *Pattern Recognition Letters, 68*(1), 126–131, Retrieved March 18, 2016.

Marivate, V. N., Nelwamondo, F. V., & Marwala, T. (2007). Autoencoder, principal component analysis and support vector regression for data imputation. arXiv:0709.2506.

Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 8,* 80–98.

Mistry, F. J., Nelwamondo, F. V., & Marwala, T. (2009). Missing data estimation using principle component analysis and autoassociative neural networks. *Journal of Systemics, Cybernatics and Informatics, 7*(3), 72–79.

Mohamed, A. K., Nelwamondo, F. V., & Marwala, T. (2007). Estimating missing data using neural network techniques, principal component analysis and genetic algorithms. In *Proceedings of the Eighteenth Annual Symposium of the Pattern Recognition Association of South Africa*.

Nelwamondo, F. V., Mohamed, S., & Marwala, T. (2007). Missing data: A comparison of neural network and expectation maximisation techniques. *Current Science, 93*(12), 1514–1521.

Petrovi, M., Petronijevi, J., Miti, M., Vukovi, N., Plemi, A., Miljkovi, Z., et al. (2015). The ant lion optimization algorithm for flexible process planning. *Journal of Production Engineering, 18*(2), 65–68.

Rana, S., John, A. H., Midi, H., & Imon, A. (2015). Robust regression imputation for missing data in the presence of outliers. *Far East Journal of Mathematical Sciences, 97*(2), 183–195.

Satheeshkumar, R., & Shivakumar, R. (2016). Ant lion optimization approach for load frequency control of multi-area interconnected power systems. *Circuits and Systems, 7,* 2357–2383.

Sidekerskiene, T., & Damasevicius, R. (2016). Reconstruction of missing data in synthetic time series using EMD. *CEUR Workshop Proceedings, 1712,* 7–12.

Van Buuren, S. (2012). *Flexible imputation of missing data*. CRC press.

Yamany, W., Tharwat, A., Fawzy, Gaber, T., & Hassanien, A. E. (2015). A new multilayer perceptrons trainer based on ant lion optimization algorithm. In *Fourth International Conference on Information Science and Industrial Applications (ISI)* (pp. 40–45).

Zainuri, N. A., Jemain, A. A., & Muda, N. (2015). A comparison of various imputation methods for missing values in air quality data. *Sains Malaysiana, 44*(3), 449–456.

Zhang, S., Jin, Z., & Zhu, X. (2011). Missing data imputation by utilizing information within incomplete instances. *Journal of Systems and Software, 84*(3), 452–459.

Zhang, S. (2011). Shell-neighbor method and its application in missing data imputation. *Applied Intelligence, 35*(1), 123–133.

Zawbaa, H. M., Emary, E., & Grosan, C. (2016). Feature selection via chaotic antlion optimization. *PLOS ONE, 11*(3),1–21. Retrieved June 2016, from https://doi.org/10.1371/journal.pone.0150652.

Chapter 8
Missing Data Estimation Using Invasive Weed Optimization Algorithm

8.1 Introduction

Missing data is an ever-present challenge in the real world. Decision-making processes that rely on accurate knowledge are quite reliant upon the availability of data from which information can be extracted. Such processes often require predictive models or other computational intelligence techniques that use the observed data as inputs. However, in some cases, data could be lost, corrupted or recorded incompletely, which affects the quality of the data negatively. The majority of the decision-making and machine learning frameworks such as artificial neural networks, support vector machines and principal component analysis cannot be used for decision-making and data analysis if the data is incomplete because these missing values can critically influence pattern recognition and classification tasks. Since the decision output should still be maintained despite the missing data, it is important to deal with the problem. Therefore, in case of incomplete or missing data, the initial step in processing the data is estimating the missing values.

The applications of missing data estimation methods are numerous. With that said, the existing methods depend on the nature of the data, the pattern of missingness and are predominantly implemented on low-dimensional datasets. Abdella and Marwala (2005) implemented a joint neural network-genetic algorithm framework to impute missing values in one dimension while Rana et al. (2015) used robust regression imputation in datasets with outliers and investigated its performance. In Zhang et al. (2011), the nonparametric iterative imputation algorithm (NIIA) method was introduced. It is a method that makes use of the knowledge in relevant data instances when estimating missing data. In papers such as Leke and Marwala (2016), Liew et al. (2011), Jerez et al. (2010), Myers (2011) and Van Buuren (2012), new techniques to impute missing data and comparisons between these and existing methods are observed. These techniques mainly cater to low-dimensional datasets with missing values but are less effective when high-dimensional datasets are considered with missingness occurring in an uncontrolled manner as will be depicted subsequently

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_8

in the chapter. The main motivation behind this work is, therefore, to introduce high-dimensional missing data estimation approaches, with emphasis on image recognition datasets. These approaches will be used to reconstruct corrupted images by estimating image pixel values.

To investigate new techniques to address the previously mentioned dimensionality drawback, we implement a deep autoencoder, with an autoencoder network being an unsupervised learning technique that endeavours to reconstruct the input data as the output by learning an approximation function which decreases the contrast between the original data, x, and the output reconstructed data, $\tilde{x}$, the same as obtained with principal component analysis (PCA). A deep autoencoder is made up of two symmetrical deep belief networks with a couple of shallow layers representing the encoding section of the network, while the second set of shallow layers represents the decoding section, with each of the individual layers being restricted Boltzmann machines (RBMs). These networks are applicable in a variety of sectors, for example, deep autoencoders were used for visuomotor learning in Finn et al. (2016) while in Ju et al. (2015), they were used with support vector machines (SVMs) to learn sparse features and perform classification tasks. Feng et al. (2014) used a deep denoising autoencoder to generate robust speech features for a noisy reverberant speech recognition system, and these networks were used to map small colour images to short binary codes (Krizhevsky and Hinton 2011). Autoencoders have several advantages such as being more flexible by introducing non-linearity in the encoding part of the network contrary to the likes of PCA, which is a key property of the pretraining procedure, they require no prior knowledge of the data properties and correlations, and they are intuitive. The main drawback of this network is its need for a significant number of samples for training and learning, which are not always readily available.

In this chapter, we propose a method for missing data estimation in high-dimensional datasets, namely, DL-IWO, based on the advantages of deep autoencoders and the invasive weed optimization algorithm. For this purpose, we begin by describing the optimization algorithm followed by the proposed model for the imputation task which consists of a deep autoencoder and the optimization algorithm. Subsequently, we present analyses of the performance of the proposed model and compare the results obtained to those of existing methods. Furthermore, we report on the results and discuss the findings from our experiments. Finally, brief conclusions are presented.

8.2 Invasive Weed Optimization (IWO) Algorithm

In Mehrabian and Lucas (2006), the authors proposed the invasive weed optimization (IWO) algorithm which mimics the invasion trait of weeds. In general, weeds are defined as plants that grow in an area where they are not wanted. Specific to horticulture, the term weed refers to a plant whose growth properties pose a menace to plants that are cultivated. Weeds portray fascinating traits such as adaptivity and robustness. In the invasive weed optimization algorithm, weeds are replaced by

points in the solution space in which a colony of points grows to an optimal value (Veenhuis 2010).

Let us say for instance that D represents the dimension of a problem, implying that the dimension of the search space is R^D. Let us further assume that P_{init} is the initial weed population size and $P_{\max}$ is the upper bound on the population size such that $1 \leq P_{\text{init}} \leq P_{\max}$ (Veenhuis 2010). Also, let W define the set of weeds with $W = \{W_1, ..., W_{\|W\|}\}$ (Kasdirin et al. 2017). Every one of the weeds, $W_i \in R^D$ represents a location in the solution space. Computing the fitness of the weed requires the use of a fitness function of the form: $F : R^D \rightarrow R$ (Veenhuis 2010).

There are two main parts to the IWO algorithm, these being: Initialization and iteration. In the initialization step, the counter of the generation of solutions is set to zero, $G = 0$. Subsequently, the initial population, W, is generated randomly by creating P_{init} with uniformly distributed values (Veenhuis 2010):

$$W_i \sim U(X_{\min}, X_{\max})^D. \tag{8.1}$$

where $X_{\min}$ and $X_{\max}$ represent the lower and upper bounds of the solution space, respectively, and are problem specific.

In the iteration step, each of the weeds in the current population is regenerated by a given number of seeds. S_{num}, defines the number of seeds, and is evaluated such that it is proportional to the fitness value of the weed being considered. This implies that it is linearly mapped depending on the population's worse and best fitness, F_{worse} and F_{best} (Veenhuis 2010):

$$S_{\text{num}} = S_{\min} + \frac{F(W_i) - F_{\text{worse}}}{F_{\text{best}} - F_{\text{worse}}}(S_{\max} - S_{\min}). \tag{8.2}$$

In Eq. 8.2, $S_{\min}$ and $S_{\max}$ represent the lower and upper bounds of seeds allowed for each weed (Veenhuis 2010). All the S_{num} seeds, S_j, are generated near the current weed by making use of a Gaussian distribution with varying standard deviation and zero mean (Veenhuis 2010):

$$S_j = W_i + \mathcal{N}(0, \sigma_G)^D. \tag{8.3}$$

In Eq. 8.3, $1 \leq \text{j} \leq S_{\text{num}}$. The standard deviation, σ_G begins at σ_{init} and is reduced in a non-linear manner over the entire run to σ_{final}. The standard deviation for the current generation is calculated by using (Veenhuis 2010)

$$\sigma_G = \sigma_{\text{final}} + \frac{(N_{\text{iter}} - G)^{\sigma_{\text{mod}}}}{(N_{\text{iter}})^{\sigma_{\text{mod}}}}(\sigma_{\text{init}} - \sigma_{\text{final}}), \tag{8.4}$$

where N_{iter} represents the upper bound on the number of generations and σ_{mod} is the non-linear modulation indicator. Following these procedures, the subsequent population is generated by uniting the current population with all the generated seeds of all the weeds. If the volume of the new population attains $P_{\max}$, this new

Table 8.1 IWO parameters

Parameter	Value
Initial population size	10
Maximum population size	40
Minimum number of seeds	0
Maximum number of seeds	5
Variance reduction exponent	2
Maximum number of iterations	1000

population is categorized as per the fitness value, and the $P_{\max}$ optimal weeds are preserved. The least favourable weeds are removed.

This algorithm has been used to investigate the time-cost-quality trade-off in projects (Paryzad and Pour 2013). Hung et al. (2010) developed a novel receiver that merges constant modulus approach (CMA) blind adaptive multiuser detection with IWO for multi-carrier code division multiple access (MC-CDMA). In Su et al. (2014), the services selection problem is modelled as a non-linear optimization problem with constraints and solved using a discrete version of the IWO algorithm. Unconstrained and constrained optimization problems are solved using a hybrid IWO and firefly algorithm in Kasdirin et al. (2017). Finally, in Yazdani and Ghodsi (2017), the problem of minimizing the total weighted tardiness and earliness criteria on a single machine is considered.

The IWO parameters used in the model implementation are given in Table 8.1 except for the number of dimensions which depends on the number of missing values in a record. The data was normalized to being in the range [0, 1], meaning the lower and upper bounds of the decision variables are 0 and 1, respectively. These parameters were chosen because they resulted in the more optimal outcomes with different combinations and permutations of values having been tested.

8.3 Missing Data Estimation Model

In this section, we describe the proposed approach that combines the advantages of deep autoencoder networks and the invasive weed optimization algorithm. When applying a network of any nature to estimate missing data, it is very important that the network model be very accurate. The proposed methodology is summarized in Fig. 8.1, where I_k and I_u represent the known and unknown features, respectively, and make up the set of inputs.

Firstly, the network is trained using the stochastic gradient descent (SGD) algorithm, with a complete set of records to identify a mapping function, f, that extracts and stores the interrelationships and correlations between features in the dataset. This leads to the output, $\vec{O}$, obtained by

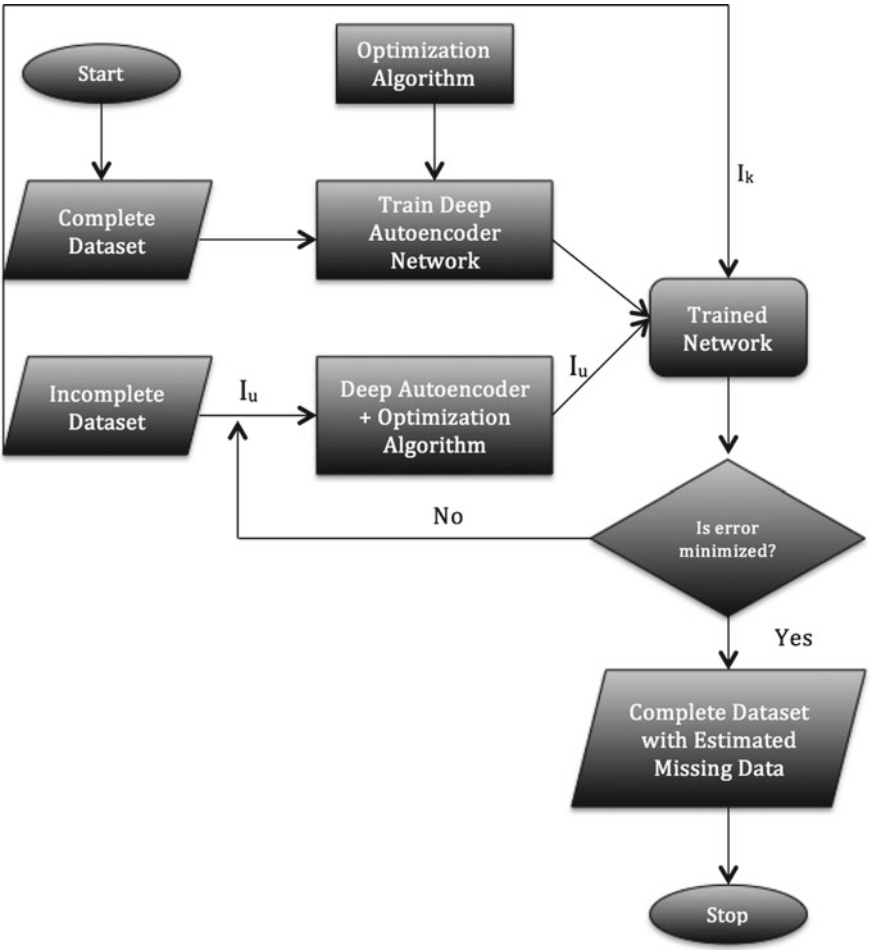

Fig. 8.1 Flow chart representation of the estimation model. © 2017 IEEE. Reproduced with permission from Leke et al. (2017)

$$\vec{O} = f\left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}\right). \tag{8.5}$$

Due to the fact that the autoencoder framework tries to represent the input as the output of the network, the output is approximately equal to the input. This results in a small error which relies upon the accuracy of the autoencoder. The error is expressed as

$$\vec{e} = \begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}, \vec{w}\right). \tag{8.6}$$

To guarantee that the error is always positive, Eq. (8.6) is squared, yielding the following equation:

$$\vec{e} = \left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}, \vec{w}\right)\right)^2. \tag{8.7}$$

The error will be minimized when the output is as close as possible to the input. This predominantly occurs when the data presented to the framework has similar characteristics as those recorded during training. Therefore, during the estimation procedure, minimizing the error symbolizes the optimization of the missing data in such a way that the new data matrix matches the pattern yielded by the complete dataset during training. The invasive weed optimization algorithm is used to further optimize the network weights, because as mentioned before, it is guaranteed to attain a global optimum point, thereby making sure the network is as accurate as possible. The algorithm as an optimization technique tries to attain a global optimum point which in the landscape of the error function, being the mean squared error, is the lowest point or valley of the function. This procedure is expressed mathematically as

$$\underset{\vec{w}}{\text{minimize}} \left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}, \vec{w}\right)\right)^2. \tag{8.8}$$

The incomplete dataset is passed as input to the bat algorithm which has the optimized deep neural network as part of the objective function. The missing data are estimated and together with the known values in the input matrix, run through the trained network to observe whether the network error has been minimized, without any changes for a few iterations. If it has, the dataset is returned with imputed values, otherwise, the imputation procedure is executed again. The stopping criteria for the missing data estimation procedure using the invasive weed optimization algorithm are either 40,000 function evaluations having been executed, or there being no change in the objective function value.

From the above description of how the deep autoencoder and the bat algorithm are used, the equation below summarizes the hybrid function of the proposed approach, with f_{IWO} being the bat algorithm data estimation operation and f_{DAE} being the function of the deep autoencoder.

$$y = f_{\text{DAE}}(W, f_{\text{IWO}}(\vec{I})), \tag{8.9}$$

where $\vec{I} = \begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}$ represent the input space of known and unknown features. This equation represents the model design whereby the complete input vector with known and estimated missing data entries obtained by executing the missing data estimation procedure (f_{IWO}) is presented to the deep regression model (f_{DAE}) to observe whether the network error has been minimized. If it is found that the error has been minimized, the output, y, will contain the known input vector values and the optimal missing data estimated values.

8.4 Experimental Results

To investigate and validate the proposed model against other existing approaches, the Mixed National Institute of Standards and Technology (MNIST) dataset of handwritten digits was used. It consists of 60,000 training samples and 10,000 test samples, each with 784 features representing the individual pixel values of the image. The black pixels are represented by 0, while the white pixels have values of 255, as such, all the variables could be considered to be continuous. The data in its current form has been cleaned of outliers, therefore, each image used in the dataset should contribute to generalizing the system. The data are normalized to be in the range [0, 1] and randomized to improve the network performance. The training set of data is further split into training and validation sets (50,000 and 10,000, respectively).

The neural network architecture is optimized using the validation set of data with the correctness in the performance of the proposed system being tested using the test data. It is worth mentioning that the objective of this research is not to propose a state-of-the-art classification model which rivals existing state-of-the-art methods but rather to propose an approach to reconstruct an image in the event of missing data (missing pixels). The optimized deep network architecture used is 784-1000-500-250-30-250-500-1000-784 as suggested initially by Hinton et al. (2006) and further verified by performing cross-validations using the validation set of data, with 784 input and output nodes, and seven hidden layers with 1000, 500, 250, 30, 250, 500 and 1000 nodes, respectively. The objective function value (mean squared error) obtained after training was 1.96% which was close to the lowest value obtained during cross-validation (2.12%). The multilayer perceptron (MLP) autoencoder used for comparison purposes has the same number of nodes in the input and output layer as the deep autoencoder, with a single hidden layer consisting of 400 nodes which is determined by varying the number of nodes in the hidden layer, and determining which architecture produces the lowest network error. It was experimentally observed that the 784-400-784 MLP autoencoder network architecture yielded a network error value of 2.5088%. The sigmoid and linear activation functions were used for the hidden and output layers of the MLP network, respectively. The sigmoid activation function is used in each layer of the deep autoencoder except for the bottleneck layer where the linear activation function is used. The training is done using the

scaled conjugate gradient (SCG) algorithm for the MLP and the stochastic gradient descent (SGD) algorithm for the deep autoencoder network, for 500 epochs. 100 samples are selected at random from the test set of data and features are then removed respecting the missing at random (MAR) and missing completely at random (MCAR) mechanisms as well as the arbitrary missing data pattern. This is done by creating a binomial matrix of the same size as the test data with zeros and ones adhering to the stated mechanisms and pattern, and replacing every occurrence of one with NaN (implying missingness). The dataset obtained from this with missing values is used to test the performance of the missing data estimation scheme, which is then compared to existing methods.

The effectiveness of the proposed approach is estimated using the mean squared error (MSE), the root mean squared logarithmic error (RMSLE), the correlation coefficient (r) and the relative prediction accuracy (RPA). Also used are the signal-to-noise ratio (SNR) and global deviation (GD). The mean squared and root mean squared logarithmic errors as well as the global deviation yield measures of the difference between the actual and predicted values and provide an indication of the capability of the estimation.

$$\text{MSE} = \frac{\sum_{i=1}^{n}(I_i - \hat{I}_i)^2}{n}, \tag{8.10}$$

$$\text{RMSLE} = \sqrt{\frac{\sum_{i=1}^{n}(\log(I_i + 1) - \log(\hat{I}_i + 1))^2}{n}}, \tag{8.11}$$

and

$$\text{GD} = \left(\frac{\sum_{i=1}^{n}(\hat{I}_i - I_i)}{n}\right)^2. \tag{8.12}$$

The correlation coefficient provides a measure of the similarity between the predicted and actual data. The output value of this measure lies in the range $[-1, 1]$ where the absolute value indicates the strength of the link, while the sign indicates the direction of the said link. Therefore, a value close to 1 signifies a strong predictive capability while a value close to -1 signifies otherwise. In the equation below, $\bar{I}$ represents the mean of the data.

$$r = \frac{\sum_{i=1}^{n}(I_i - \bar{I}_i)(\hat{I}_i - \overline{\hat{I}}_i)}{\left[\sum_{i=1}^{n}(I_i - \bar{I}_i)^2 \sum_{i=1}^{n}(\hat{I}_i - \overline{\hat{I}}_i)^2\right]^{1/2}}. \tag{8.13}$$

The relative prediction accuracy (RPA) on the other hand measures the number of estimates made within a specific tolerance with the tolerance dependent on the sensitivity required by the application. The tolerance is set to 10% as it seems favourable for the application domain. This measure is given by

$$\text{RPA} = \frac{n_\tau}{n} \times 100. \tag{8.14}$$

The Squared Error (SE) measures a quadratic scoring rule that records the average magnitude of the error. This can be obtained by calculating the variance square root, also referred to as the standard deviation. It reduces the variance in the error, contrary to the MSE, hence, its application in this work. The SE can be obtained by using the formula:

$$\text{SE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(I_i - \hat{I}_i)^2}. \tag{8.15}$$

The mean absolute error (MAE) measures the average magnitude of the errors in a dataset without considering direction. Under ideal scenarios, SE values are always greater than the MAE values, and in case of equality, the error values are said to have the same magnitude. This error can be calculated using the following equation:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|I_i - \hat{I}_i\right|. \tag{8.16}$$

The coefficient of determination is a metric regularly applied in statistical analysis tasks aimed at assessing the performance of a model in the explanation and prediction of future outputs. It is also referred to as the R-squared statistic obtained by the following:

$$\text{COD} = \left(\frac{\sum_{i=1}^{n}(I_i - \bar{I}_i)(\hat{I}_i - \overline{\hat{I}}_i)}{\left[\sum_{i=1}^{n}(I_i - \bar{I}_i)^2 \sum_{i=1}^{n}(\hat{I}_i - \overline{\hat{I}}_i)^2\right]^{1/23}}\right)^2. \tag{8.17}$$

The signal-to-noise ratio used in this chapter is obtained by

$$\text{SNR} = \frac{\text{var}(I - \hat{I})}{\text{var}(\hat{I})}, \tag{8.18}$$

where in Eqs. (8.10)–(8.13) and Eqs. (8.15)–(8.17), n is the number of samples while I and $\hat{I}$ represent the real test set values and predicted missing output values, respectively. In Eq. (8.14), n_r represents the number of correctly predicted outputs.

In this analysis, the novel DL-IWO approach is compared against existing approaches in the literature (MLP-PSO MLP-SA and MLP-GA). The results are grouped in Figs. 8.2, 8.3, 8.4 and 8.5 and Tables 8.2 and 8.3. The results reveal that the DL-IWO approach outperforms all the other approaches. The mean squared error is given in Fig. 8.2. It shows a 0.45% of error for the DL-IWO while the MLP-PSO, MLP-SA and MLP-GA yielded 5.53, 31.86 and 32.61% error, respectively.

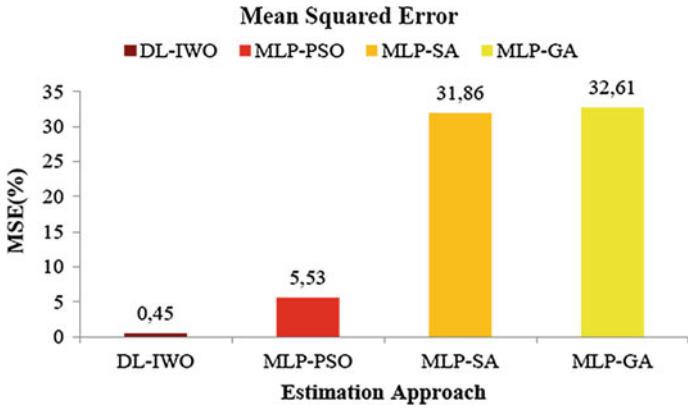

Fig. 8.2 Mean squared error versus estimation approach

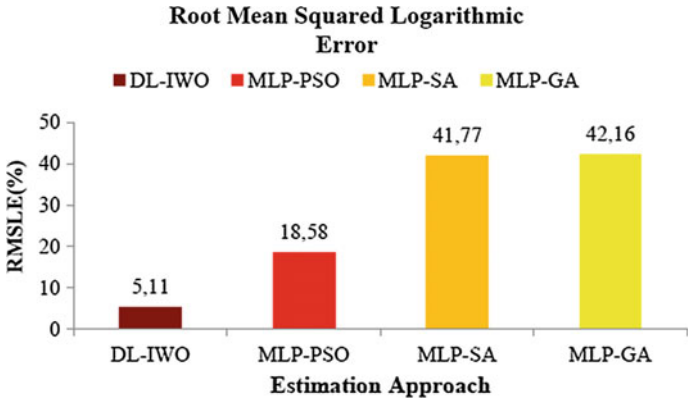

Fig. 8.3 Root mean squared logarithmic error versus estimation approach

Figure 8.3 depicts the root mean squared logarithmic error for the approaches analysed, including the novel DL-IWO approach. It confirms better performance of the DL-IWO approach when compared to the other approaches. The DL-IWO exhibits a 5.11% of RMSLE while we obtained 18.58% of RMSLE for the MLP-PSO. MLP-SA and MLP-GA showed RMSLE values of 41.77 and 42.16%, respectively.

In Fig. 8.4, the RPA values of the approaches are revealed while in Fig. 8.5, we depict the correlation coefficient values of these approaches. In Fig. 8.4, the DL-IWO yields the better RPA value of 88.25% compared to MLP-PSO, MLP-SA and MLP-GA yielding values of 54.42, 8.75 and 10.25%, respectively.

Furthermore, in Fig. 8.5, the DL-IWO approach shows better correlation coefficient value of 97.67%, while the MLP-SA and the MLP-GA showed −3.7 and 0.28% correlation between the estimates and the real values. The MLP-PSO approach is the second-best performer resulting in a value of 73.65%.

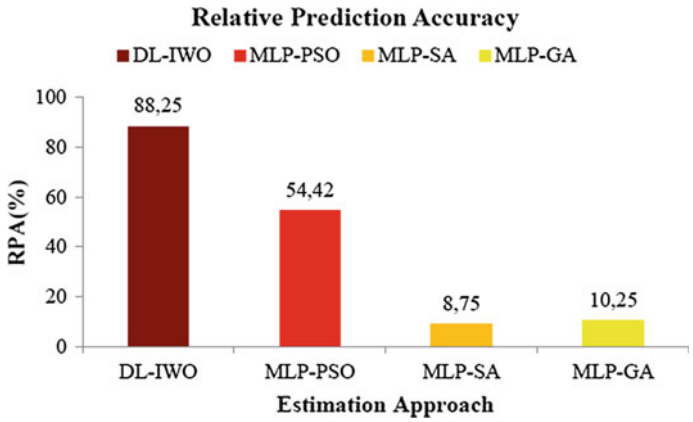

Fig. 8.4 Relative prediction accuracy versus estimation approach

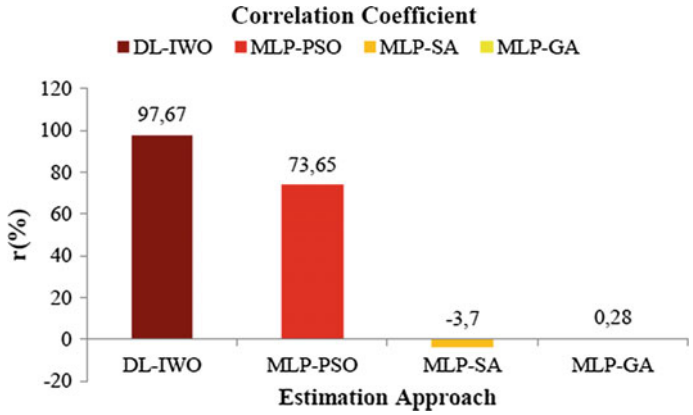

Fig. 8.5 Correlation coefficient versus estimation approach

In Table 8.2, the dimensions column refers to the number of missing values in a sample/record. Tables 8.2 and 8.3 further back the findings from Figs. 8.2, 8.3, 8.4 and 8.5 showing that the proposed DL-IWO approach yielded the lowest objective function values in the estimation of missing values in each sample, as well as the best COD, GD, MAE, SE and SNR values.

In Table 8.4, we present results obtained from statistically analysing the estimates obtained by the DL-IWO approach when compared against the MLP-PSO, MLP-SA and MLP-GA approaches using the t-test. The t-test null hypothesis (H_0) assumes that there is no significant difference in the means of the missing data estimates obtained by the DL-IWO, MLP-PSO, MLP-SA and MLP-GA methods. The alternative hypothesis (H_A), however, indicates that there is a significant difference in the means of the missing data estimates obtained by the four methods.

Table 8.2 DL-IWO mean squared error objective value per sample

Sample	Dimensions	DL-IWO	MLP-PSO	MLP-SA	MLP-GA
1	76	**2.07**	11.48	9.04	9.04
2	89	**0.69**	6.01	5.81	5.81
3	82	**4.74**	11.26	14.80	14.80
4	86	**1.21**	8.42	6.19	6.19
5	82	**2.59**	7.03	7.70	7.70
6	62	**1.02**	4.65	8.71	8.71
7	88	**1.69**	7.63	12.61	12.61
8	71	**4.89**	12.21	12.09	12.09
9	79	**1.01**	4.24	8.14	8.14
10	84	**1.34**	7.06	13.36	13.36

The bold values indicate further back the findings that the proposed method yields the best results

Table 8.3 DL-IWO additional metrics

Method	DL-IWO	MLP-PSO	MLP-SA	MLP-GA
COD	**95.4**	54.24	0.14	0.08
GD	**0.02**	1.10	13.89	14.83
MAE	**3.31**	15.20	48.72	48.91
SE	**6.67**	23.52	56.44	57.1
SNR	**5.16**	59.76	219.89	205.01

The bold values indicate further back the findings that the proposed method yields the best results

Table 8.4 Statistical analysis of DL-IWO model results

Pairs compared	P-Values (95% confidence level)
DL-IWO:MLP-PSO	2.51×10^{-15}
DL-IWO:MLP-SA	2.0×10^{-174}
DL-IWO:MLP-GA	4.0×10^{-180}

Table 8.4 reveals that there is a significant difference at a 95% confidence level in the means of the estimates obtained by the DL-IWO when compared to the MLP-PSO, MLP-SA and MLP-GA, yielding p-values of 2.51×10^{-15}, 2.0×10^{-174} and 4.0×10^{-180}, respectively, when all three pairs are compared. This, therefore, indicates that the null hypothesis (H_0), which assumes that there is no significant difference in the means between the DL-IWO and the other three methods can be rejected in favour of the alternative hypothesis (H_A) at a 95% confidence level.

In the top row of Fig. 8.6, we depict 10 images with missing pixel values which are to be estimated prior to classification tasks being performed by statistical methods. In the bottom row of the same figure, we show the reconstructed images from using the DL-IWO approach, while in the top and bottom rows of Fig. 8.7, we observe the reconstructed images when the MLP-PSO and MLP-GA approaches were used,

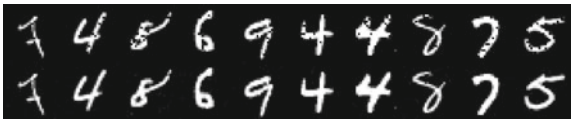

Fig. 8.6 Top row: Corrupted images—Bottom row: DL-IWO reconstructed images

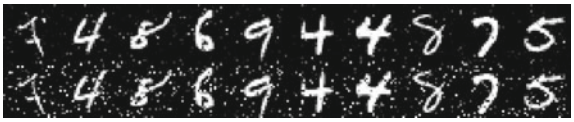

Fig. 8.7 Top row: MLP-PSO reconstructed images—Bottom row: MLP-GA reconstructed images

respectively. The reconstructed images using the MLP-PSO and MLP-GA introduce much noise, more so in the bottom row than in the top row, as opposed to when the DL-IWO approach was applied. Furthermore, closer inspection reveals that the images are not fully reconstructed as not all pixel values within the images are estimated correctly.

8.5 Conclusion

This chapter presented an investigation of the estimation of missing data via a novel approach. The estimation method comprises of a deep autoencoder network to replicate the input data, in combination with the invasive weed optimization algorithm to estimate the missing data. The performance of the model is investigated and compared against existing methods including the MLP autoencoder with genetic algorithm, simulated annealing and particle swarm optimization. The results obtained reveal that the proposed system yielded more accurate estimates when compared to the MLP hybrid systems. This is made evident when the mean squared error, root mean squared logarithmic error, correlation coefficient, relative prediction accuracies, signal-to-noise ratio and global deviation are considered, with the proposed approach yielding the best values of these. Also, when the objective function value is considered during the estimation process, it is observed that the proposed approach results in the lowest values for this for each instance.

References

Abdella, M., & Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. In *3rd International Conference on Computational Cybernetics, ICCC* (pp. 207–212). IEEE.

Feng, X., Zhang, Y., & Glass, J. R. (2014). Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *International Conference on Acoustic, Speech and Signal Processing (ICASSP)* (pp. 1759–1763). IEEE.

Finn, C., Tan, X., Duan, Y., Darrell, T., Levine, S., & Abbeel, P. (2016). Deep Spatial Autoencoders for Visuomotor Learning. In *International Conference on Robotics and Automation (ICRA)* (pp. 512–519). IEEE.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Hung, H. L., Chao, C. C., Cheng, C. H., & Huang, Y. F. (2010). Invasive weed optimization method based blind multiuser detection for MC-CDMA interference suppression over multipath fading channel. In *International Conference on Systems, Man and Cybernatics (SMC)* (pp. 2145–2150).

Jerez, J. M., Molina, I., García-Laencina, P. J., Alba, E., Ribelles, N., Martín, M., et al. (2010). Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine, 50*(2), 105–115. Elsevier.

Ju, Y., Guo, J., & Liu, S. (2015). A deep learning method combined sparse autoencoder with SVM. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 257–260). IEEE.

Kasdirin, H. A., Yahya, N. M., Aras, M. S. M., & Tokhi, M. O. (2017). Hybridizing invasive weed optimization with firefly algorithm for unconstrained and constrained optimization problems. *Journal of Theoretical and Applied Information Technology, 95*(4), 912–927.

Krizhevsky, A., & Hinton, G. E. (2011). Using very deep autoencoders for content based image retrieval. In *19th European Symposium on Artificial Neural Networks (ESANN),* 27–29 April 2011. Bruges, Belgium.

Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference in Swarm Intelligence* (pp. 259–270). Springer International Publishing.

Liew, A. W.-C., Law, N.-F., & Yan, H. (2011). Missing value imputation for gene expression data: Computational techniques to recover missing data from available information. *Briefings in Bioinformatics, 12*(5), 498–513. Oxford University Press.

Mehrabian, A. R., & Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics, 1,* 355–366.

Myers, T. A. (2011). Goodbye, listwise deletion: Presenting hot deck imputation as an easy and effective tool for handling missing data. *Communication Methods and Measures, 5*(4), 297–310. Taylor & Francis.

Paryzad, B., & Pour, N. S. (2013). Time-cost-quality trade-off in project with using invasive weed optimization algorithm. *Journal Basic and Applied Scientific Research, 3*(11), 134–142.

Rana, S., John, A. H., Midi, H., & Imon, A. (2015). Robust regression imputation for missing data in the presence of outliers. *Far East Journal of Mathematical Sciences, 97*(2), 183. Pushpa Publishing House.

Su, K., Ma, L., Guo, X., & Sun, Y. (2014). An efficient discrete invasive weed optimization algorithm for web services selection. *Journal of Software, 9*(3), 709–715.

Van Buuren, S. (2012). *Flexible imputation of missing data*. CRC press.

Veenhuis, C. (2010). Binary invasive weed optimization. In *Second World Congress on Nature and Biologically Inspired Computing* (pp. 449–454).

Yazdani, M., & Ghodsi, R. (2017). Invasive weed optimization algorithm for minimizing total weighted earliness and tardiness penalties on a single machine under aging effect. *International Robotics and Automation Journal, 2*(1), 1–5.

Zhang, S., Jin, Z., & Zhu, X. (2011). Missing data imputation by utilizing information within incomplete instances. *Journal of Systems and Software, 84*(3), 452–459. Elsevier.

Chapter 9
Missing Data Estimation Using Swarm Intelligence Algorithms from Reduced Dimensions

9.1 Auto-Associative Neural Network

Autoencoder networks are defined as networks that try to regenerate the inputs as outputs in the output layer (Lu and Hsu 2002), and this guarantees that the network can predict new input values as the outputs when presented with new inputs. These autoencoders are made up of one input layer and one output layer both having the same number of neurons, resulting in the term auto-associative (Lu and Hsu 2002). Besides these two layers, a narrow-hidden layer exists, and it is important that this layer contains fewer neurons than there are in the input and output layers with the aim of this being to apply encoding and decoding procedures when trying to find a solution to a given task (Mistry et al. 2008). These networks have been used in a variety of applications (Hines et al. 1998; Atalla and Inman 1998; Smauoi and Yakoob 2003; Marwala 2001, 2013; Marwala and Chakraverty 2006). The main concept that defines the operation of autoencoder networks is the notion that the mapping from the input to the output, $x^{(i)} \rightarrow y^{(i)}$ stores very important information and the key inherent architecture present in the input, $x^{(i)}$, that is contrarily hypothetical (Hines et al. 1998; Leke and Marwala 2016). An autoencoder takes x as the input and transcribes it into y, which is a hidden representation of the input, by making use of a mapping function, f_θ, that is deterministic. This function is expressed as (Leke and Marwala 2016; Isaacs 2014):

$$f_\theta(x) = s(Wx + b). \tag{9.1}$$

The parameter, θ, is made up of the weights W and biases b. s represents the sigmoidactivation function which is given by

$$s = \frac{1}{1 + e^{-x}}. \tag{9.2}$$

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_9

y is then mapped to a vector, z, representing the reconstruction of the inputs from this hidden representation. This reconstructed vector is obtained by using the following equations (Leke and Marwala 2016):

$$z = g_{\theta'}(y) = s\left(W'y + b'\right), \tag{9.3}$$

or

$$z = g_{\theta'}(y) = W'y + b'. \tag{9.4}$$

In the above equations, θ' is made up of the transpose of the weights matrix and the vector of biases from Eq. 9.1. Equation 9.3 is the autoencoder output function with a sigmoid transfer function (Eq. 9.2), while Eq. 9.4 is the linear output equation. After these operations, the network can then be fine-tuned by applying a supervised learning approach (Isaacs 2014). The network is said to have tied weights when the weights matrix is transposed. The vector z is not described as an exacting transformation of x, but rather as the criteria of a distribution $p(X|Z = z)$ in probabilistic terms. The hope is these criteria will result in x with high probability (Isaacs 2014). The resulting equation is as follows (Leke and Marwala 2016):

$$p(X|Y = y) = p(X|Z = g_{\theta'}(y)). \tag{9.5}$$

This leads to an associated regeneration disparity that forms the basis for the objective function to be used by the optimization algorithm. This disparity is usually represented by (Leke and Marwala 2016)

$$L(x, z) \propto -\log p(x|z). \tag{9.6}$$

α indicates proportionality. The equation above could be represented by (Sartori et al. 2005)

$$\delta_{AE}(\theta) = \sum_{t} L\left(x^{(t)}, g_\theta\left(f_\theta\left(x^{(t)}\right)\right)\right). \tag{9.7}$$

Autoencoder networks have been used in a variety of application areas, by several researchers, with the focus being on the problem of missing data (Abdella and Marwala 2005; Baek and Cho 2003; Tim et al. 2004; Brain et al. 2006).

9.2 Missing Data Estimation Model

In this section, we describe the approach that combines the advantages of autoencoder networks and the optimization algorithms. When applying a network of any nature to estimate missing data, it is very important that the network model be very accurate.

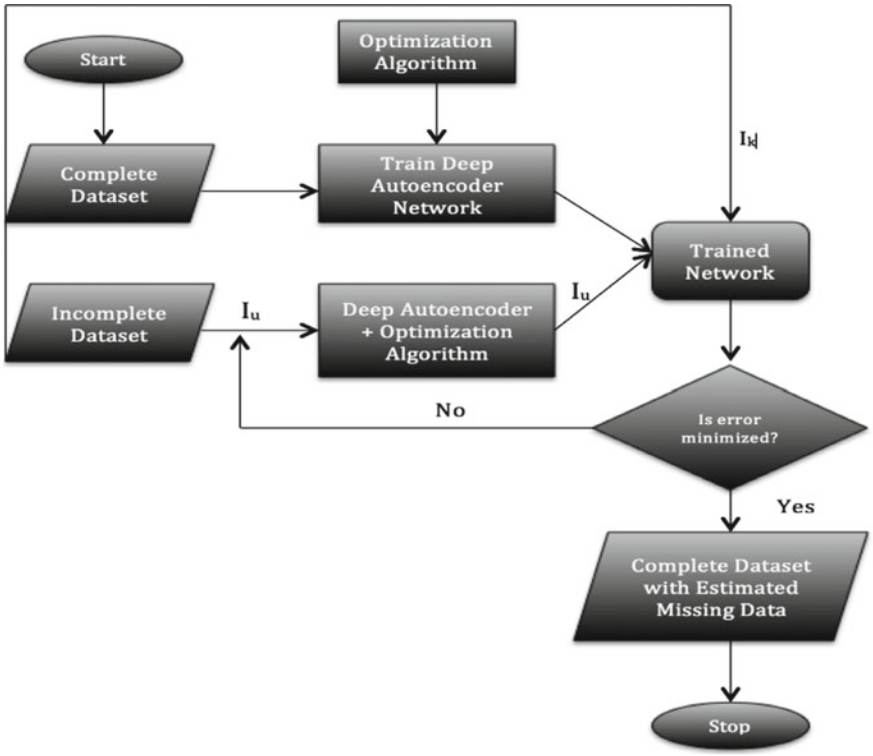

Fig. 9.1 Flow chart representation of estimation model. © 2017 IEEE. Reproduced with permission from Leke et al. (2017)

The proposed methodology is summarized in Fig. 9.1, where I_k and I_u represent the known and unknown features, respectively, and make up the set of inputs.

First, the network is trained using the scaled conjugate gradient (SCG) algorithm, with a complete set of records to identify a mapping function, f, that extracts and stores the interrelationships and correlations between features in the dataset. This leads to the output, $\vec{O}$, obtained by (Leke and Marwala 2016):

$$\vec{O} = f\left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}\right). \tag{9.8}$$

Due to the fact that the autoencoder framework tries to represent the input as the output of the network, the output is approximately equal to the input. This results in a small error which relies upon the accuracy of the autoencoder. The error is expressed as (Leke and Marwala 2016):

$$\vec{e} = \begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}, \vec{w}\right). \tag{9.9}$$

To guarantee that the error is always positive, Eq. 9.9 is squared, yielding the following equation:

$$\vec{e} = \left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}, \vec{w}\right)\right)^2. \tag{9.10}$$

The error will be minimized when the output is as close as possible to the input. This predominantly occurs when the data presented to the framework has similar characteristics as those recorded during training. Therefore, during the estimation procedure, minimizing the error symbolizes the optimization of the missing data in such a way that the new data matrix matches the pattern yielded by the complete dataset during training. The optimization algorithms are used to further optimize the network weights, because they are guaranteed to attain a global optimum point, thereby making sure the network is as accurate as possible. The algorithms as optimization techniques try to attain a global optimum point which in the landscape of the error function, being the mean squared error, is the lowest point or valley of the function. This procedure is expressed mathematically as (Leke and Marwala 2016):

$$\underset{\vec{w}}{\text{minimize}} \left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix} - f\left(\begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}, \vec{w}\right)\right)^2. \tag{9.11}$$

The incomplete dataset is passed as input to the optimization algorithms which have as part of the objective function, the optimized neural network. The missing data is estimated and together with the known values, run through the trained network to observe whether the network error has been minimized, without any changes for a few iterations. If it has, the dataset is returned with imputed values; otherwise, the imputation procedure is executed again. The stopping criteria for the missing data estimation procedure using the optimization algorithms are either 40000 function evaluations having been executed, or there being no change in the objective function value.

From the above description of how the autoencoder and the optimization algorithms are used, the equation below summarizes the hybrid function of the proposed approaches, with f_{OA} being the optimization algorithm data estimation operation and f_{AE} being the function of the autoencoder.

$$y = f_{AE}\left(W, f_{\text{OA}}\left(\vec{I}\right)\right), \tag{9.12}$$

where $\vec{I} = \begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}$ represent the input space of known and unknown features. This equation represents the model design whereby the complete input vector with known and estimated missing data entries obtained by executing the missing data estimation procedure (f_{OA}) is presented to the regression model (f_{AE}) to observe whether the network error has been minimized. If it is found that the error has been minimized, the output, y, will contain the known input vector values and the optimal missing data estimated values.

9.3 Experimental Results

To investigate and validate the proposed models against other existing approaches, the Mixed National Institute of Standards and Technology (MNIST) dataset of handwritten digits (LeCun 2016) was used. It consists of 60,000 training samples and 10,000 test samples, each with 784 features representing the individual pixel values of the image. The black pixels are represented by 0, while the white pixels have values of 255; as such, all the variables could be considered as being continuous. The data in its current form has been cleaned of outliers; therefore, each image used in the dataset should contribute to generalizing the system. The data are normalized to being in the range [0, 1] and randomized to improve the network performance. The training set of data is further split into training and validation sets (50000 and 10000, respectively).

The neural network architecture is optimized using the validation set of data with the correctness in performance of the proposed systems being tested using the test data. The optimized deep network architecture used is 784-1000-500-250-30-250-500-1000-784, verified by performing cross-validations using the validation set of data, with 784 input and output nodes, and seven hidden layers with 1000, 500, 250, 30, 250, 500 and 1000 nodes, respectively. From this network, the bottleneck layer outputs were used to conduct the experiments in this chapter. The narrow network used as a result comprised of 30 input nodes, 15 nodes in the one hidden layer and 30 nodes in the output layer. It is similar in shape to that presented in Fig. 9.2.

The data were still normalized to be in the range [0, 1]. The sigmoid and linear activation functions were used for the hidden and output layers of the narrow network, respectively. The training is done using the scaled conjugate gradient (SCG) algorithm. 100 samples are selected at random from the test set of data and features are then removed respecting the missing at random (MAR) and missing completely at random (MCAR) mechanisms as well as the arbitrary missing data pattern. This is done by creating a binomial matrix of the same size as the test data with zeros and ones adhering to the stated mechanisms and pattern, and replacing every occurrence of one with NaN (implying missingness). Also, these datasets are created with 1, 5 and 10% of missing data. The datasets obtained from this, with missing values are used to test the performance of the missing data estimation schemes, which are

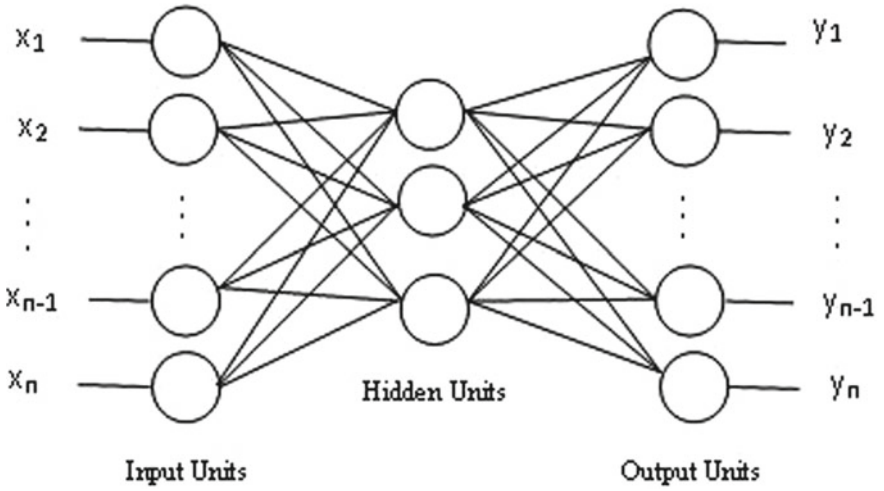

Fig. 9.2 Autoencoder network

then compared against existing methods. Furthermore, we will verify whether reconstructing the images from the complete datasets generated by these approaches yield accurate images that can be used for classification tasks.

The effectiveness of the proposed approaches was estimated using the mean squared error (MSE) and the root mean squared logarithmic error (RMSLE). Also used are the mean absolute error (MAE) and the standard error (SE). The mean squared and root mean squared logarithmic errors as well as the global deviation yield measures of the difference between the actual and predicted values, and provides an indication of the capability of the estimation.

$$MSE = \frac{\sum_{i=1}^{n}\left(I_i - \hat{I}_i\right)^2}{n}, \tag{9.13}$$

$$RMSLE = \sqrt{\frac{\sum_{i=1}^{n}\left(\log(I_i + 1) - \log\left(\hat{I}_i + 1\right)\right)^2}{n}}, \tag{9.14}$$

and

$$GD = \left(\frac{\sum_{i=1}^{n}\left(\hat{I}_i - I_i\right)}{n}\right)^2, \tag{9.15}$$

where in Eqs. 9.13–9.15, n is the number of samples while I and $\hat{I}$ represent the real test set values and predicted missing output values, respectively. In this analysis, the approaches will be compared against existing approaches in the literature (MLP-

PSO, MLP-SA and MLP-GA (Abdella and Marwala 2005; Leke et al. 2014). The results are grouped in figures and tables below.

9.3.1 Ant Colony Optimization Results

The mean squared error is given in Fig. 9.3. It shows 6.08% of error for MLP-ACO while MLP-PSO, MLP-SA and MLP-GA yielded 6.08, 24.1 and 24.1% error for 1% of missing data, respectively. It depicts 3.35, 3.35, 23.53 and 23.53% of error for the MLP-ACO, MLP-PSO, MLP-SA and MLP-GA approaches for 5% of missing data, respectively. Furthermore, it shows 6.17, 6.19, 22.76 and 22.76% of error for the MLP-ACO, MLP-PSO, MLP-SA and MLP-GA approaches for 10% of missing data, respectively.

Figure 9.4 depicts the root mean squared logarithmic error for the approaches analysed, including the MLP-ACO approach. It confirms the better performance of the MLP-ACO approach when compared to the other approaches. It can be observed that it is on par in terms of performance with the MLP-PSO approach. The MLP-ACO exhibits 18.26% of RMSLE while we obtained 18.26% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 39.83 and 39.83% for 1% of missing data, respectively. Furthermore, the MLP-ACO exhibits 11.95% of RMSLE while we obtained 11.95% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 39.03 and 39.03% for 5% of missing data, respectively. Moreover, the MLP-ACO exhibits 16.64% of RMSLE while we obtained 16.66% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 38.33 and 38.33% for 10% of missing data, respectively.

In Fig. 9.5, the GD values of the approaches are revealed. We observe that the MLP-ACO yields a GD value of 2% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 2, 23.79 and 23.79% for 1% of missing data, respectively. In

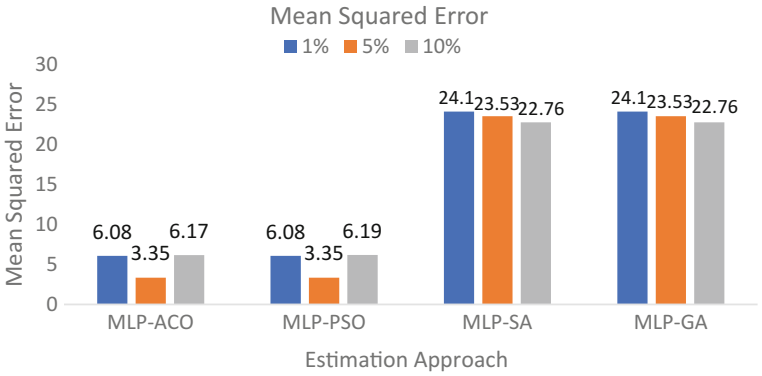

Fig. 9.3 Mean squared error versus estimation approach

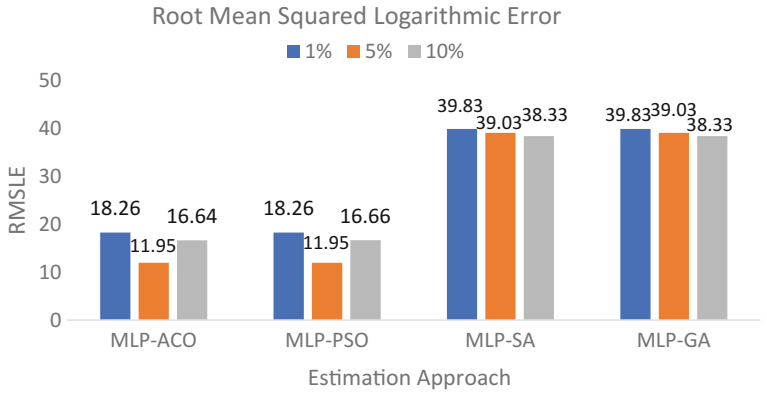

Fig. 9.4 Root mean squared logarithmic error versus estimation approach

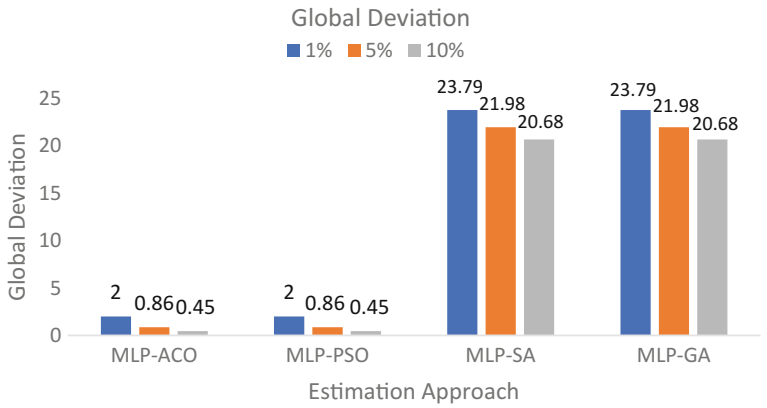

Fig. 9.5 Global deviation versus estimation approach

addition, we observe that the MLP-ACO yields a GD value of 0.86% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.86, 21.98 and 21.98% for 5% of missing data, respectively. Moreover, we observe a GD value of 0.45% for the MLP-ACO compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.45, 20.68 and 20.68% for 10% of missing data, respectively.

9.3.2 Ant-Lion Optimizer Results

The mean squared error is given in Fig. 9.6. It shows 4.4% of error for MLP-ALO while MLP-PSO, MLP-SA and MLP-GA yielded 4.4, 17.42 and 17.42% error for 1% of missing data, respectively. It depicts 4.82, 4.83, 26.65 and 26.65% of error for

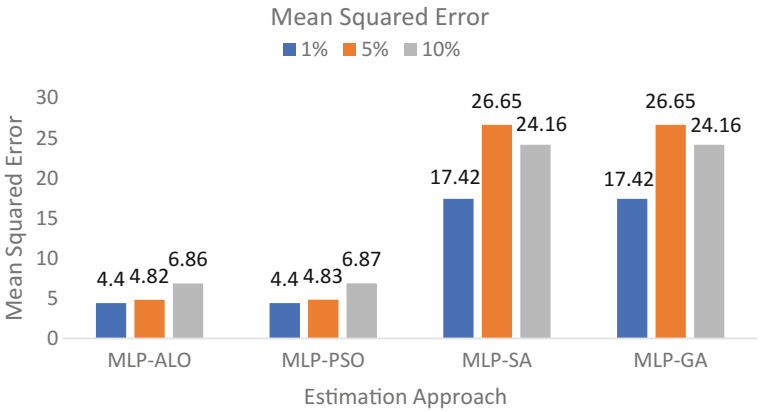

Fig. 9.6 Mean squared error versus estimation approach

the MLP-ALO, MLP-PSO, MLP-SA and MLP-GA approaches for 5% of missing data, respectively. Furthermore, it shows 6.86, 6.87, 24.16 and 24.16% of error for the MLP-ALO, MLP-PSO, MLP-SA and MLP-GA approaches for 10% of missing data, respectively.

Figure 9.7 depicts the root mean squared logarithmic error for the approaches analysed, including the MLP-ALO approach. It confirms the better performance of the MLP-ALO approach when compared to the other approaches. It can be observed that it is on par in terms of performance with the MLP-PSO approach. The MLP-ALO exhibits 13.38% of RMSLE while we obtained 13.38% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 34.61 and 34.61% for 1% of missing data, respectively. Furthermore, the MLP-ALO exhibits 14.98% of RMSLE while we obtained 15% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 41.12 and 41.12% for 5% of missing data, respectively. Moreover, the MLP-ACO exhibits 17.66% of RMSLE while we obtained 17.66% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 39.18 and 39.18% for 10% of missing data, respectively.

In Fig. 9.8, the GD values of the approaches are revealed. We observe that the MLP-ALO yields a GD value of 2.2% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 2.2, 16.71 and 16.71% for 1% of missing data, respectively. In addition, we observe that the MLP-ALO yields a GD value of 0.1% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.1, 25.01 and 25.01% for 5% of missing data, respectively. Moreover, we observe a GD value of 0.29% for the MLP-ACO compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.29, 21.84 and 21.84% for 10% of missing data, respectively.

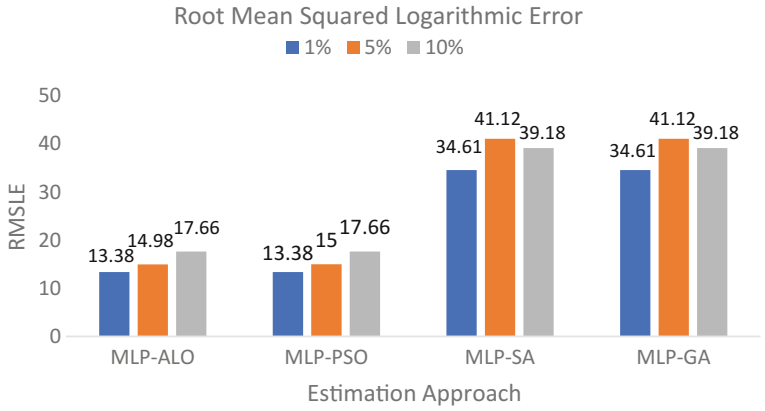

Fig. 9.7 Root mean squared logarithmic error versus estimation approach

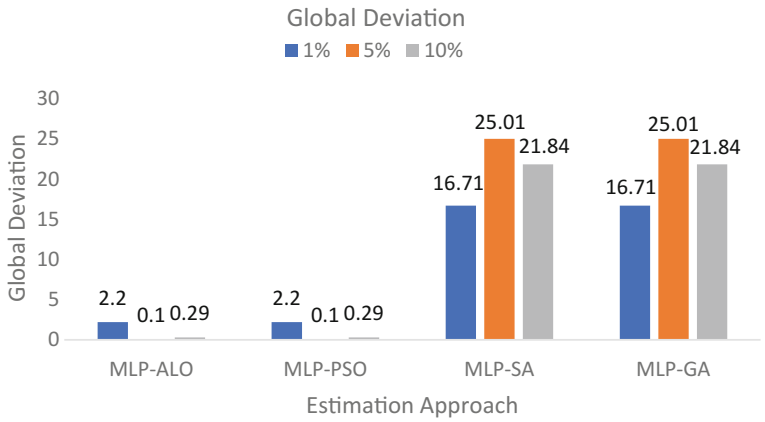

Fig. 9.8 Global deviation versus estimation approach

9.3.3 Bat Algorithm Results

The mean squared error is given in Fig. 9.9. It shows 7.79% of error for MLP-BAT while MLP-PSO, MLP-SA and MLP-GA yielded 7.79, 17.61 and 17.61% error for 1% of missing data, respectively. It depicts 5.21, 4.52, 23.19 and 23.19% of error for the MLP-BAT, MLP-PSO, MLP-SA and MLP-GA approaches for 5% of missing data, respectively. Furthermore, it shows 6.86, 6.87, 24.16 and 24.16% of error for the MLP-BAT, MLP-PSO, MLP-SA and MLP-GA approaches for 10% of missing data, respectively.

Figure 9.10 depicts the root mean squared logarithmic error for the approaches analysed, including the MLP-BAT approach. It confirms the better performance of the MLP-BAT approach when compared to the other approaches. The MLP-BAT

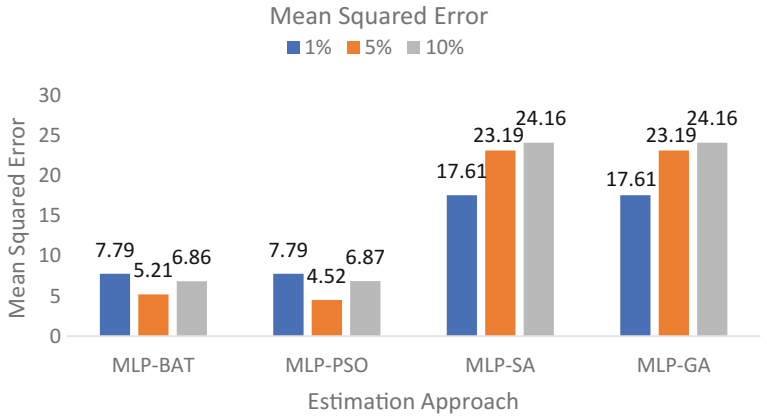

Fig. 9.9 Mean squared error versus estimation approach

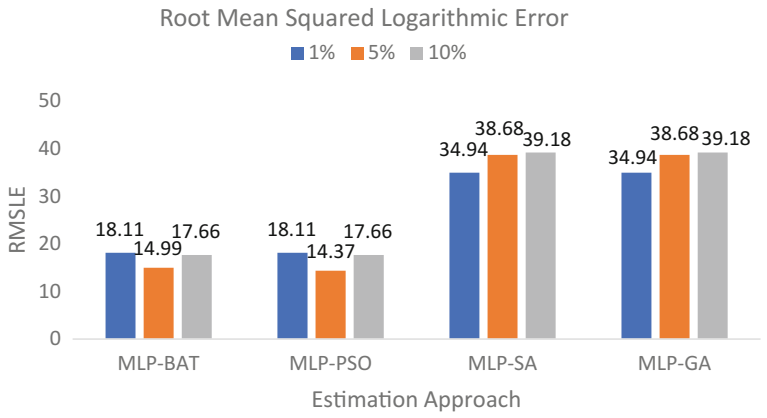

Fig. 9.10 Root mean squared logarithmic error versus estimation approach

exhibits 18.11% of RMSLE while we obtained 18.11% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 34.94 and 34.94% for 1% of missing data, respectively. Furthermore, the MLP-BAT exhibits 14.99% of RMSLE while we obtained 14.37% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 38.68 and 38.68% for 5% of missing data, respectively. Moreover, the MLP-BAT exhibits 17.66% of RMSLE while we obtained 17.66% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 39.18 and 39.18% for 10% of missing data, respectively.

In Fig. 9.11, the GD values of the approaches are revealed. We observe that the MLP-BAT yields a GD value of 2.9% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 2.9, 17.32 and 17.32% for 1% of missing data, respectively. In addition, we observe that the MLP-BAT yields a GD value of 0.65%

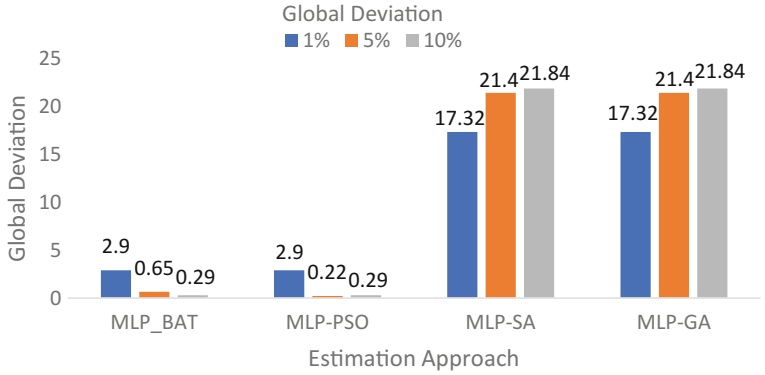

Fig. 9.11 Global deviation versus estimation approach

compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.22, 21.4 and 21.4% for 5% of missing data, respectively. Moreover, we observe a GD value of 0.29% for the MLP-BAT compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.29, 21.84 and 21.84% for 10% of missing data, respectively.

9.3.4 Cuckoo Search Algorithm Results

The mean squared error is given in Fig. 9.12. It shows 9.02% of error for MLP-CS while MLP-PSO, MLP-SA and MLP-GA yielded 9.02, 27.21 and 27.21% error for 1% of missing data, respectively. It depicts 5.96, 5.98, 29.61 and 29.61% of error for the MLP-CS, MLP-PSO, MLP-SA and MLP-GA approaches for 5% of missing data, respectively. Furthermore, it shows 6.89, 6.89, 22.36 and 22.36% of error for the MLP-CS, MLP-PSO, MLP-SA and MLP-GA approaches for 10% of missing data, respectively.

Figure 9.13 depicts the root mean squared logarithmic error for the approaches analysed, including the MLP-CS approach. It confirms the better performance of the MLP-CS approach when compared to the other approaches. The MLP-CS exhibits 19.6% of RMSLE while we obtained 19.61% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 40.77 and 40.77% for 1% of missing data, respectively. Furthermore, the MLP-CS exhibits 16.32% of RMSLE while we obtained 16.35% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 42.73 and 42.73% for 5% of missing data, respectively. Moreover, the MLP-CS exhibits 17.71% of RMSLE while we obtained 17.71% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 38.13 and 38.13% for 10% of missing data, respectively.

In Fig. 9.14, the GD values of the approaches are revealed. We observe that the MLP-CS yields a GD value of 6.14% compared to the MLP-PSO, MLP-SA and MLP-

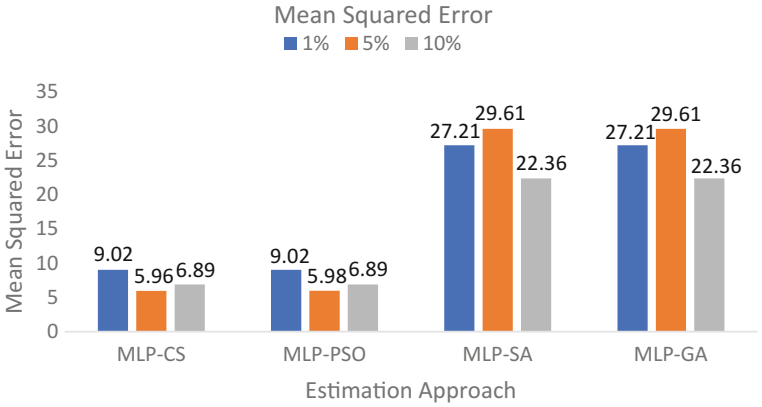

Fig. 9.12 Mean squared error versus estimation approach

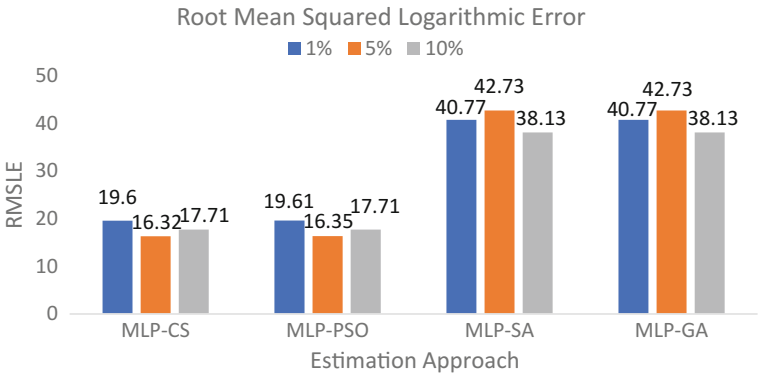

Fig. 9.13 Root mean squared logarithmic error versus estimation approach

GA yielding values of 6.14, 23.47 and 23.47% for 1% of missing data, respectively. In addition, we observe that the MLP-CS yields a GD value of 0.0029% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.0031, 27.06 and 27.06% for 5% of missing data, respectively. Moreover, we observe a GD value of 0.62% for the MLP-CS compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.63, 20.6 and 20.6% for 10% of missing data, respectively.

9.3.5 Firefly Algorithm Results

The mean squared error is given in Fig. 9.15. It shows 6.36% of error for MLP-FA while MLP-PSO, MLP-SA and MLP-GA yielded 6.37, 25.94 and 25.94% error for 1% of missing data, respectively. It depicts 4.17, 4.17, 20 and 20% of error for the

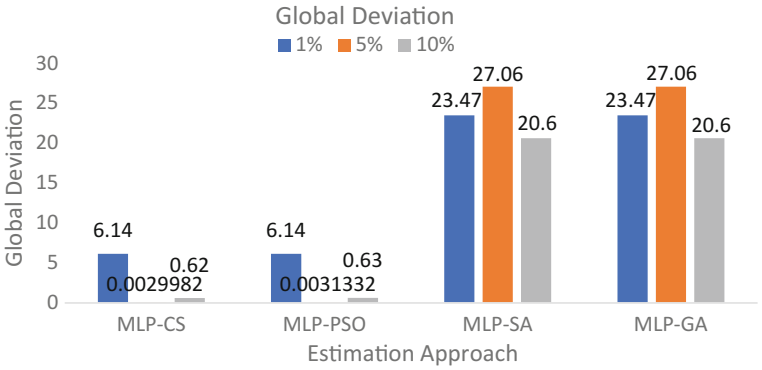

Fig. 9.14 Global deviation versus estimation approach

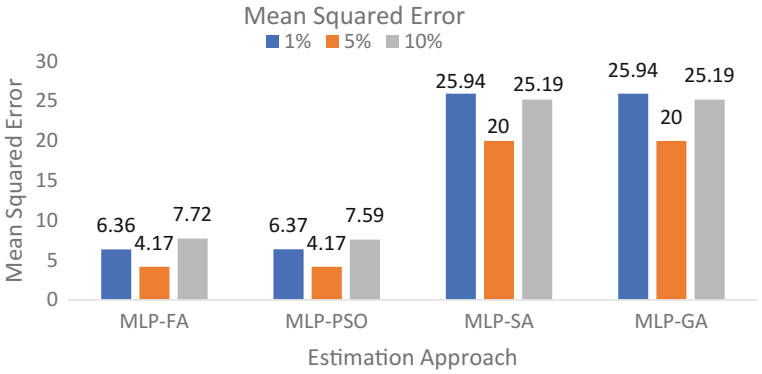

Fig. 9.15 Mean squared error versus estimation approach

MLP-FA, MLP-PSO, MLP-SA and MLP-GA approaches for 5% of missing data, respectively. Furthermore, it shows 7.72, 7.59, 25.19 and 25.19% of error for the MLP-FA, MLP-PSO, MLP-SA and MLP-GA approaches for 10% of missing data, respectively.

Figure 9.16 depicts the root mean squared logarithmic error for the approaches analysed, including the MLP-FA approach. It confirms the better performance of the MLP-FA approach when compared to the other approaches. The MLP-FA exhibits 16.7% of RMSLE while we obtained 16.7% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 40.84 and 40.84% for 1% of missing data, respectively. Furthermore, the MLP-FA exhibits 13.89% of RMSLE while we obtained 13.89% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 36.5 and 36.5% for 5% of missing data, respectively. Moreover, the MLP-FA exhibits 19.08% of RMSLE while we obtained 18.88% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 39.81 and 39.81% for 10% of missing data, respectively.

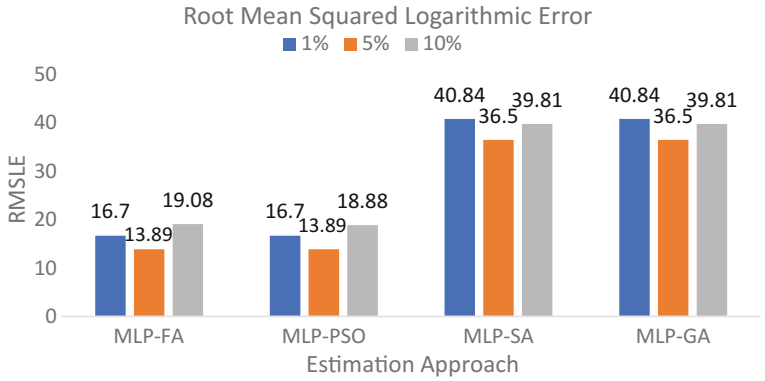

Fig. 9.16 Root mean squared logarithmic error versus estimation approach

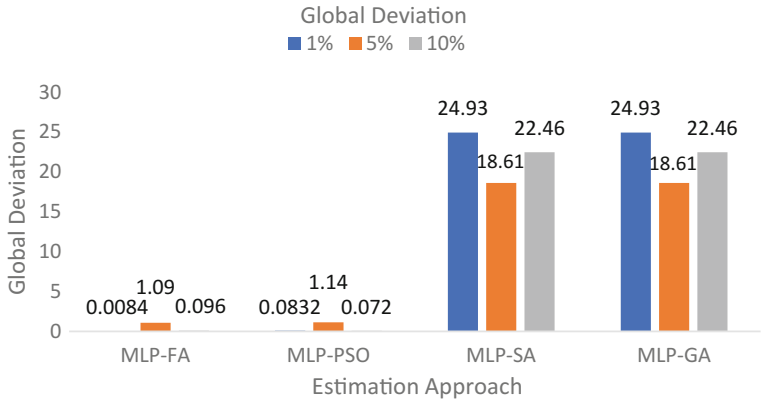

Fig. 9.17 Global deviation versus estimation approach

In Fig. 9.17, the GD values of the approaches are revealed. We observe that the MLP-FA yields a GD value of 0.0084% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.0832, 24.93 and 24.93% for 1% of missing data, respectively. In addition, we observe that the MLP-FA yields a GD value of 1.09% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 1.14, 18.61 and 18.61% for 5% of missing data, respectively. Moreover, we observe a GD value of 0.096% for the MLP-FA compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.072, 22.46 and 22.46% for 10% of missing data, respectively.

9.3.6 *Invasive Weed Optimization Results*

The mean squared error is given in Fig. 9.18. It shows 5.86% of error for MLP-IWO while MLP-PSO, MLP-SA and MLP-GA yielded 5.86, 33.4 and 33.4% error for 1% of missing data, respectively. It depicts 6.32, 6.33, 21.67 and 21.67% of error for the MLP-IWO, MLP-PSO, MLP-SA and MLP-GA approaches for 5% of missing data, respectively. Furthermore, it shows 6.21, 6.21, 24.99 and 24.99% of error for the MLP-IWO, MLP-PSO, MLP-SA and MLP-GA approaches for 10% of missing data, respectively.

Figure 9.19 depicts the root mean squared logarithmic error for the approaches analysed, including the MLP-IWO approach. It confirms the better performance of the MLP-IWO approach when compared to the other approaches. The MLP-IWO exhibits 15.66% of RMSLE while we obtained 15.66% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 44.93 and 44.93% for 1% of missing data, respectively. Furthermore, the MLP-IWO exhibits 17.36% of RMSLE while we obtained 17.37% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 37.84 and 37.84% for 5% of missing data, respectively. Moreover, the MLP-IWO exhibits 17.09% of RMSLE while we obtained 17.09% of RMSLE for MLP-PSO as well. MLP-SA and MLP-GA show RMSLE values of 39.86 and 39.86% for 10% of missing data, respectively.

In Fig. 9.20, the GD values of the approaches are revealed. We observe that the MLP-IWO yields a GD value of 0.43% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.43, 31.15 and 31.15% for 1% of missing data, respectively. In addition, we observe that the MLP-IWO yields a GD value of 0.98% compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.99, 20.28 and 20.28% for 5% of missing data, respectively. Moreover, we observe a GD value of 0.0929% for the MLP-IWO compared to the MLP-PSO, MLP-SA and MLP-GA yielding values of 0.0939, 22.79 and 22.79% for 10% of missing data, respectively.

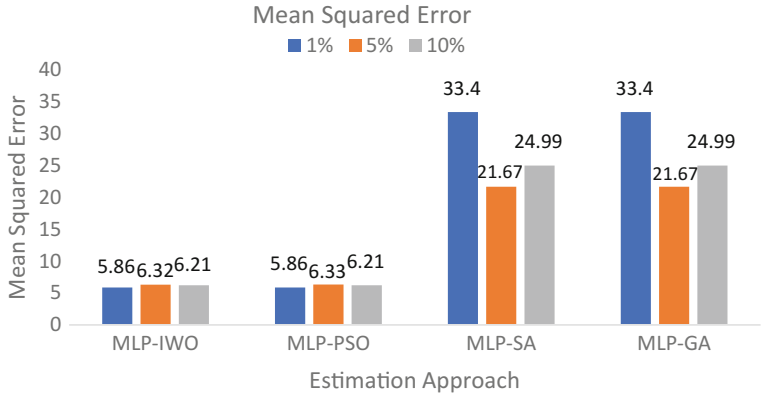

Fig. 9.18 Mean squared error versus estimation approach

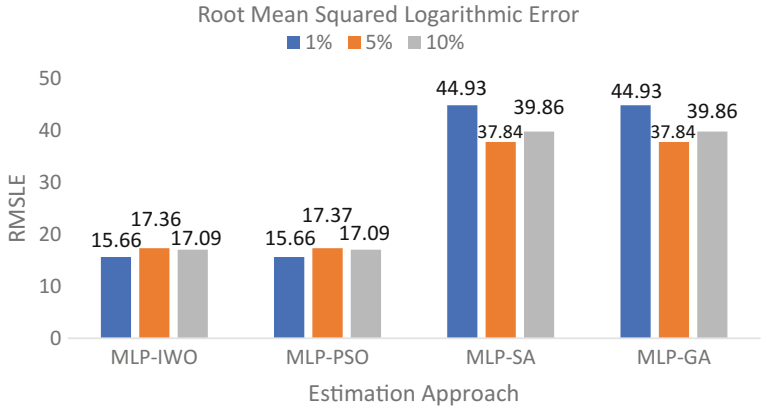

Fig. 9.19 Root mean squared logarithmic error versus estimation approach

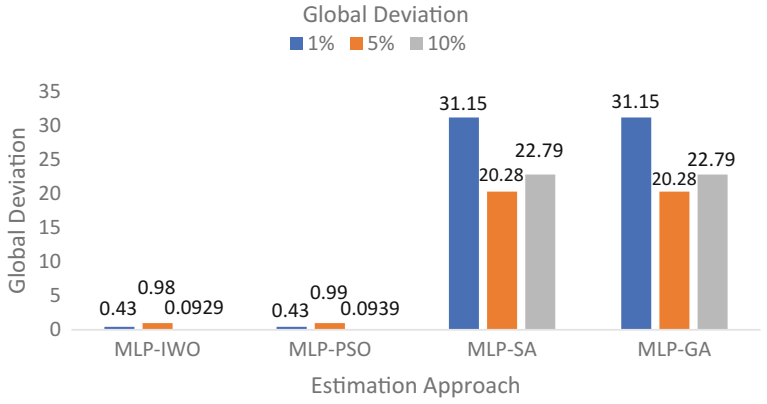

Fig. 9.20 Global deviation versus estimation approach

9.4 Conclusion

This chapter presented an investigation of the estimation of missing data from reduced feature dimensions. The outputs from the bottleneck layer were used as the data for the experiments in this chapter. The approaches presented were tested for 1, 5 and 10% of missingness in the dataset. It was observed that the MLP-PSO method performed on par with the new approaches proposed.

References

Abdella, M., & Marwala, T. (2005). *The use of genetic algorithms and neural networks to approximate missing data in database* (Vol. 24, pp. 577–589).

Atalla, M. J., & Inman, D. J. (1998). On model updating using neural networks. *Mechanical Systems and Signal Processing, 12,* 135–161.

Baek, K., & Cho, S. (2003). Bankruptcy prediction for credit risk using an auto-associative neural network in Korean firms. In *IEEE Conference on Computational Intelligence for Financial Engineering* (pp. 25–29). Hong Kong, China.

Brain, L. B., Marwala, T., & Tettey, T. (2006). Autoencoder networks for HIV classification. *Current Science, 91*(11), 1467–1473.

Hines, J. W., Robert, E. U., & Wrest, D. J. (1998). Use of autoassociative neural networks for signal validation. *Journal of Intelligent and Robotic Systems, 21*(2), 143–154.

Isaacs, J. C. (2014). Representational learning for sonar ATR. In *SPIE Defense+Security*. International Society for Optics and Photonics.

Leke, C., Twala, B., & Marwala, T. (2014). Modeling of missing data prediction: Computational intelligence and optimization algorithms. In *International Conference on Systems, Man and Cybernetics (SMC)* (pp. 1400–1404).

LeCun, Y. (2016). *The MNIST database of handwritten digits*. Retrieved January 1, 2016, from http://yann.lecun.com/exdb/mnist/.

Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference in Swarm Intelligence* (pp. 259–270) Springer International Publishing.

Leke, C., Ndjiongue, A. R., Twala, B., & Marwala, T. (2017). Deep learning-bat high-dimensional missing data estimator. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 483–488). IEEE.

Lu, P. J., & Hsu, T. C. (2002). Application of autoassociative neural network on gas-path sensor data validation. *Journal of Propulsion and Power, 18*(4), 879–888.

Marwala, T. (2001). Probabilistic fault identification using a committee of neural networks and vibration data. *Journal of Aircraft, 38*(1), 138–146.

Marwala, T., & Chakraverty, S. (2006). Fault classification in structures with incomplete measured data using autoassociative neural networks and genetic algorithm. *Current Science, 90*(4), 542–548.

Marwala, T. (2013). *Economic modelling using artificial intelligence methods*. London: Springer.

Mistry, J., Nelwamondo, F., & Marwala, T. (2008). Estimating missing data and determining the confidence of the estimate data. In *Seventh International Conference on Machine Learning and Applications* (pp. 752–755). San Diego, CA, USA.

Sartori, N., Salvan, A., & Thomaseth, K. (2005). Multiple imputation of missing values in a cancer mortality analysis with estimated exposure dose. *Computational Statistics & Data Analysis, 49*(3), 937–953.

Smauoi, N., & Al-Yakoob, S. (2003). Analyzing the dynamics of cellular flames using Karhunen-Loeve decomposition and autoassociative neural networks. *Society for Industrial and Applied Mathematics, 24,* 1790–1808.

Tim, T., Mutajogire, M., & Marwala, T. (2004). Stock market prediction using evolutionary neural networks. In *Fifteenth Annual Symposium of the Pattern Recognition, PRASA* (pp. 123–133).

Chapter 10
Deep Learning Framework Analysis

10.1 Introduction

Datasets nowadays such as those that record production, manufacturing and medical data may suffer from the problem of missing data at different phases of the data collection and storage processes. Faults in measuring instruments or data transmission lines are predominant causes of missing data. The occurrence of missing data results in difficulties in decision-making and analysis tasks which rely on access to complete and accurate data, resulting in data estimation techniques which are not only accurate but also efficient. Several methods exist as a means to alleviate the problems presented by missing data ranging from deleting records with missing attributes (listwise and pairwise data deletion) to approaches that employ statistical and artificial intelligence methods such as those presented in the next paragraph. The problem though is, some of the statistical and naive approaches more often than not produce biased approximations, or they make false assumptions about the data and correlations within the data. These have an adverse effect on the decision-making processes which are data dependent.

The applications of missing data estimation techniques are vast and with that said, the existing methods depend on the nature of the data, the pattern of missingness and are predominantly implemented on low-dimensional datasets. Abdella and Marwala (2005) implemented a joint neural network-genetic algorithm framework to impute missing values in one dimension. Leke et al. (2014) used a combination of particle swarm optimization and simulated annealing and genetic algorithm with a multilayer perceptron (MLP) autoencoder network yielding good results in one dimension. Vukosi et al. (2007) used the MLP autoencoder networks, principal component analysis and support vector machines in combination with the genetic algorithm to impute missing variables. In Ssali and Marwala (2007), a decision tree is combined with the MLP autoencoder network to estimate missing data in datasets yielding increased accuracy. In papers such as Zhang (2011), Jerez et al. (2010), Liew et al. (2011), Myers (2011), Schafer and Graham (2002) and Van Buuren (2012),

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_10

new techniques to impute missing data and comparisons between these and existing methods are observed. Recently in Leke and Marwala (2016), a technique involving a deep network framework and a swarm intelligence algorithm was proposed to handle missing data in high-dimensional datasets, a first of its nature, with promising outcomes. It is because of this work that we conduct the research in this chapter to investigate the possibility of using a deep neural network of a similar nature with fewer hidden layers to observe whether the accuracy obtained can be improved upon or even equalled.

Investigating new techniques to address the previously mentioned drawbacks led to the implementation of a deep autoencoder, which is by definition an unsupervised learning technique that tries to recall the input space by learning an approximation function which diminishes the disparity between the original data, x, and the reconstructed data, $\hat{x}$, the same as with the principal component analysis (PCA).

A deep autoencoder comprises two symmetrical deep belief networks with a couple of shallow layers representing the encoding section of the network, while the second set of shallow layers represent the decoding section, with each of the individual layers being a restricted Boltzmann machine (RBM). The RBMs are stacked together to form the overall deep autoencoder network which is subsequently fine-tuned using the back-propagation algorithm. These networks are applicable in a variety of sectors; for example, deep autoencoders were used for visuomotor learning by Finn et al. (2016) while Ju et al. (2015) used deep autoencoder networks with support vector machines (SVMs) to learn sparse features and perform classification tasks. Brain et al. (2006) used an autoencoder for HIV classification, and in Krizhevsky and Hinton (2011), these networks were used to map small colour images to short binary codes. Autoencoders have several advantages such as being more flexible by introducing non-linearity in the encoding part of the network contrary to the likes of the PCA, which is a key property of the pretraining procedure, they require no prior knowledge of the data properties and correlations, and also, they are intuitive. The main drawback of this network is its need for a significant number of samples for training and learning, which are not always readily available.

10.2 Missing Data Estimation Model

In this section, we present information on the model which uses a combination of a deep learning regression model, a deep autoencoder network and optimization techniques, being the ant colony optimization, ant-lion optimizer, bat, cuckoo search, firefly and invasive weed optimization algorithms, to approximate the missing data. Figure 10.1 illustrates how the regression model and the optimization algorithms will be used.

Two predominant features of an autoencoder being: (i) its auto-associative nature and (ii) the butterfly-like structure of the network resulting from a bottleneck trait in the hidden layers were the reasons behind the network being used. Autoencoders are also ideal courtesy of their ability to replicate the input data by learning certain

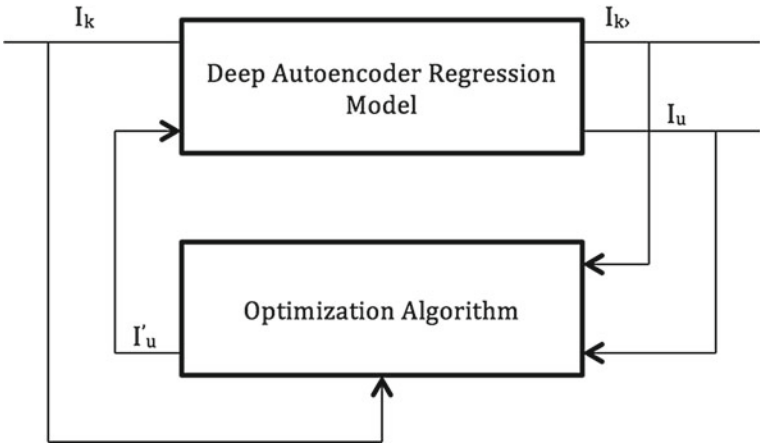

Fig. 10.1 Data imputation configuration. © 2017 Springer. Reproduced with permission from Leke et al. (2017)

linear and non-linear correlations and covariances present in the input space, by projecting the input data into lower dimensions. The only condition required is that the hidden layer(s) have fewer nodes than the input layer, though it is dependent on the application. Prior to optimizing the regression model parameters, it is necessary to identify the network structure where the structure depends on the number of layers, the number of hidden units per hidden layer, the activation functions used and the number of input and output variables. After this, the parameters can then be approximated using the training set of data. The parameter approximation procedure is performed for a given number of training cycles, with the optimal number obtained by analysing the validation error. The aim of this was to avoid overtraining the network and to use the fastest training approach without compromising on accuracy. The optimal number of training cycles was found to be 500. The training procedure estimates weight parameters such that the network output is as close as possible to the target output.

The optimization algorithms are used to estimate the missing values by optimizing an objective function which has as part of it the trained network. They will use values from the population as part of the input to the network, and the network will recall these values and they will form part of the output. The complete data matrix containing the estimated values and observed values will be fed into the autoencoder as input. Some inputs are known, with others unknown and they will be estimated using the regression method and the optimization algorithms as described at the beginning of the paragraph. The symbols I_k and I_u as used in Figs. 10.1 and 10.2 represent the known and unknown/missing values, respectively.

Considering that the approach makes use of a deep autoencoder, it is imperative that the autoencoder architecture match the output to the input. This trait is expected

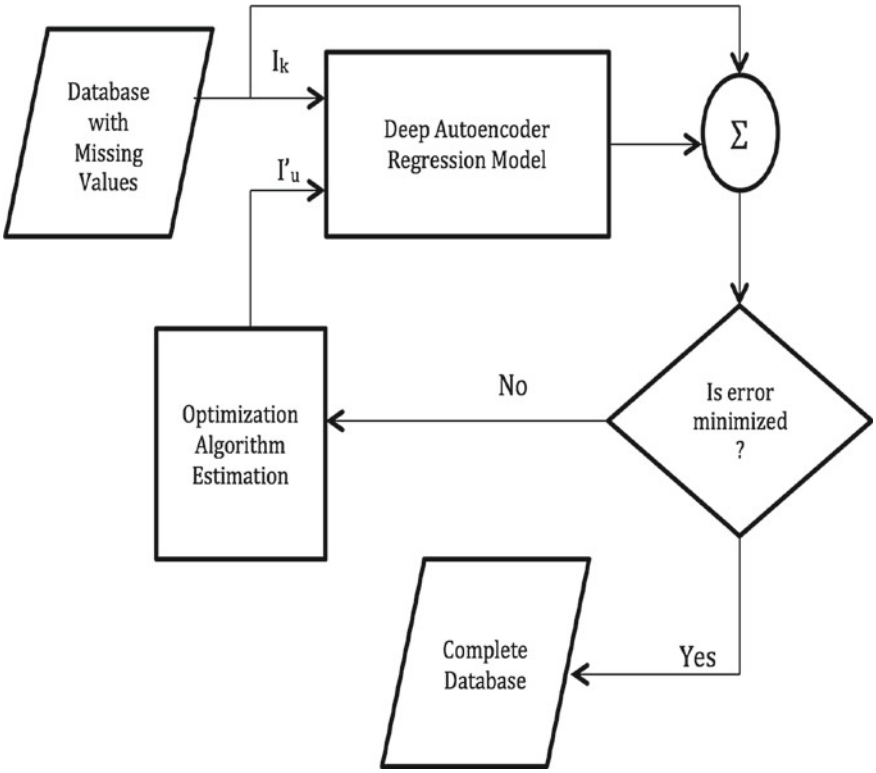

Fig. 10.2 Deep autoencoder and optimization algorithms missing data estimator structure. © 2017 Springer. Reproduced with permission from Leke et al. (2017)

when a dataset with familiar correlations recorded in the network is used. The error used is the disparity between the target output and the network output, expressed as

$$\delta = \vec{I} - f\left(\vec{W}, \vec{I}\right), \tag{10.1}$$

where $\vec{I}$ and $\vec{W}$ represent the inputs and the weights, respectively.

The square of Eq. 10.1 is used to always guarantee that the error is positive. This results in the following equation:

$$\delta = \left(\vec{I} - f\left(\vec{W}, \vec{I}\right)\right)^2. \tag{10.2}$$

Courtesy of the fact that the input and output vectors contain both I_k and I_u, the error function is rewritten as

$$\delta = \left(\begin{bmatrix} I_k \\ I_u \end{bmatrix} - f\left(\begin{Bmatrix} I_k \\ I_u \end{Bmatrix}, w \right) \right)^2. \tag{10.3}$$

Equation 10.3 is the objective function used and minimized by the optimization algorithms to estimate I_u, with f being the regression model function. From the above descriptions of how the deep autoencoder and the optimization algorithms operate, the equation below summarizes the function of the proposed approach, with f_{OA} being the optimization algorithms estimation operation and f_{DAE} being the function of the deep autoencoder.

$$y = f_{\text{DAE}}\left(W, f_{\text{OA}}\left(\vec{I}\right)\right), \tag{10.4}$$

where $\vec{I} = \begin{bmatrix} \overrightarrow{I_k} \\ \overrightarrow{I_u} \end{bmatrix}$ represents the input space of known and unknown features.

10.3 Experimental Analysis

To investigate and validate the models against other existing approaches, the Mixed National Institute of Standards and Technology (MNIST) dataset of handwritten digits was used. It consists of 60,000 training samples and 10,000 test samples, each with 784 features representing the individual pixel values of the image. The black pixels are represented by 0, while the white pixels have values of 255, as such, all the variables could be considered as being continuous. The data in its current form has been cleaned of outliers; therefore, each image used in the dataset should contribute to generalizing the system. The data are normalized to being in the range [0, 1], and randomized to improve the network performance. The training set of data is further split into training and validation sets (50,000 and 10,000, respectively).

The neural network architecture is optimized using the validation set of data with the correctness in performance of the proposed system being tested using the test data. It is worth mentioning that the objective is not to propose a state-of-the-art classification model which rivals existing state-of-the-art methods, but rather to investigate approaches to reconstruct an image in the event of missing data (missing pixels). The deep network architectures used are of size 784-1000-30-1000-784 and 784-1000-250-30-250-1000-784 based on the initial suggestion by Hinton et al. (2006). The multilayer perceptron (MLP) autoencoder used for comparison purposes has the same number of nodes in the input and output layer as the deep autoencoder, with a single hidden layer consisting of 400 nodes which is determined by varying the number of nodes in the hidden layer, and determining which architecture produces the lowest network error. It was experimentally observed that the 784-400-784 MLP autoencoder network architecture yielded a network error value of 2.5088%. The sigmoid and linear activation functions are used for the hidden and output layers of

the MLP network, respectively. The sigmoid activation function is used in each layer of the deep autoencoder except for the bottleneck layer where the linear activation function is used. The training is done using the scaled conjugate gradient (SCG) algorithm for the MLP and the stochastic gradient descent (SGD) algorithm for the deep autoencoder network, for 500 epochs. Samples are selected at random from the test set of data and features are then removed respecting the missing at random (MAR) and missing completely at random (MCAR) mechanisms as well as the arbitrary missing data pattern. This is done by creating a binomial matrix of the same size as the test data with zeros and ones adhering to the stated mechanisms and pattern and replacing every occurrence of one with NaN (implying missingness). The dataset obtained from this with missing values is used to test the performance of the missing data estimation schemes, which are then compared against existing methods.

The effectiveness of the approaches is estimated using the mean squared error (MSE), the root mean squared logarithmic error (RMSLE), the correlation coefficient (r) and the relative prediction accuracy (RPA). Also used are the signal-to-noise ratio (SNR) and global deviation (GD). The mean squared and root mean squared logarithmic errors as well as the global deviation yield measures of the difference between the actual and predicted values, and provide an indication of the capability of the estimation:

$$\mathrm{MSE} = \frac{\sum_{i=1}^{n}\left(I_i - \hat{I}_i\right)^2}{n}, \tag{10.5}$$

$$\mathrm{RMSLE} = \sqrt{\frac{\sum_{i=1}^{n}\left(\log(I_i + 1) - \log\left(\hat{I}_i + 1\right)\right)^2}{n}}, \tag{10.6}$$

and

$$\mathrm{GD} = \left(\frac{\sum_{i=1}^{n}\left(\hat{I}_i - I_i\right)}{n}\right)^2. \tag{10.7}$$

The correlation coefficient provides a measure of the similarity between the predicted and actual data. The output value of this measure lies in the range $[-1, 1]$ where the absolute value indicates the strength of the link, while the sign indicates the direction of said link. Therefore, a value close to 1 signifies a strong predictive capability while a value close to -1 signifies otherwise. In the equation below, $\bar{I}$ represents the mean of the data.

$$r = \frac{\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)\left(\hat{I}_i - \bar{\hat{I}}_i\right)}{\left[\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)^2 \sum_{i=1}^{n}\left(\hat{I}_i - \bar{\hat{I}}_i\right)^2\right]^{1/2}}. \tag{10.8}$$

The relative prediction accuracy (RPA), on the other hand, measures the number of estimates made within a specific tolerance, with the tolerance dependent on the sensitivity required by the application. The tolerance is set to 10% as it seems favourable for the application domain. This measure is given by

$$\text{RPA} = \frac{n_\tau}{n} \times 100. \tag{10.9}$$

The squared error (SE) measures a quadratic scoring rule that records the average magnitude of the error. This can be obtained by calculating the variance square root, also referred to as the standard deviation. It reduces the variance in the error, contrary to the MSE, hence its application in this work. SE can be obtained by using the formula:

$$\text{SE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(I_i - \hat{I}_i\right)^2}. \tag{10.10}$$

The mean absolute error (MAE) measures the average magnitude of the errors in a dataset without considering direction. Under ideal scenarios, the SE values are always greater than the MAE values, and in case of equality, the error values are said to have the same magnitude. This error can be calculated using the following equation:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|I_i - \hat{I}_i\right|. \tag{10.11}$$

The coefficient of determination is a metric regularly applied in statistical analysis tasks aimed at assessing the performance of a model in the explanation and prediction of future outputs. It is also referred to as the R-squared statistic, obtained by the following:

$$\text{COD} = \left(\frac{\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)\left(\hat{I}_i - \bar{\hat{I}}_i\right)}{\left[\sum_{i=1}^{n}\left(I_i - \bar{I}_i\right)^2 \sum_{i=1}^{n}\left(\hat{I}_i - \bar{\hat{I}}_i\right)^2\right]^{1/2}}\right)^2. \tag{10.12}$$

The signal-to-noise ratio used in this chapter is obtained by

$$\text{SNR} = \frac{\text{var}\left(I - \hat{I}\right)}{\text{var}\left(\hat{I}\right)}, \tag{10.13}$$

where in Eqs. (10.5)–(10.8) and Eqs. (10.10)–(10.12), n is the number of samples while I and $\hat{I}$ represent the real test set values and predicted missing output values, respectively. In Eq. (10.9), n_r represents the number of correctly predicted outputs.

10.4 Experimental Results (Three Hidden Layers)

In this section, we present the results obtained from running the experiments with the 784-1000-30-1000-784 deep autoencoder network in conjunction with the optimization algorithms.

10.4.1 Ant Colony Optimization (DL-ACO)

Considering the evaluation metrics presented in Sect. 10.3, the performance of the DL-ACO method is evaluated and compared to existing methods (MLP-GA, MLP-SA and MLP-PSO) by estimating the missing attributes concurrently, wherever missing data may be ascertained. The scenarios investigated were such that any sample/record could have at least 62, and at most 97, missing attributes (dimensions) to be approximated. The MLP network used has a structure of 784-400-784, with 784 input and output nodes in the input and output layers, respectively, and 400 nodes in the one hidden layer.

Figures 10.3 and 10.4 show the performance and comparison between the DL-ACO, MLP-PSO, MLP-SA and MLP-GA approaches. Figures 10.3 and 10.4 are bar charts that show the MSE and RMSLE values for the DL-ACO when compared to MLP-PSO, MLP-SA and MLP-GA.

We observe 0.81, 5, 31.02 and 31.21% of MSE and 6.57, 17.7, 41.22 and 41.4% of RMSLE for DL-ACO, MLP-PSO, MLP-SA and MLP-GA, respectively. The DL-ACO yielded the lowest MSE value when compared to the other approaches. Table 10.1 further backs the findings from Figs. 10.3 and 10.4 showing that the DL-ACO approach yielded the best COD, MAE, SE, RPA, SNR, GD and r values.

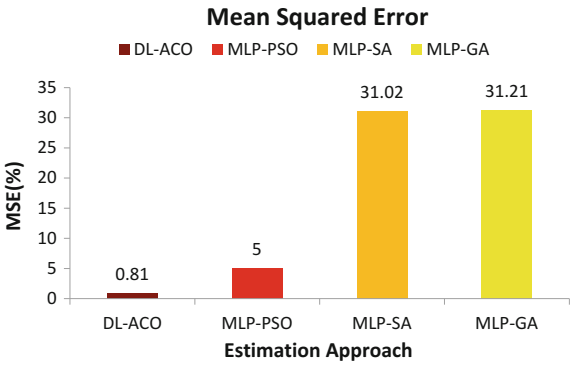

Fig. 10.3 Mean squared error versus estimation approach

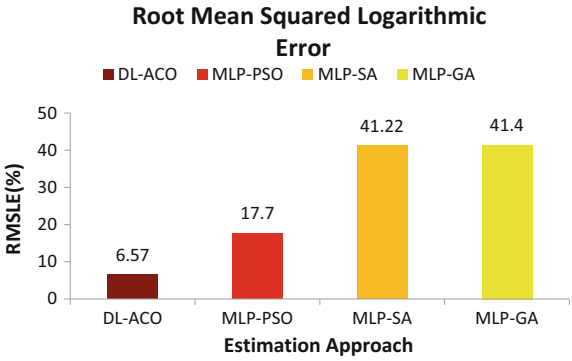

Fig. 10.4 Root mean squared logarithmic error versus estimation approach

Table 10.1 DL-ACO additional metrics

Method	DL-ACO	MLP-PSO	MLP-SA	MLP-GA
COD	**91.2**	57.4	0.0130	0.0697
MAE	**3.62**	14.48	47.39	48.03
RPA	**86.85**	54.65	10.94	10.08
SE	**8.97**	22.35	55.7	55.87
SNR	**9.45**	54.77	207.9	214.96
GD	**0.0074**	0.95	13.19	13.43
r	**95.5**	75.76	–1.14	–2.64

The bold values indicate the algoritm that yields the best results

10.4.2 Ant-Lion Optimizer (DL-ALO)

Figures 10.5 and 10.6 show the performance and comparison of the DL-ALO against the MLP-PSO, MLP-SA and MLP-GA approaches. Figures 10.5 and 10.6 are bar charts that show the MSE and RMSLE values for the DL-ALO when compared to the MLP-PSO, MLP-SA and MLP-GA.

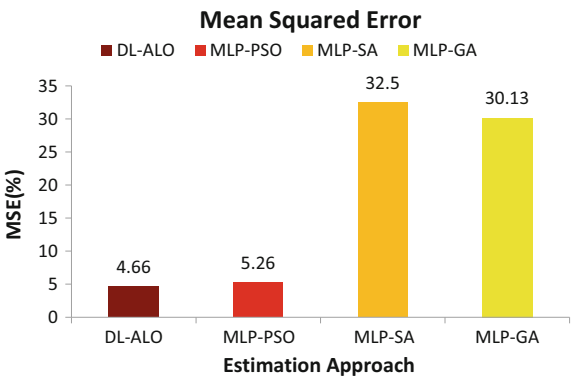

Fig. 10.5 Mean squared error versus estimation approach

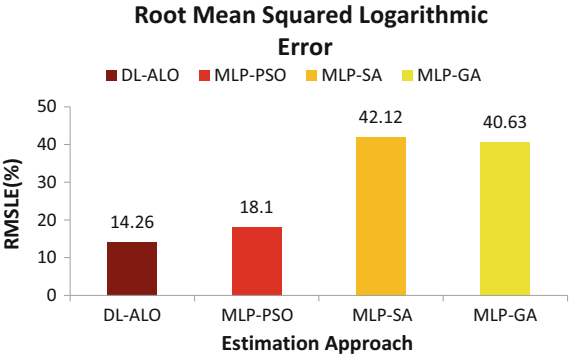

Fig. 10.6 Root mean squared logarithmic error versus estimation approach

Table 10.2 DL-ALO additional metrics

Method	DL-ACO	MLP-PSO	MLP-SA	MLP-GA
RPA	**77.68**	54.01	9.12	12.04
COD	**72.31**	57.26	2.39e-09	0.4
MAE	**9.89**	15.06	49.18	46.49
r	**85.04**	75.67	–0.0005	6.34
SE	**21.59**	22.94	57.01	54.89
SNR	217.9	**58.03**	218.75	200.36
GD	**0.39**	0.97	14.56	12.99

The bold values indicate the algoritm that yields the best results

We observed 4.66, 5.26, 32.5 and 30.13% of MSE, and 14.26, 18.1, 42.12 and 40.63% of RMSLE for DL-ALO, MLP-PSO, MLP-SA and MLP-GA, respectively. The DL-ALO yielded the lowest MSE value when compared to the other approaches. The MLP-SA and MLP-GA yielded 9.12 and 12.04% of RPA, respectively, while the DL-ALO and the MLP-PSO, respectively, yielded 77.68 and 54.01%, as presented in Table 10.2. As observed, the DL-ALO approach obtained the best values for all the performance metrics except for the SNR metric where the MLP-PSO approach performed best.

10.4.3 Cuckoo Search Algorithm (DL-CS)

Figures 10.7 and 10.8 show the performance and comparison of the DL-CS to the MLP-PSO, MLP-SA and MLP-GA. The approach introduced as will be observed differs significantly from existing methods which cater to cases whereby one or no more than 10 missing values are estimated per instance.

Before heading to the analysis of the results, note that the mean squared error is a deviation that measures the average of the squares of the errors. The optimum

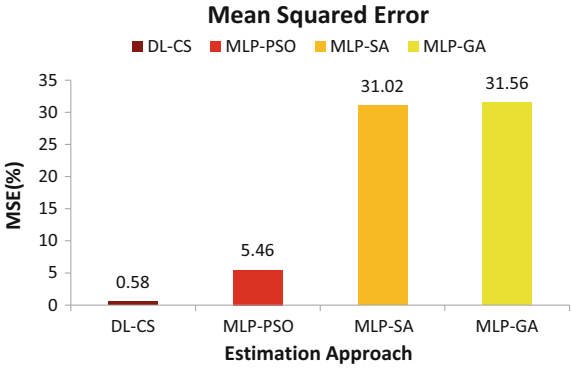

Fig. 10.7 Mean squared error versus estimation approach

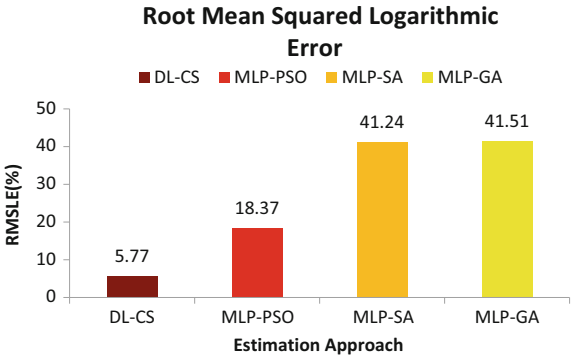

Fig. 10.8 Root mean squared logarithmic error versus estimation approach

performance of an estimator corresponds to a minimum value of the MSE. Another measure is the root mean squared logarithmic error which measures the difference between samples predicted by the models. As in the case of the MSE, the proposed approach gives better performance yielding the minimum value of the RMSLE. In evaluating data mining approaches, we also look at the correlation coefficient. It quantifies the type of dependence between samples. An estimator performs better when the correlation coefficient is higher. As with the correlation coefficient, the prediction accuracy is also given in percentage and represents the mean absolute deviation with the optimum accuracy corresponding to 100%. Figures 10.7 and 10.8 are bar charts that show the MSE and RMSLE values for the DL-CS when compared to the MLP-PSO, MLP-SA and MLP-GA.

We observed 0.58, 5.46, 31.02 and 31.56% of MSE, and 5.77, 18.37, 41.24 and 41.51% of RMSLE for DL-CS, MLP-PSO, MLP-SA and MLP-GA, respectively. The DL-CS yielded the lowest MSE value when compared to the others. These results are validated by the correlation coefficient as observed in Table 10.3. The DL-CS and MLP-PSO yielded 96.49 and 72.18% correlation values, respectively, while MLP-SA and MLP-GA showed correlations of 1.64 and 3.27%, respectively.

Table 10.3 DL-CS additional metrics

Method	DL-CS	MLP-PSO	MLP-SA	MLP-GA
RPA	**87.17**	57.58	9.67	12.5
COD	**93.11**	52.1	0.0268	0.11
MAE	**3.57**	15.01	47.74	47.66
r	**96.49**	72.18	1.64	3.27
SE	**7.65**	23.37	55.7	56.18
SNR	**7.67**	60.38	196.91	186.82
GD	**0.0156**	1.16	14.86	15.06

The bold values indicate the algoritm that yields the best results

The MLP-SA and MLP-GA yielded 9.67 and 12.5% of RPA, respectively, while DL-CS and MLP-PSO, respectively, yielded 87.17 and 57.58%, as seen in Table 10.3. As observed, the DL-CS approach obtained the best results for all four metrics presented graphically.

10.4.4 Firefly Algorithm (DL-FA)

Figures 10.9 and 10.10 show the performance and comparison of DL-FA against the MLP-PSO, MLP-SA and MLP-GA approaches. Figures 10.9 and 10.10 are bar charts that show the MSE and RMSLE values for DL-FA when compared to MLP-PSO, MLP-SA and MLP-GA.

We observed 2.69, 5.94, 30.79 and 33.11% of MSE, and 12.59, 18.93, 40.95 and 42.33% of RMSLE for DL-FA, MLP-PSO, MLP-SA and MLP-GA, respectively. The DL-FA yielded the lowest MSE value when compared to the other approaches.

The MLP-SA and MLP-GA yielded 10.59 and 8.31% of RPA, respectively, while the DL-FA and MLP-PSO, respectively, yielded 58.12 and 51.69%, as presented in

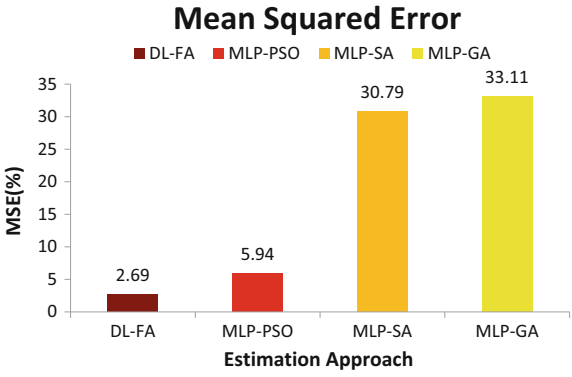

Fig. 10.9 Mean squared error versus estimation approach

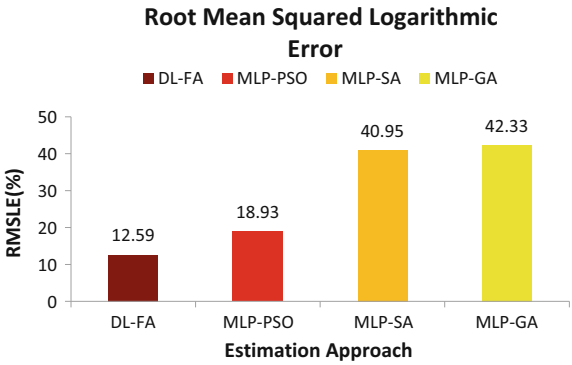

Fig. 10.10 Root mean squared logarithmic error versus estimation approach

Table 10.4 DL-FA additional metrics

Method	DL-FA	MLP-PSO	MLP-SA	MLP-GA
RPA	**58.12**	51.69	10.59	8.31
COD	**76.65**	54.76	0.06	0.24
MAE	**10.64**	16	47.57	49.66
r	**87.55**	74	2.41	–4.91
SE	**16.4**	24.36	55.49	57.54
SNR	**32.82**	61.11	220.88	234.58
GD	**0.25**	1.03	12.59	13.37

The bold values indicate the algoritm that yields the best results

Table 10.4. As observed, the DL-FA approach obtained the best values for all the performance metrics.

10.4.5 Bat Algorithm (DL-BAT)

Figures 10.11 and 10.12 show the performance and comparison of DL-BAT with MLP-PSO, MLP-SA and MLP-GA as well as Table 10.5. The approach investigated as will be observed differs significantly from existing methods. The results revealed that the DL-BAT approach outperformed the other approaches. The squared error is given in Fig. 10.11. It shows a 9.45% of error for DL-BAT while MLP-PSO, MLP-SA and MLP-GA obtained 22.61, 55.61 and 56.04% error values, respectively.

Figure 10.12 shows the correlation coefficient of the four approaches analysed, including the DL-BAT approach. It confirms the better performance of the DL-BAT approach when compared to the others. The DL-BAT exhibited a 95.24% of correlation with 83.77% of RPA while we obtained 76.73% of correlation coefficient for the MLP-PSO, and –2.01 and –3.21% of correlation for the MLP-SA and MLP-GA, respectively. The MLP-PSO depicted 56.34% of RPA, while the MLP-SA and MLP-GA produce values of 10.05 and 10.05%, respectively.

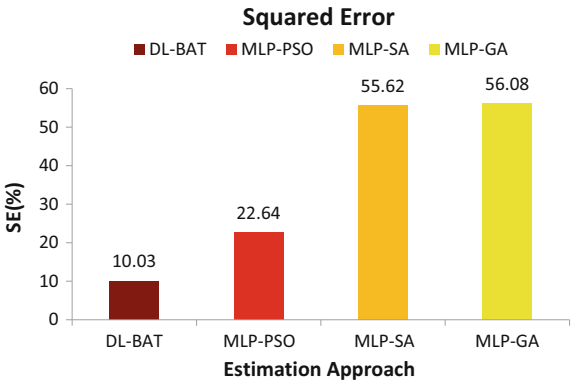

Fig. 10.11 Squared error versus estimation approach

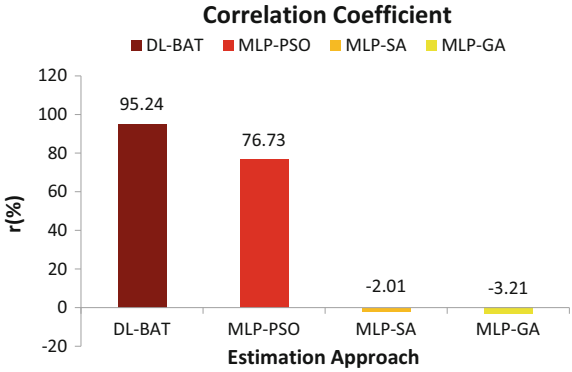

Fig. 10.12 Correlation coefficient versus estimation approach

Table 10.5 DL-BAT additional metrics

Method	DL-BAT	MLP-PSO	MLP-SA	MLP-GA
COD	**90.7**	58.88	0.04	0.1
GD	**0.06**	0.81	12.19	12.52
MAE	**5.28**	14.3	47.72	48.24
MSE	**1.01**	5.12	30.94	31.45
RMSLE	**7.53**	17.67	41.02	41.38
RPA	**83.77**	56.34	10.05	10.05
SNR	**10.16**	53.71	227.04	230.12

The bold values indicate the algoritm that yields the best results

These findings were further confirmed by the values in Table 10.5 with the DL-BAT system yielding the lowest COD, GD, MAE, MSE, RMSLE, RPA and SNR values.

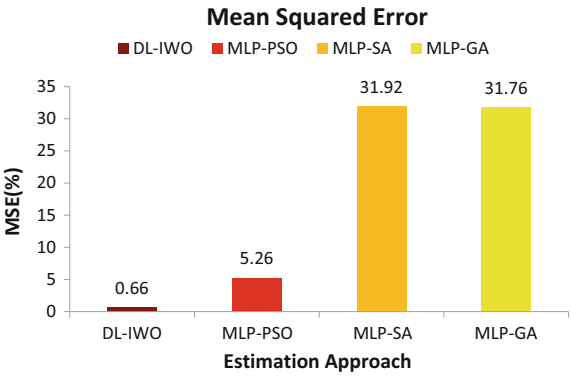

Fig. 10.13 Mean squared error versus estimation approach

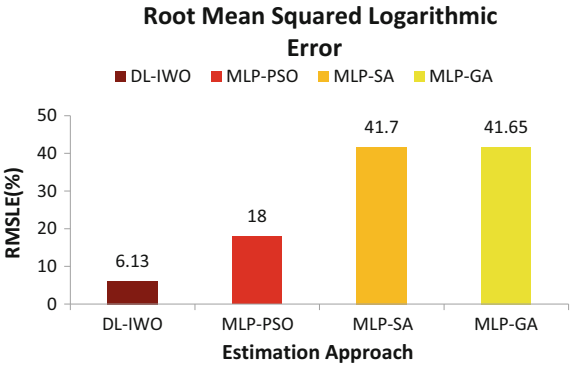

Fig. 10.14 Root mean squared logarithmic error versus estimation approach

10.4.6 Invasive Weed Optimization Algorithm (DL-IWO)

In this analysis, the DL-IWO approach was compared against existing approaches in the literature (MLP-PSO, MLP-SA and MLP-GA). The results were grouped in Figs. 10.13 and 10.14, and Table 10.6. The results revealed that the DL-IWO approach outperformed all the other approaches. The mean squared error is given in Fig. 10.13. It showed a 0.66% of error for the DL-IWO while the MLP-PSO, MLP-SA and MLP-GA yielded 5.26, 31.92 and 31.76% error, respectively.

Figure 10.14 depicts the root mean squared logarithmic error for the approaches analysed, including the DL-IWO approach. It confirms the better performance of the DL-IWO approach when compared to the other approaches. The DL-IWO exhibited a 6.13% of RMSLE while we obtained 18% of RMSLE for the MLP-PSO. The MLP-SA and MLP-GA showed RMSLE values of 41.7 and 41.65%, respectively.

Table 10.6 further confirms the findings from Figs. 10.13 and 10.14 showing that the DL-IWO approach yielded the best COD, GD, MAE, RPA, r, SE and SNR values.

Table 10.6 DL-IWO additional metrics

Method	DL-IWO	MLP-PSO	MLP-SA	MLP-GA
COD	**93.27**	56.75	0.05	0.08
GD	**0.01**	0.94	13.86	14.32
MAE	**3.83**	14.66	48.75	48.19
RPA	**85.19**	55.49	8.87	9.76
r	**96.57**	75.33	–2.18	2.86
SE	**8.12**	22.94	56.5	56.36
SNR	**7.54**	56.16	221.78	207.02

The bold values indicate the algoritm that yields the best results

10.5 Experimental Results (Five Hidden Layers)

In this section, we present the results obtained from running the experiments with the 784-1000-250-30-250-1000-784 deep autoencoder network in conjunction with the optimization algorithms.

10.5.1 Ant Colony Optimization (DL-ACO)

Considering the evaluation metrics presented in Sect. 10.3, the performance of the DL-ACO method is evaluated and compared against existing methods (MLP-GA, MLP-SA and MLP-PSO) by estimating the missing attributes concurrently, wherever missing data may be ascertained. The scenarios investigated were such that any sample/record could have at least 62, and at most 97 missing attributes (dimensions) to be approximated. The MLP network used has a structure of 784-400-784, with 784 input and output nodes in the input and output layers, respectively, and 400 nodes in the one hidden layer.

Figures 10.15 and 10.16 show the performance and comparison between the DL-ACO, the MLP-PSO, MLP-SA and MLP-GA approaches. Figures 10.15 and 10.16 are bar charts that show the MSE and the RMSLE values for the DL-ACO when compared to the MLP-PSO, MLP-SA and MLP-GA.

We observe 0.73, 5, 31.02 and 31.21% of MSE and 6.43, 17.7, 41.22 and 41.4% of RMSLE for the DL-ACO, MLP-PSO, MLP-SA and MLP-GA, respectively. The DL-ACO yielded the lowest MSE value when compared to the other approaches. Table 10.7 further confirms the findings from Figs. 10.15 and 10.16 showing that the DL-ACO approach yielded the best COD, MAE, SE, RPA, SNR, GD and r values.

Table 10.7 DL-ACO additional metrics

Method	DL-ACO	MLP-PSO	MLP-SA	MLP-GA
COD	**91.99**	57.4	0.0130	0.0697
MAE	**3.67**	14.48	47.39	48.03
RPA	**87.17**	54.65	10.94	10.08
SE	**8.56**	22.35	55.7	55.87
SNR	**8.77**	54.77	207.9	214.96
GD	**0.0070**	0.95	13.19	13.43
r	**95.91**	75.76	–1.14	–2.64

The bold values indicate the algoritm that yields the best results

10.5.2 Ant-Lion Optimizer (DL-ALO)

Figures 10.17 and 10.18 show the performance and comparison of DL-ALO against the MLP-PSO, MLP-SA and MLP-GA approaches. Figures 10.17 and 10.18 are bar charts that show the MSE and RMSLE values for DL-ALO when compared to MLP-PSO, MLP-SA and MLP-GA.

We observed 3.37, 5.26, 32.5 and 30.13% of MSE, and 12.2, 18.1, 42.12 and 40.63% of RMSLE for DL-ALO, MLP-PSO, MLP-SA and MLP-GA, respectively. The DL-ALO yielded the lowest MSE value when compared to the other approaches.

The MLP-SA and the MLP-GA yielded 9.12 and 12.04% of RPA, respectively, while the DL-ALO and the MLP-PSO, respectively, yielded 77.68 and 54.01%, as presented in Table 10.8. As observed, the DL-ALO approach obtained the best values for all the performance metrics except for the SNR metric where the MLP-PSO approach performed best.

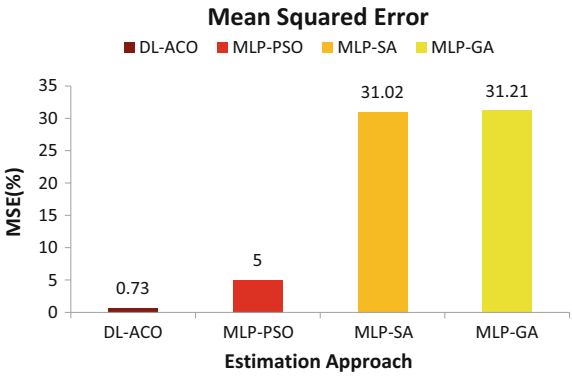

Fig. 10.15 Mean squared error versus estimation approach

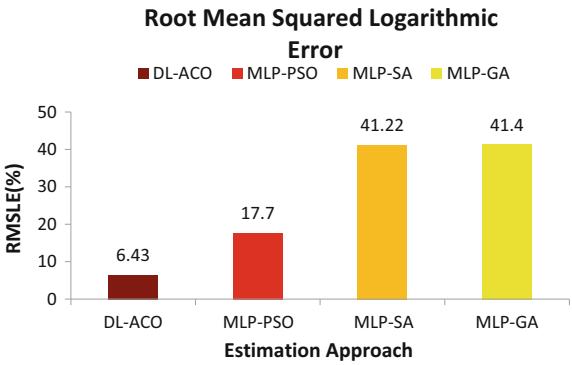

Fig. 10.16 Root mean squared logarithmic error versus estimation approach

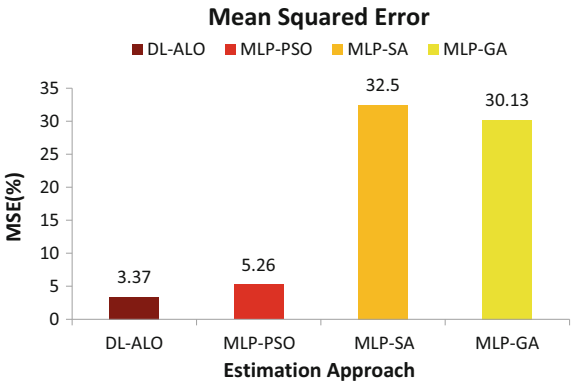

Fig. 10.17 Mean squared error versus estimation approach

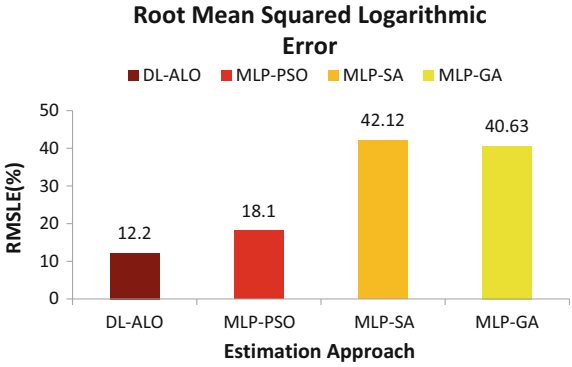

Fig. 10.18 Root mean squared logarithmic error versus estimation approach

10.5.3 Cuckoo Search Algorithm (DL-CS)

Figures 10.19 and 10.20 show the performance and comparison of DL-CS with MLP-PSO, MLP-SA and MLP-GA. The approach introduced as will be observed differs

Table 10.8 DL-ALO additional metrics

Method	DL-ACO	MLP-PSO	MLP-SA	MLP-GA
RPA	**74.57**	54.01	9.12	12.04
COD	**77.94**	57.26	2.39e-09	0.4
MAE	**9.38**	15.06	49.18	46.49
r	**88.28**	75.67	−0.0005	6.34
SE	**18.37**	22.94	57.01	54.89
SNR	110.71	**58.03**	218.75	200.36
GD	**0.13**	0.97	14.56	12.99

The bold values indicate the algoritm that yields the best results

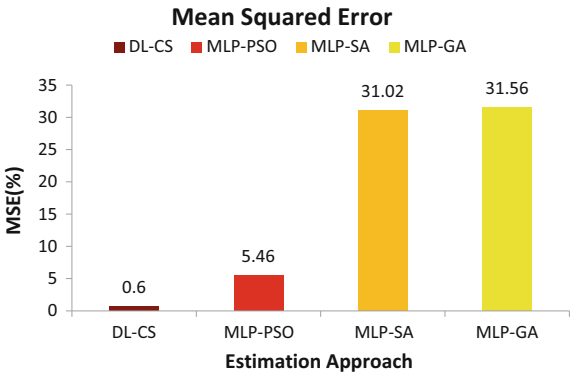

Fig. 10.19 Mean squared error versus estimation approach

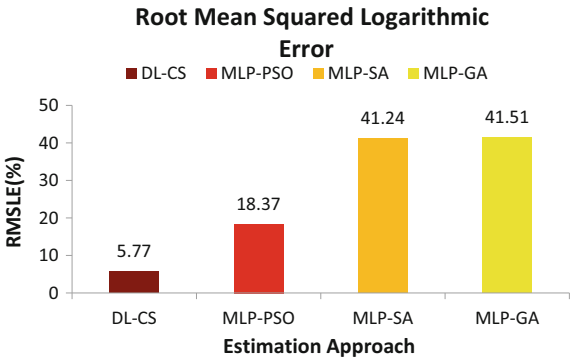

Fig. 10.20 Root mean squared logarithmic error versus estimation approach

significantly from existing methods which cater to cases whereby one or no more than 10 missing values are estimated per instance.

Before heading to the analysis of the results, note that the mean squared error is a deviation that measures the average of the squares of the errors. The optimum performance of an estimator corresponds to a minimum value of MSE. Another measure is the root mean squared logarithmic error which measures the difference

Table 10.9 DL-CS additional metrics

Method	DL-CS	MLP-PSO	MLP-SA	MLP-GA
RPA	**86.46**	54.07	9.08	11.74
COD	**93.13**	52.1	0.0268	0.11
MAE	**3.88**	15.01	47.74	47.66
r	**96.5**	72.18	1.64	3.27
SE	**7.72**	23.37	55.7	56.18
SNR	**7.8**	60.38	196.91	186.82
GD	**0.0262**	1.16	14.86	15.06

The bold values indicate the algoritm that yields the best results

between samples predicted by the models. As in the case of MSE, the proposed approach gives better performance yielding the minimum value of the RMSLE. In evaluating data mining approaches, we also look at the correlation coefficient. It quantifies the type of dependence between samples. An estimator performs better when the correlation coefficient is higher. As with the correlation coefficient, the prediction accuracy is also given in percentage and represents the mean absolute deviation with the optimum accuracy corresponding to 100%. Figures 10.19 and 10.20 are bar charts that show the MSE and RMSLE values for the DL-CS when compared to the MLP-PSO, MLP-SA and MLP-GA.

We observed 0.6, 5.46, 31.02 and 31.56% of MSE, and 5.91, 18.37, 41.24 and 41.51% of RMSLE for DL-CS, MLP-PSO, MLP-SA and MLP-GA, respectively. DL-CS yielded the lowest MSE value when compared to the others. These results are validated by the correlation coefficient as observed in Table 10.9. DL-CS and MLP-PSO yielded 96.5 and 72.18% correlation values, respectively, while MLP-SA and MLP-GA showed correlations of 1.64 and 3.27%, respectively.

MLP-SA and MLP-GA yielded 9.08 and 11.74% of RPA, respectively, while DL-CS and MLP-PSO, respectively, yielded 86.46 and 54.07%, as seen in Table 10.9. As observed, the DL-CS approach obtained the best figures for all four metrics presented graphically.

10.5.4 Firefly Algorithm (DL-FA)

Figures 10.21 and 10.22 show the performance and comparison of DL-FA against the MLP-PSO, MLP-SA and MLP-GA approaches. Figures 10.21 and 10.22 are bar charts that show the MSE and RMSLE values for DL-FA when compared to MLP-PSO, MLP-SA and MLP-GA.

We observed 2.2, 5.94, 30.79 and 33.11% of MSE, and 11.67, 18.93, 40.95 and 42.33% of RMSLE for DL-FA, MLP-PSO, MLP-SA and MLP-GA, respectively. DL-FA yielded the lowest MSE value when compared to the other approaches.

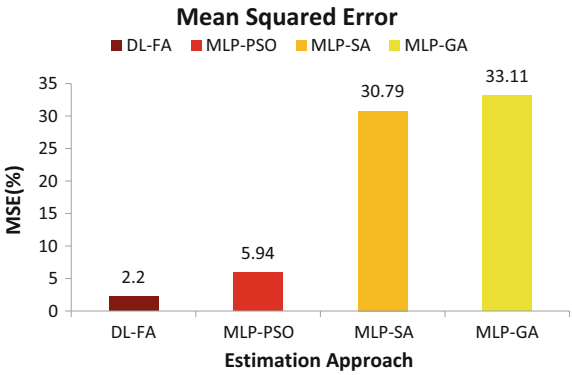

Fig. 10.21 Mean squared error versus estimation approach

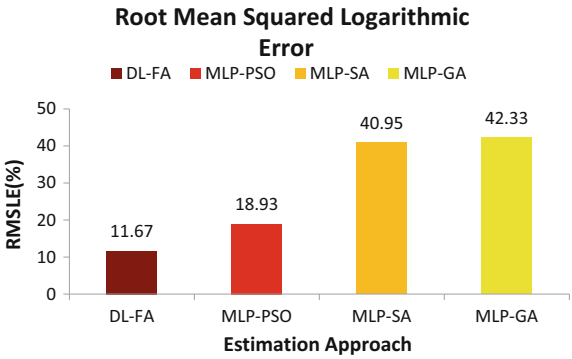

Fig. 10.22 Root mean squared logarithmic error versus estimation approach

Table 10.10 DL-FA additional metrics

Method	DL-FA	MLP-PSO	MLP-SA	MLP-GA
RPA	**57.18**	51.69	10.59	8.31
COD	**81.66**	54.76	0.06	0.24
MAE	**9.73**	16	47.57	49.66
r	**90.37**	74	2.41	–4.91
SE	**14.83**	24.36	55.49	57.54
SNR	**24.09**	61.11	220.88	234.58
GD	**0.28**	1.03	12.59	13.37

The bold values indicate the algoritm that yields the best results

MLP-SA and MLP-GA yielded 10.59 and 8.31% of RPA, respectively, while DL-FA and MLP-PSO, respectively, yielded 57.18 and 51.69%, as presented in Table 10.10. As observed, the DL-FA approach obtained the best values for all the performance metrics.

10.5.5 Bat Algorithm (DL-BAT)

Figures 10.23 and 10.24 show the performance and comparison of DL-BAT with MLP-PSO, MLP-SA and MLP-GA as well as Table 10.11. The approach investigated as will be observed differs significantly from existing methods. The results reveal that the DL-BAT approach outperforms the other approaches. The squared error is given in Fig. 10.23. It shows a 9.08% of error for DL-BAT while MLP-PSO, MLP-SA and MLP-GA obtain 22.64, 55.62 and 56.08% error values, respectively.

Figure 10.24 shows the correlation coefficient of the four approaches analysed, including the DL-BAT approach. It confirms the better performance of the DL-BAT approach when compared to the others. DL-BAT exhibits a 96.1% of correlation with 85.09% of RPA while we obtain 76.73% of correlation coefficient for MLP-PSO, and –2.01 and –3.21% of correlation for MLP-SA and MLP-GA, respectively. MLP-PSO depicts 56.34% of RPA, while MLP-SA and MLP-GA produced values of 10.05 and 10.05%, respectively.

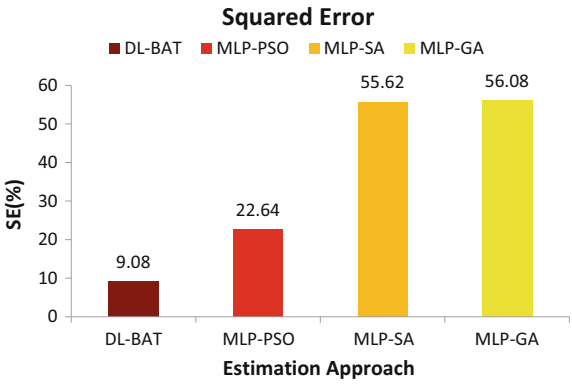

Fig. 10.23 Squared error versus estimation approach

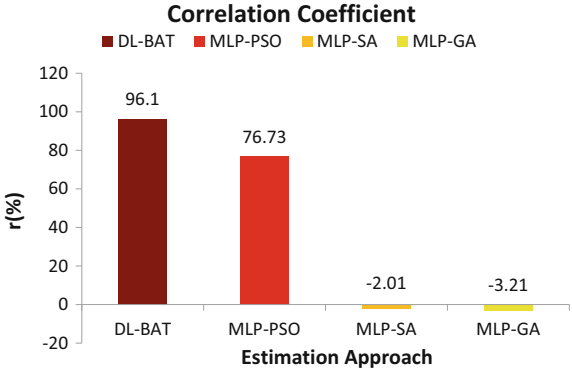

Fig. 10.24 Correlation coefficient versus estimation approach

Table 10.11 DL-BAT additional metrics

Method	DL-BAT	MLP-PSO	MLP-SA	MLP-GA
COD	**92.35**	58.88	0.04	0.1
GD	**0.05**	0.81	12.19	12.52
MAE	**4.69**	14.3	47.72	48.24
MSE	**0.82**	5.12	30.94	31.45
RMSLE	**6.82**	17.67	41.02	41.38
RPA	**85.09**	56.34	10.05	10.05
SNR	**8.19**	53.71	227.04	230.12

The bold values indicate the algoritm that yields the best results

These findings are further confirmed by the values in Table 10.11 with the DL-BAT system yielding the lowest COD, GD, MAE, MSE, RMSLE, RPA and SNR values.

10.5.6 Invasive Weed Optimization Algorithm (DL-IWO)

In this analysis, the DL-IWO approach is compared against existing approaches in the literature (MLP-PSO, MLP-SA and MLP-GA). The results are grouped in Figs. 10.25, 10.26, and Table 10.12. The results reveal that the DL-IWO approach outperforms all the other approaches. The mean squared error is given in Fig. 10.25. It shows a 0.56% of error for DL-IWO while MLP-PSO, MLP-SA and MLP-GA yield 5.26, 31.92 and 31.76% error, respectively.

Figure 10.26 depicts the root mean squared logarithmic error for the approaches analysed, including the DL-IWO approach. It confirms the better performance of the DL-IWO approach when compared to the other approaches. DL-IWO exhibits a 5.72% of RMSLE while we obtain 18% of RMSLE for MLP-PSO. MLP-SA and MLP-GA show RMSLE values of 41.7 and 41.65%, respectively.

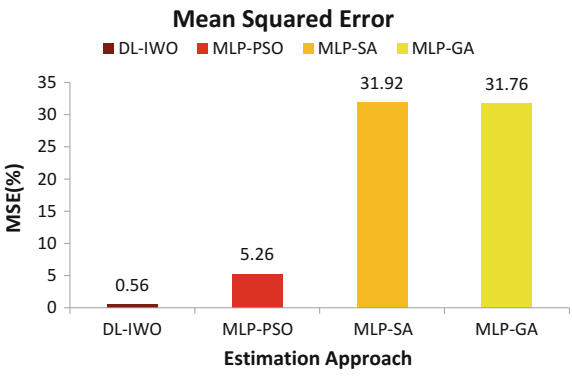

Fig. 10.25 Mean squared error versus estimation approach

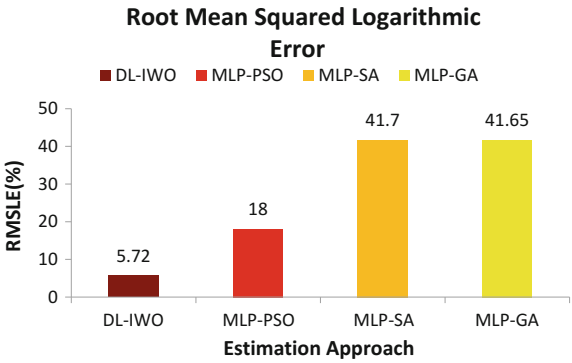

Fig. 10.26 Root mean squared logarithmic error versus estimation approach

Table 10.12 DL-IWO additional metrics

Method	DL-IWO	MLP-PSO	MLP-SA	MLP-GA
COD	**94.39**	56.75	0.05	0.08
GD	**0.01**	0.94	13.86	14.32
MAE	**3.72**	14.66	48.75	48.19
RPA	**86.49**	55.49	8.87	9.76
R	**97.16**	75.33	–2.18	2.86
SE	**7.47**	22.94	56.5	56.36
SNR	**6.44**	56.16	221.78	207.02

The bold values indicate the algoritm that yields the best results

Tables 10.12 further confirm the findings from Figs. 10.25 and 10.26 showing that the DL-IWO approach yielded the best COD, GD, MAE, RPA, R, SE and SNR values.

10.6 Conclusion

In this chapter, different deep network architectures were tested. It was observed that these networks in combination with optimization algorithms outperformed the existing narrow network models. In addition, it was observed that these networks, with three and five hidden layers performed on par with the seven hidden-layer network models, pointing to the fact that it is not necessary to use seven hidden layers as three will suffice, and training a three hidden-layered network requires lesser time than when training a seven hidden-layer deep neural network.

References

Abdella, M., & Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. In *3rd International Conference on Computational Cybernetics, (ICCC)* (pp. 207–212). IEEE.

Brain, L. B., Marwala, T., & Tettey, T. (2006). Autoencoder networks for HIV classification. *Current Science, 91*(11), 1467–1473.

Finn C., Tan, X., Duan, Y., Darrell, T., Levine, S., & Abbeel, P. (2016). Deep spatial autoencoders for visuomotor learning. In *International Conference on Robotics and Automation (ICRA)* (pp. 512–519).

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7):1527–1554.

Jerez, J. M., Molina, I., Garcıa-Laencina, P. J., Alba, E., Ribelles, N., Martın, M., et al. (2010). Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine, 50*(2), 105–115. Elsevier.

Ju, Y., Guo, J., & Liu, S. (2015). A deep learning method combined sparse autoencoder with SVM. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 257–260).

Krizhevsky, A., & Hinton, G. E. (2011). Using very deep autoencoders for content based image retrieval. In *19th European Symposium on Artificial Neural Networks (ESANN)*. Bruges, Belgium, 27–29 April 2011.

Leke, C., Twala, B., & Marwala, T. (2014). Modeling of missing data prediction: Computational intelligence and optimization algorithms. In *International Conference on Systems, Man and Cybernetics (SMC)* (pp. 1400–1404). IEEE.

Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A Swarm intelligence-deep neural network approach. In *International Conference in Swarm Intelligence* (pp. 259–270). Springer International Publishing.

Leke, C., Ndjiongue, A. R., Twala, B., & Marwala, T. (2017). A deep learning-cuckoo search method for missing data estimation in high-dimensional datasets. In *International Conference in Swarm Intelligence* (pp. 561–572). Springer, Cham.

Liew, A. W.-C., Law, N.-F., & Yan, H. (2011). Missing value imputation for gene expression data: computational techniques to recover missing data from available information. *Briefings in Bioinformatics, 12*(5), 498–513. Oxford University Press.

Myers, T. A. (2011). Goodbye, listwise deletion: Presenting hot deck imputation as an easy and effective tool for handling missing data. *Communication Methods and Measures*, *5*(4), 297–310. Taylor & Francis.

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods, 7*(2), 147. American Psychological Association.

Ssali, G., & Marwala, T. (2007). Estimation of missing data using computational intelligence and decision trees. arXiv:0709.1640.

Van Buuren, S. (2012). *Flexible imputation of missing data*. CRC press.

Vukosi, M. N., Nelwamondo, F. V., & Marwala, T. (2007). Autoencoder, principal component analysis and support vector regression for data imputation. arXiv:0709.2506.

Zhang, S. (2011). Shell-neighbor method and its application in missing data imputation. *Applied Intelligence, 35*(1), 123–133. Springer.

Chapter 11
Concluding Remarks

11.1 Summary of the Book

This book studied missing data, missing data mechanisms and missing data patterns, and the missing data estimation problem (Ramoni and Sebastiani 2001; Tremblay et al. 2010; Polikar et al. 2010; Twala 2009; Rubin 1978; Allison 2000). Furthermore, this book presents classical missing data estimation techniques as well as machine learning approaches to missing data (Allison 2002; Kalousis and Hilario 2000; Pérez et al. 2002; Marwala 2009; Dempster et al. 1997). Furthermore, this book presents artificial intelligence approaches to address the missing data problem (Twala and Cartwright 2010; Twala et al. 2008; Twala and Phorah 2010; Marwala 2015; Abdella and Marwala 2005). In addition, challenges to missing data imputation are presented.

This book also studied the theory of deep learning and its applications (Deng et al. 2013; Deng and Yu 2014; Hinton et al. 2006; Alex et al. 2012). Furthermore, this book presented information on the building blocks of deep learning frameworks such as, restricted Boltzmann machines and contrastive divergence (Leke and Marwala 2016; Hinton 2010; Tieleman 2008; LeCun et al. 2015). Moreover, this book presented information on some of the well-known deep learning frameworks such as, deep belief networks, convolutional neural networks and recurrent neural networks, to name a few (Hinton et al. 2006; Zhang and Wu 2013; Le Roux and Bengio 2008; Krizhevsky et al. 2012; Simard et al. 2003; Mikolov et al. 2010; Feng et al. 2014; Grossberg 2013).

This book also introduced a novel missing data estimation approach called deep learning-bat algorithm (DL-BAT) which combined the advantages of deep autoencoder networks and the bat algorithm. The bat algorithm is a meta-heuristic swarm intelligence technique based on the echolocation trait of bats (Yang 2010c; Zhou et al. 2014; Yang 2011; Teodoro et al. 2012).

This book also studied a novel missing data estimation approach called deep learning-cuckoo search algorithm (DL-CS) which combined the advantages of deep autoencoder networks and the cuckoo search optimization algorithm. The cuckoo

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*,
Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2_11

search algorithm is a population-based stochastic optimization technique inspired by the brood paratisism of cuckoo birds (Yang and Debb 2009, 2014; Vasanthakumar et al. 2015; Wang et al. 2016; Ali and Mohamed 2016).

This book studied a novel missing data estimation approach termed deep learning-firefly algorithm (DL-FA) which was designed with the aim of combining the advantages of deep autoencoder networks and the firefly algorithm. The firefly algorithm is a nature-inspired meta-heuristic algorithm based on the flashing patterns and behaviour of fireflies (Yang 2010a, b; Yang et al. 2012).

This book introduced a novel missing data estimation approach called deep learning-bat ant colony optimization algorithm (DL-ACO) which combined the advantages of deep autoencoder networks and the ant colony optimization algorithm. Ant colony optimization algorithm is an algorithm that mimics the innate behaviour of ants (Monteiro et al. 2012; Dorigo et al. 1991; Zecchina et al. 2006; Liu et al. 2013).

This book studied a novel missing data estimation approach called deep learning-ant-lion optimizer algorithm (DL-ALO) which combined the advantages of deep autoencoder networks and the ant-lion optimizer algorithm. Ant-lion optimizer algorithm is a meta-heuristic algorithm that mimics the interaction between ants, prey and the ant-lion species (Mirjalili 2015; Gupta and Saxena 2016; Satheeshkumar and Shivakumar 2016; Yamany et al. 2015).

This book studied a novel missing data estimation approach called deep learning-invasive weed optimization algorithm (DL-IWO) combining the advantages of deep autoencoder networks and the invasive weed optimization algorithm. Invasive weed optimization algorithm is a technique that mimics the invasion trait of weeds (Mehrabian and Lucas 2006; Veenhuis 2010; Hung et al. 2010).

This book studied the possibility of reconstructing images from reduced dimensions and using narrow artificial intelligence frameworks. This was aimed at addressing the high-execution-time drawback witnessed when deep neural network frameworks are used. The narrow artificial intelligence framework considered was a multilayer perceptron auto-associative neural network (Marwala 2001, 2013; Marwala and Chakraverty 2006).

This book studied the impact of using deep autoencoder networks with varying number of hidden layers in a deep learning framework analysis approach.

References

Abdella, M., & Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database. In *3rd International Conference on Computational Cybernetics, ICCC* (pp. 207–212). IEEE.

Alex, K., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc., (last accessed: May 2016). [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Ali, F. A., & Mohamed, A. T. (2016). A hybrid cuckoo search algorithm with Nelder-Mead method for solving global optimization problems. *SpringerPlus, 5*(1), 473. Springer International Publishing.

Allison, P. D. (2000). Multiple imputation for missing data. *Sociological Methods & Research, 28*(3), 301–309.

Allison, P. D. (2002). *Missing data*. Thousand Oaks, CA: Sage.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1997). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society, 39*(1), 1–38.

Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., & Williams, J. (2013). Recent advances in deep learning for speech research at Microsoft. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 8604–8608).

Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends in Signal Processing, 7*(3–4), 197–387.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). Positive feedback as a search strategy. Tech. Rep.

Feng, X., Zhang, Y., & Glass, J. (2014). Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1759–1763).

Grossberg, S. (2013). Recurrent neural networks. *Scholarpedia, 8*(2), 1888.

Gupta, E., & Saxena, A. (2016). Performance evaluation of antlion optimizer based regulator in automatic generation control of interconnected power system. *Journal of Engineering, 2016,* 1–14.

Hinton, G. (2010). A practical guide to training restricted Boltzmann machines. *Momentum, 9*(1), 926.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.

Hung, H. L., Chao, C. C., Cheng, C. H., & Huang, Y. F. (2010). Invasive weed optimization method based blind multiuser detection for mc-cdma interference suppression over multipath fading channel. *International Conference on Systems, Man and Cybernatics (SMC)* (pp. 2145–2150).

Kalousis, A., & Hilario, M. (2000). Supervised knowledge discovery from incomplete data. In *Proceedings of the 2nd International Conference on Data Mining*. Retrieved October 2016, from http://cui.unige.ch/AI-group/research/metal/Papers/missingvalues.ps.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Le Roux, N., & Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation, 20*(6), 1631–1649.

LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). Deep learning. *Nature, 521,* 436–444.

Leke, C., & Marwala, T. (2016). Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference in Swarm Intelligence* (pp. 259–270). Springer International Publishing.

Liu, X. J., Yi, H., & Ni, Z.-H. (2013). Application of ant colony optimization algorithm in process planning optimization. *Journal of Intelligent Manufacturing, 24*(1), 1–13.

Marwala, T. (2009). *Computational intelligence for missing data imputation: Estimation and management knowledge optimization techniques*. New York: Information Science Reference.

Marwala, T. (2001). Probabilistic fault identification using a committee of neural networks and vibration data. *Journal of Aircraft, 38*(1), 138–146.

Marwala, T., & Chakraverty, S. (2006). Fault classification in structures with incomplete measured data using autoassociative neural networks and genetic algorithm, 90(4).

Marwala, T. (2013). *Economic modelling using artificial intelligence methods*. UK: Springer.

Marwala, T. (2015). *Causality, correlation, and artificial intelligence for rational decision making*. Singapore: World Scientific.

Mehrabian, A. R., & Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics, 1,* 355–366.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 8,* 80–98.

Monteiro, M. S. R., Fontes, D. B. M. M., & Fontes, F. A. C. C. (2012). *Ant colony optimization: a literature survey*. Universidade do Porto, Faculdade de Economia do Porto, FEP Working Papers. Retrieved January 2016, from http://EconPapers.repec.org/RePEc:por:fepwps:474.

Pérez, A., Dennis, R. J., Gil, J. F. A., Róndon, M. A., & López, A. (2002). Use of the mean, hot deck and multiple imputation techniques to predict outcome in intensive care unit patients in Colombia. *Journal of Statistics in Medicine, 21*(24), 3885–3896.

Polikar, R., De Pasquale, J., Mohammed, H. S., Brown, G., & Kuncheva, L. I. (2010). Learn++mf: A random subspace approach for the missing feature problem. *Pattern Recognition, 43*(11), 3817–3832.

Ramoni, M., & Sebastiani, P. (2001). Robust learning with missing data. *Journal of Machine Learning, 45*(2), 147–170.

Rubin, D. (1978). Multiple imputations in sample surveys-a phenomenological Bayesian approach to nonresponse. *Proceedings of the Survey Research Methods Section of the American Statistical Association, 1,* 20–34.

Satheeshkumar, R., & Shivakumar, R. (2016). Ant lion optimization approach for load frequency control of multi-area interconnected power systems. *Circuits and Systems, 7,* 2357–2383.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). *Best practices for convolutional neural networks applied to visual document analysis* (pp. 958). IEEE.

Teodoro C. B., Leandro d-. S. C., & Luiz L. (2012). Bat-Inspired Optimization Approach for the Brushless DC Wheel Motor Problem. *Transactions on Magnetics, IEEE, 48*(2), 947–950.

Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, ser. New York, NY, USA: ACM, pp. 1064–1071. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390290. Accessed May 2016.

Tremblay, M. C., Dutta, K., & Vandermeer, D. (2010). Using data mining techniques to discover bias patterns in missing data. *Journal of Data and Information Quality,* 2(1).

Twala, B., & Cartwright, M. (2010). Ensemble missing data techniques for software effort prediction. *Intelligent Data Analysis, 14*(3), 299–331.

Twala, B. (2009). An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence, 23*(5), 373–405.

Twala, B. E. T. H., Jones, M. C., & Hand, D. J. (2008). Good methods for coping with missing data in decision trees. *Pattern Recognition Letters, 29*(7), 950–956.

Twala, B., & Phorah, M. (2010). Predicting incomplete gene microarray data with the use of supervised learning algorithms. *Pattern Recognition Letters, 31,* 2061–2069.

Vasanthakumar, S., Kumarappan, N., Arulraj, R. & Vigneysh, T. (2015). Cuckoo Search Algorithm based Environmental Economic Dispatch of Microgrid System with Distributed Generation. In *International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)* (pp. 575–580). IEEE.

Veenhuis, C. (2010). Binary invasive weed optimization. In *Second World Congress on Nature and Biologically Inspired Computing* (pp. 449–454).

Wang, J., Zhou, B., & Zhou, S. (2016). An improved cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation. *Computational Intelligence and Neuroscience*, 8.

Yamany, W., Tharwat, A., Fawzy, M., Gaber, T., & Hassanien, A. E. (2015). A new multilayer perceptrons trainer based on ant lion optimization algorithm. In *Fourth International Conference on Information Science and Industrial Applications (ISI)* (pp. 40–45).

Yang, X. S., & Debb, S. (2009). Cuckoo Search via Levy Flights. *World Congress on Nature and Biologically Inspired Computing (NaBIC), 48*(2), 210–214.

Yang, X.-S. (2010a). Firefly algorithm, Levy flights and global optimization. In M. Bramer, R. Ellis, M. Petridis (Eds.), *Research and Development in Intelligent Systems XXVI* (pp. 209–218). London: Springer.

Yang, X. S. (2010b). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation, 2*(2), 78–84.

Yang, X. S. (2010c). A New Metaheuristic Bat-Inspired Algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NISCO), Studies in Computational Intelligence* (pp. 65–74). Springer, Heidelberg.

Yang, X. S. (2011). Bat Algorithm for Multi-objective Optimization. *International Journal of Bio-Inspired Computation, 3*(5), 267–274.

Yang, X. S., Hosseini, S. S. S., & Gandomi, A. H. (2012). Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. *Applied Soft Computing, 12*(3), 1180–1186.

Yang, X. S., & Debb, S. (2014). Cuckoo search: recent advances and applications. *Neural Computing and Applications, 24*(1), 169–174.

Zecchina, A. C., Simpsona, A. R., Maiera, H. R., Leonarda, M., Roberts, A. J., & Berrisforda, M. J. (2006). Application of two ant colony optimisation algorithms to water distribution system optimisation. *Mathematical and Computer Modelling, 44*(5–6), 451–468.

Zhang, X. L., & Wu, J. (2013). Deep belief networks based voice activity detection. *IEEE Transactions on Audio, Speech and Language Processing, 21*(4), 697–710.

Zhou, Y., Xie, J., Li, L. & Ma, M. (2014). Cloud Model Bat Algorithm. *The Scientific World Journal, 2014*(237102), 1–11. Hindawi Publishing Corporation.

Index

C. A. Leke and T. Marwala, *Deep Learning and Missing Data in Engineering Systems*, Studies in Big Data 48, https://doi.org/10.1007/978-3-030-01180-2