Chapter 7
Missing Data Approaches to Economic Modeling: Optimization Approach

Abstract This chapter introduces an auto-associative network with optimization methods for modelling economic data. This resulting architecture is a missing data estimation technique, and this is used to predict the production volume by treating it as a missing variable. The autoassociative network is created using a multi-layered perceptron network, while the optimization techniques which are implemented are particle swarm optimization, genetic algorithms and simulated annealing. The results obtained are then compared.

7.1 Introduction

In this chapter, inference is treated as a correlation phenomenon by applying the auto-associative multi-layer perceptron network (Marwala 2009). This, in essence, yields a missing data estimation problem. Moreau et al. (2012) applied this approach for estimating missing data in the life cycle inventory of hydroelectric power plants, while Tsai and Yang (2012) applied neural networks to improve measurement invariance assessments in survey research data that had some values missing. Kim and Shin (2012) applied the factoring likelihood technique for non-monotone missing data estimation while Rey-del-Castillo and Cardeñosa (2012) applied fuzzy min-max neural networks for missing data imputation.

The missing data framework implemented in this chapter is constructed using a multi-layered perceptron, and the missing data is estimated using three optimization methods; namely; particle swarm optimization, genetic algorithm, and simulated annealing (Marwala 2010, 2012; Marwala and Lagazio 2011). The developed framework is then tested on manufacturing data from the South African Reserve Bank. The next section describes the missing data estimation framework.

T. Marwala, *Economic Modeling Using Artificial Intelligence Methods*, Advanced Information and Knowledge Processing, DOI 10.1007/978-1-4471-5010-7_7,
© Springer-Verlag London 2013

7.2 Missing Data Estimation Method

The missing data estimation procedure suggested in this chapter involves the application of a neural network model that is trained to recall itself (i.e. predict its input vector) and is called an auto-associative neural network (Miranda et al. 2012; Makki and Hosseini 2012). Mathematically, the auto-associative model can be written as follows (Marwala 2009):

$$\{Y\} = f(\{X\}, \{W\}) \tag{7.1}$$

In Eq. 7.1, $\{Y\}$ is the output vector, $\{X\}$ the input vector and $\{W\}$ is the free parameter vector. In the case of a neural network, the free parameters are called weights. Because the model is trained to predict its own input vector, the input vector $\{X\}$ is approximately equal to output vector $\{Y\}$ and consequently $\{X\} \approx \{Y\}$. In actual fact, the input vector $\{X\}$ and output vector $\{Y\}$ will not always be perfectly the same, therefore an error function expressed as the difference between the input and output vector is defined (Marwala 2009):

$$\{e\} = \{X\} - \{Y\} \tag{7.2}$$

Substituting the value of $\{Y\}$ from Eq. 7.1 into Eq. 7.2, the following expression is obtained (Marwala 2009):

$$\{e\} = \{X\} - f(\{X\}, \{W\}) \tag{7.3}$$

Because the aim is for the error to be minimized and non-negative, the function can be modified as a square of Eq. 7.3 (Marwala 2009):

$$\{e\} = (\{X\} - f(\{X\}, \{W\}))^2 \tag{7.4}$$

For missing data, some of the values for the input vector $\{X\}$ are not obtainable. Therefore, we can classify the input vector elements into $\{X\}$ known vector represented by $\{X_k\}$ and $\{X\}$ unknown vector represented by $\{X_u\}$. Modifying Eq. 7.4 in terms of $\{X_k\}$ and $\{X_u\}$ we have (Marwala 2009):

$$\{e\} = \left(\begin{Bmatrix} \{X_k\} \\ \{X_u\} \end{Bmatrix} - f\left(\begin{Bmatrix} \{X_k\} \\ \{X_u\} \end{Bmatrix}, \{W\} \right) \right)^2 \tag{7.5}$$

The error vector in Eq. 7.5 can be condensed into a scalar by integrating over the size of the input vector and the number of training examples as follows (Marwala 2009):

$$E = \left\| \left(\begin{Bmatrix} \{X_k\} \\ \{X_u\} \end{Bmatrix} - f\left(\begin{Bmatrix} \{X_k\} \\ \{X_u\} \end{Bmatrix}, \{W\} \right) \right) \right\| \tag{7.6}$$

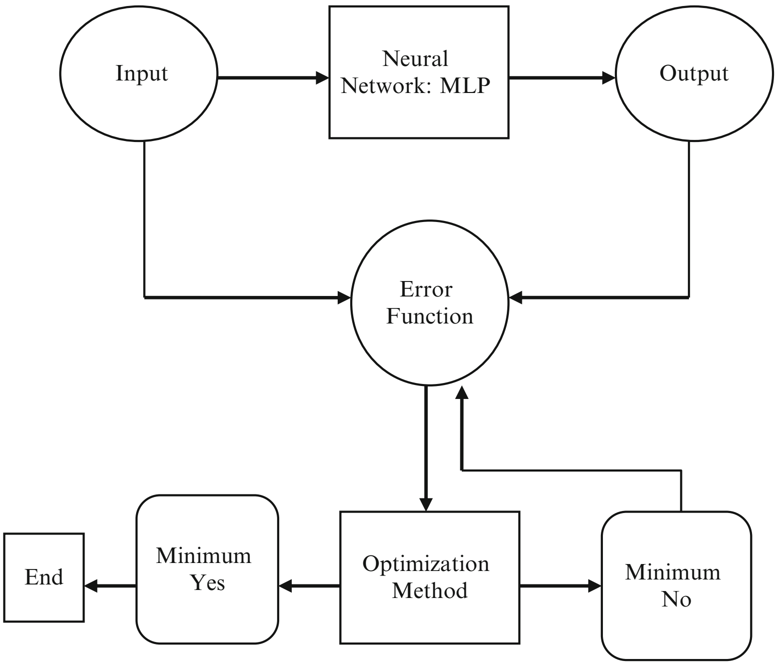

Fig. 7.1 Schematic representation of the missing data estimation model

The objective function expressed in Eq. 7.6 is known as the missing data estimation equation. To estimate the missing input values, Eq. 7.6 is minimized and, in this chapter, artificial intelligence techniques called particle swarm optimization (Kennedy and Eberhart 1995, 2001; Shi and Eberhart 1998; Kennedy 1997) and simulated annealing (Kirkpatrick et al. 1983; Černý 1985; Metropolis et al. 1953; Granville et al. 1994) are applied. It must be taken into account that any optimization technique or a combination of these can be applied to realize this objective. Particle swarm optimization and simulated annealing are selected for the reason that they both have a higher probability of identifying the global optimum solution than traditional optimization techniques such as the scaled conjugate gradient technique, which was used for training the MLP network in Chaps. 3 and 4. For the minimization of Eq. 7.6 to be successful, the identification of a global optimum solution, as opposed to local one, is unequivocally critical because if this is not attained, then a wrong approximation of the missing data will be realized. The missing data process described in this section is illustrated in Fig. 7.1 (Marwala 2009).

Briefly, the objective function known as the missing data estimation equation is derived from the error function of the input and output vector achieved from the trained neural network. The missing data estimation equation is then minimized using the particle swarm optimization method, genetic algorithm and simulated annealing to estimate the missing variables given the observed variables $\{X_k\}$ and the model f explaining the interrelationships and the rules describing the data.

7.3 Auto-associative Networks for Missing Data Estimation

The mathematical background to multilayer perceptron neural networks and auto-associative networks are explained in this section. This chapter applies multi-layered perceptron neural networks to construct auto-associative neural networks (Marwala 2012). As described in Chaps. 3 and 4, the relationship between the output y and input x can be written as follows, for the MLP network (Marwala 2012):

$$y_k = \sum_{j=0}^{M} w_{kj}^{(2)} \tanh\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right) \tag{7.7}$$

where $w_{ji}^{(1)}$ and $w_{kj}^{(2)}$ denotes weights in the first and second layer, respectively, going from input i to hidden unit j, M is the number of hidden units, and d is the number of output units. In this chapter, as it was described in Chaps. 3 and 4, the network weights in Eq. 7.7 are estimated using the maximum-likelihood approach and the scaled conjugate gradient optimization method.

An auto-associative network is a network that is trained to remember its inputs. This implies that, every time an input is given to the network, the output is the approximated input. These networks have been applied in a number of applications including novelty detection, missing data estimation, feature selection, and data compression (Kramer 1992; Marwala 2009).

There has been more interest in treating the missing data problem by approximation or imputation (Abdella 2005; Abdella and Marwala 2005, 2006; Nelwamondo and Marwala 2007; Nelwamondo 2008). The mixture of the auto-associative neural network and genetic algorithm has been shown to be a successful technique to approximate missing data. The method for estimating missing data in this chapter depends on the identification of the relationships or correlations between the variables that make up the dataset, and the multi-layered perceptron is able to achieve this (Kramer 1992).

Other successful applications of auto-associative network includes its use in fault detection in turbine blades (Lemma and Hashim 2012; Dervilis et al. 2012; Palmé et al. 2011), face recognition (Wang and Yang 2011) and speech recognition (Sivaram et al. 2010).

It must be noted that, on using auto-associative neural networks for data compression, the network has fewer nodes in the hidden layer. Nonetheless, for missing data estimation it is vital that the network is as accurate as possible and that this accuracy is not necessarily achieved through few hidden nodes as is the case when these networks are used for data compression. The auto-associative network is shown in Fig. 7.2 (Marwala 2009).

In this chapter, global optimum techniques, particle swarm optimization and simulated annealing were applied to identify the global optimum solution and are the subject of the next two sections.

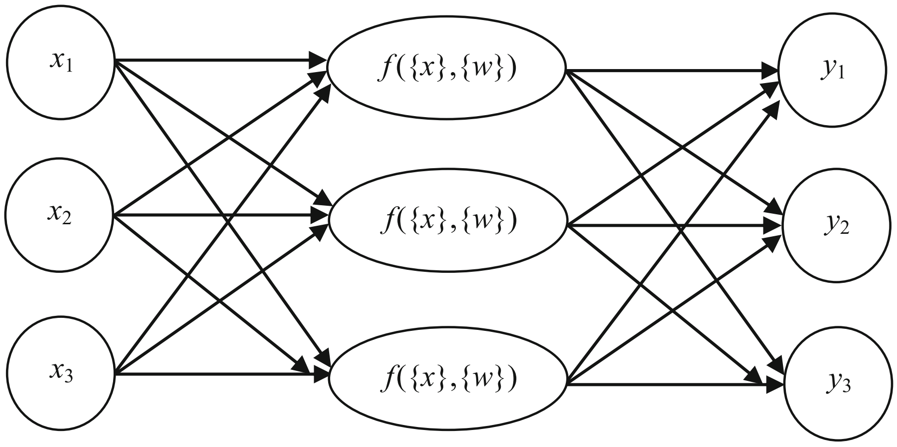

Fig. 7.2 An auto-associative MLP network having two layers of adaptive weights

7.4 Particle Swarm Optimization

This chapter applies particle swarm optimization (PSO) to solve Eq. 7.6. PSO is a stochastic, population-based evolutionary procedure that has been extensively used for the optimization of complex problems (Engebrecht 2005). It is inspired by principles that are based on swarm intelligence. Swarm intelligence consists of two aspects and these are: group knowledge and individual knowledge. Each member of a swarm acts by balancing between individual knowledge and group knowledge.

When solving problems using PSO, an objective function is formulated indicating the desired outcome. In this chapter, the objective function is the missing data estimation function represented by Eq. 7.6. To achieve an optimum missing data estimation function state, a social network representing a population of possible solutions is randomly generated. The individuals within this social network interrelate with their neighbours and are called particles. A process to update these particles is undertaken by assessing the fitness of each particle. Each particle is able to recall the position where it had its best success as measured by the missing data estimation function. The best solution of the particle is called the local best and each particle makes this information on the local best accessible to their neighbors and, in turn, also observe their neighbors' success.

The PSO was developed by Kennedy and Eberhart (1995) and it was inspired by algorithms that model the "flocking behavior" seen in birds. PSO has been very successful in optimizing complex problems. Marwala (2005) used PSO to improve finite element models to better reflect the measured data. This method was compared to a finite element model updating approach that used simulated annealing and a genetic algorithm. The proposed methods were tested on a simple beam and an unsymmetrical H-shaped structure. It was observed that, on average, the PSO method gave the most accurate results followed by simulated annealing and then the genetic algorithm.

Dindar and Marwala (2004) successfully used PSO to optimize the structure of a committee of neural networks. The results obtained from the optimized networks were found to give better results than both un-optimized networks and the committee of networks.

Ransome et al. (2005) successfully used PSO to optimize the position of a patient during radiation therapy. In this application, a patient positioning system integrating a robotic arm was designed for proton beam therapy. A treatment image was aligned with a pre-defined reference image and this was attained by aligning the radiation and reference field boundaries and then registering the patient's anatomy relative to the boundary. Methods for both field boundary and anatomy alignment, including particle swarm optimization, were implemented. It was found that the PSO was successful to overcome problems in existing solutions.

Farzi et al. (2013) applied PSO to choose the best portfolio in 50 supreme Tehran Stock Exchange companies and optimize the rate of return, risks, liquidity, and sharp ratio. The results were then compared to Markowitz's approach (Markowitz 1952) and genetic algorithms and it was observed that, although the return of the portfolio of PSO model was less than in Markowitz approach model, it was able to decrease the risk.

Nasir et al. (2012) applied a dynamic neighbourhood learning based particle swarm optimizer for global numerical optimization and the results indicated good performance on locating the global optimum solution on complicated and multi-modal fitness functions when compared to five other types of PSO. Muthukaruppan and Er (2012) applied the PSO for diagnosis of coronary artery disease while Kalatehjari et al. (2012) applied PSO for slope stability analysis of homogeneous soil slopes. Gholizadeh and Fattahi (2012) applied PSO for design optimization of tall steel buildings, while Karabulut and Ibrikci (2012) applied PSO to identify transcription factor binding sites.

When applying PSO, each particle which is represented by two vectors: $p_i(k)$ the position and $v_i(k)$ the velocity at step k. Positions and velocities of particles are randomly generated and then updated using the position of the best solution that a specific particle has encountered during the simulation called $pbest_i$ and the best particle in the swarm which is called $gbest(k)$. The updated velocity of a particle i can be estimated using the following equation (Kennedy and Eberhart 1995):

$$v_i(k+1) = \gamma v_i(k) + c_1 r_1 \left(pbest_i - p_i(k)\right) + c_2 r_2 \left(gbest(k) - p_i(k)\right) \tag{7.8}$$

Here, γ is the inertia of the particle, c_1 and c_2 are the 'trust' parameters, r_1 and r_2 are random numbers between 0 and 1. In Eq. 7.8, the first expression is the current motion, the second expression is the particle memory influence, and the third expression is the swarm influence. The updated position of a particle i can be estimated using these equations (Kennedy and Eberhart 1995):

$$p_i(k+1) = p_i(k) + v_i(k+1) \tag{7.9}$$

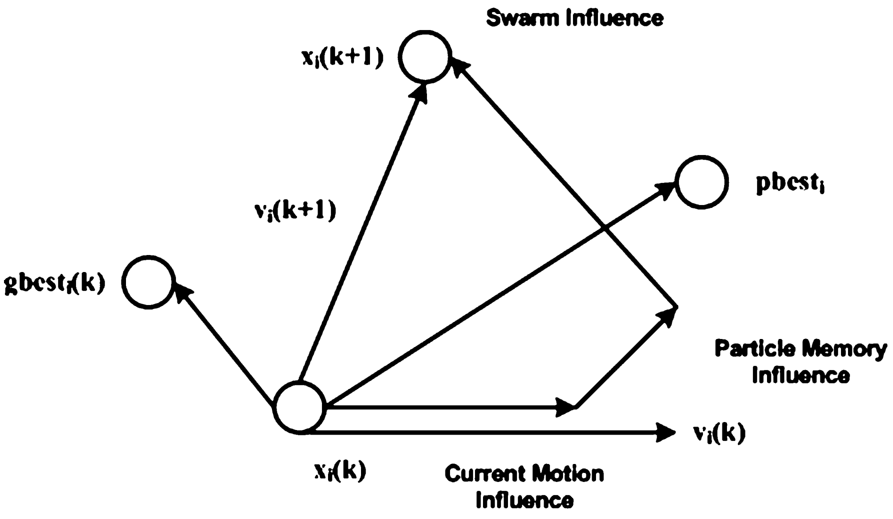

Fig. 7.3 Velocity and particle update in particle swarm optimization

The inertia of the particle regulates the relationship between the current velocity of the particle and the previous velocity. The trust parameter c_1 represents how much confidence the current particle has on itself, while the trust parameter c_2 represents the confidence the current particle has on the population. The parameters r_1 and r_2 are random numbers between 0 and 1 and they allow the swarm to explore the space.

The implementation of PSO can be summarized as follows, and is also shown in Fig. 7.3 (Kennedy and Eberhart 1995; Marwala 2010):

1. Randomly initialize a population of particles' positions and velocities.
2. Estimate the velocity for each particle in the swarm using Eq. 7.8.
3. Update the position of each particle using Eq. 7.9.
4. Repeat Steps 2 and 3 until convergence.

7.5 Genetic Algorithms (GA)

The missing data estimation method presented in this chapter also uses a genetic algorithm to estimate the missing data by minimizing Eq. 7.6. A genetic algorithm is a population-based, probabilistic technique that operates to identify a solution to a problem from a population of possible solutions (Goldberg 1989, 2002; Holland 1975; Marwala 2009). It is applied to identify estimated solutions to challenging problems through the similarity of the principles of evolutionary biology to computer science (Goldberg 2002; Marwala 2009; Tettey and Marwala 2006).

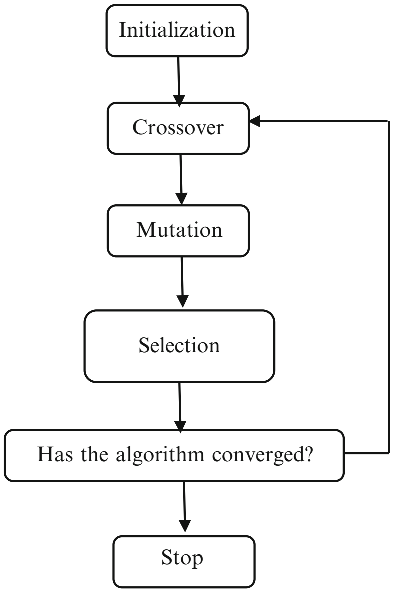

Fig. 7.4 Flow chart of the genetic algorithm method

It was derived from Darwin's theory of evolution where members of the population compete to survive and reproduce, while the weaker members die-out from the population.

Every individual has a fitness value indicating how well it fulfills the objective of solving the problem. New individual solutions are created during a cycle of generations, where selection and recombination operations occur, alike how gene transfer occurs to the current individuals. This continues until a termination condition is achieved, then the best individual by far is deemed to be the estimation for missing data. This chapter explains the application of a genetic algorithm to optimize Eq. 7.6.

Successful applications of the genetic algorithm include optimizing rough set partitions (Crossingham and Marwala 2008), missing data imputation (Hlalele et al. 2009), finite element updating (Marwala 2002, 2010), controlling fermentation (Marwala 2004), fault diagnosis (Marwala and Chakraverty 2006), HIV prediction (Leke et al. 2006), training neural networks (Marwala 2007), stock market prediction (Marwala et al. 2001), bearing fault classification (Mohamed et al. 2006), optimal weight classifier selection (Hulley and Marwala 2007) and call performance classification (Patel and Marwala 2009).

When applying the genetic algorithm, the following steps are followed: initialization, crossover, mutation, selection, reproduction, and termination. The three most important aspects of using a genetic algorithm are the definition of the objective function, implementation of the genetic representation, and implementation of the genetic operators (Marwala 2012). The details of genetic algorithms are shown in Fig. 7.4.

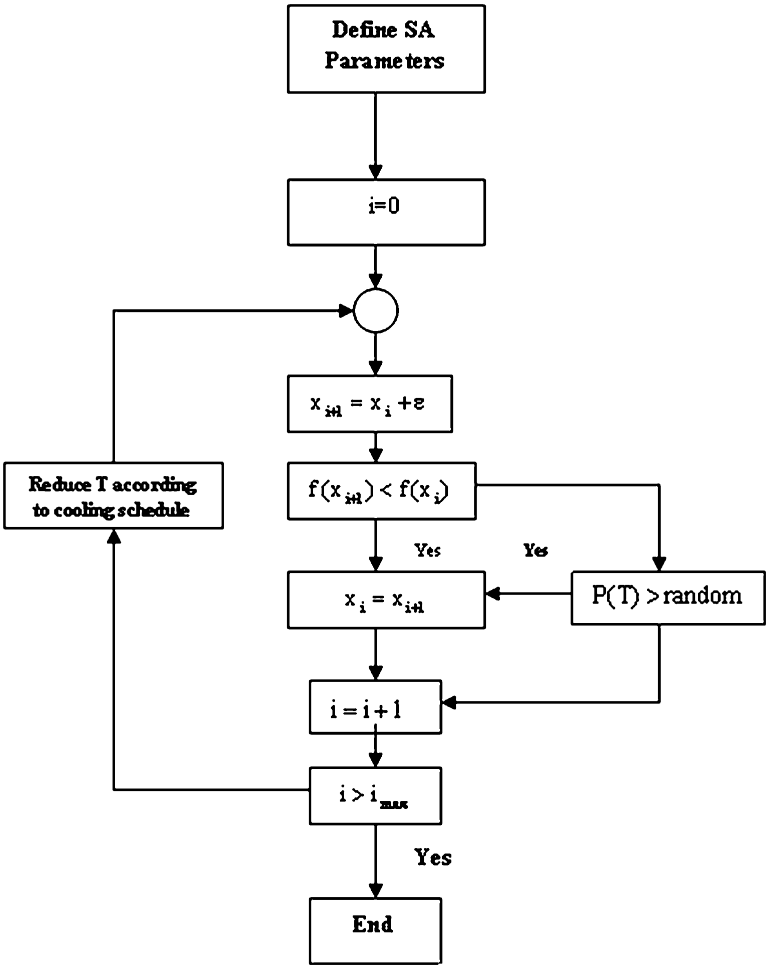

Fig. 7.5 The diagram of simulated annealing

7.5.1 *Initialization*

At this stage, a population of individual solutions is randomly created. This initial population is sampled so as to cover a good representation of the solution space.

7.5.2 *Crossover*

The crossover operator mixes genetic information in the population by cutting pairs of chromosomes at random points along their length and exchanging the cut sections over (Goldberg 2002, 1989; Marwala 2010; Banzhaf et al. 1998). In this chapter, a one crossover point method is selected. This is done by copying a binary string from the beginning of a chromosome to the crossover point from one parent, and the rest is copied from the second parent. For example, if two chromosomes in binary space $a = \mathbf{1100}1011$ and $b = 1101\mathbf{1111}$ undergo a one-point crossover at the midpoint, then the resulting offspring may be $c = 11001111$.

7.5.3 Mutation

The mutation operator introduces new information into the chromosome and, by so doing, prevents the genetic algorithm simulation from being trapped in a local optimum solution (Goldberg 2002; Marwala 2010). In this chapter, adaptive mutation is applied by randomly producing adaptive directions with respect to the previous successful or unsuccessful generation. The feasible region is bounded by the constraints and a step size is selected along each direction whereby linear constraints and bounds are not violated.

7.5.4 Selection

In every generation, a selection of the proportion of the present population is chosen to create a new population. This selection is achieved by applying the fitness-based technique, where solutions that are fitter, as measured by Eq. 7.6, have a higher probability of survival. Some selection methods rank the fitness of each solution and choose the best solution, while other procedures rank a randomly designated aspect of the population. There are quite many selection procedures and in this chapter we use roulette-wheel selection (Goldberg 2002). Roulette-wheel selection is a genetic operator used for choosing possible solutions in a GA optimization procedure.

In this method, each likely method is allocated a fitness function that is applied to map the probability of selection with each individual solution. Let's say, if the fitness f_{i} is of individual i in the population, then the probability that this individual is chosen is (Goldberg 2002):

$$p_i = \frac{f_i}{\sum\limits_{j=1}^{N} f_j} \tag{7.10}$$

Here, N is the total population size.

This technique ensures that solutions with higher fitness values have higher probabilities of survival than those with a lower fitness value. The benefit of this is that, even though a solution may have a low fitness value, it may still have some aspects that are advantageous in the future.

7.5.5 Termination

The technique described is repeated until a termination condition has been achieved, either for the reason that a chosen solution that satisfies the objective function has been identified or for the reason that a stated number of generations have been realized or the solution has converged or any combination of these.

7.6 Simulated Annealing (SA)

Simulated Annealing (SA) is a Monte Carlo technique that is applied to identify an optimal solution. It was inspired by the annealing process where metals re-crystalize or liquids freeze. In the annealing process, the object is heated until it is molten, then it is gradually cooled in such a way that the metal, at any time, is nearly in thermodynamic equilibrium. As the temperature of the object is cooled, the system becomes more organized and tends to a frozen state at $T = 0$. If the cooling procedure is done unsatisfactorily or the initial temperature of the object is not adequately high, the system may turn into a meta-stable state demonstrating that the system is stuck in a local minimum energy state.

Liu et al. (2012) successfully applied the simulated annealing method in multi-criteria network path problems, while Shao and Zuo (2012) used it for higher dimensional projection depth. Milenkovic et al. (2012) successfully applied a fuzzy simulated annealing method for project time–cost trade-off, while Fonseca et al. (2012) applied simulated annealing to the high school timetabling problem. Other successful applications of simulated annealing include antenna array design in multi-input-multi-output radar (Dong et al. 2012), efficient bitstream extraction for scalable video (Wan et al. 2012), image reconstruction (Martins et al. 2012) and optimal sensor placement (Tian et al. 2012).

Simulated annealing has its origins from the work of Metropolis et al. (1953) and it comprises selecting the initial state and temperature, maintaining temperature constant, changing the initial formation, and calculating the error at the new state. If the new error is lower than the old error, then accept the new state, otherwise if the error is higher, then accept this state with a low probability. Simulated annealing substitutes a current solution with a "nearby" random solution with a probability that depends on the difference between the corresponding function values and the temperature. The temperature drops during the course of the procedure until it approaches zero and at this stage there are less random changes in the solution. Simulated annealing identifies the global optimum but it can reach infinite time in doing so. The probability of accepting the reversal is given by Boltzmann's equation (Černý 1985):

$$P(\Delta E) = \frac{1}{Z} \exp\left(-\frac{\Delta E}{T}\right) \tag{7.11}$$

Here, ΔE is the variance in error between the old and new states, T is the temperature of the system and Z is the normalization factor that guarantees that when the probability is integrated over to infinity it becomes 1.

7.6.1 Simulated Annealing Parameters

As described by Marwala (2010), applying simulated annealing means that a number of parameters and selections require to be stated: the state space, the objective function, the candidate generator process, the acceptance probability function, and the annealing temperature schedule. The selection of these parameters is important because it has an impact on the efficacy of the SA technique. Nevertheless, there is no optimal mode for selecting these parameters that will function for all problems and there is also no methodical routine of optimally selecting these parameters for a given problem. Accordingly, the selection of these parameters is mainly subjective and the technique of trial and error is extensively applied.

7.6.2 Transition Probabilities

When SA is applied, a random walk procedure is used for a given temperature. This random walk procedure involves moving from one temperature to another. The probability of moving from one state to another is called the transition probability. This probability is dependent on the current temperature, the order of producing the candidate solution, and the acceptance probability function. In this chapter, a Markov Monte Carlo (MMC) technique is applied to ensure a transition from one state to another. The MMC generates a chain of possible missing data estimates and accepts or rejects them using the Metropolis algorithm (Metropolis et al. 1953).

7.6.3 Monte Carlo Method

The Monte Carlo technique is a computational procedure that applies recurring random sampling to estimate a result (Arya et al. 2012; Klopfer et al. 2012). Jeremiah et al. (2012) applied Monte Carlo sampling for efficient hydrological model parameter optimization, while Giraleas et al. (2012) applied the Monte Carlo procedure for analysing productivity change using growth accounting and frontier-based approaches. Fang et al. (2012) applied Monte Carlo simulation for variability quantification in finite element models. Other applications of Monte Carlo simulation include evaluating reliability indices accounting omission of random repair time for distribution systems (Arya et al. 2012) and characterization and optimization of pyroelectric X-ray sources (Klopfer et al. 2012).

7.6.4 Markov Chain Monte Carlo (MCMC)

MCMC is a procedure of simulating a chain of states through a random walk. It entails a Markov process and a Monte Carlo simulation (Sheridan et al. 2012). Fishman (2012) successfully applied MCMC for counting contingency tables, while Hettiarachchi et al. (2012) successfully applied a marginalized Markov Chain Monte Carlo method for model based analysis of EEG data. Botlani-Esfahani and Toroghinejad (2012) successfully applied a Bayesian neural network and the reversible jump Markov Chain Monte Carlo Method to forecast the grain size of hot strip low carbon steels while Laloy et al. (2012) successfully applied the MCMC to analyse mass conservative three-dimensional water tracer distribution. Stošić et al. (2012) successfully applied the MCMC to optimize river discharge measurements.

If a system whose evolution is expressed by a stochastic process $\{x_1, x_2, \ldots, x_i\}$ of random variables is considered, a random variable x_i inhabits a state x at discrete time i. The list of all states that all random variables can probably occupy is known as the state space. If the probability that the system is in state x_{i+1} at time $i+1$ depends entirely on the fact that it was in state x_i at time i, then the random variables $\{x_1, x_2, \ldots, x_i\}$ form a Markov chain. For MCMC, the transition between states is attained by introducing a random noise (ε) to the current state as follows (Laloy et al. 2012):

$$x_{i+1} = x_i + \varepsilon \tag{7.12}$$

7.6.5 Acceptance Probability Function: Metropolis Algorithm

When the present state has been attained, it is either accepted or rejected. In this chapter, the acceptance of a state is conducted using the Metropolis algorithm (Metropolis et al. 1953; Shao et al. 2012; Vihola 2012; Lee et al. 2012). Zhou et al. (2012) successfully applied Metropolis-Hastings sampling for system error registration. In the Metropolis procedure, on sampling a stochastic process $\{x_1, x_2, \ldots, x_n\}$ consisting of random variables, random changes to x are introduced and are either accepted or rejected according to the following criterion:

$$\begin{aligned}
&if\ E_{new} < E_{old}\ accept\ state\ (s_{new}) \\
&else \\
&accept\ \ (s_{new})\ with\ probability \\
&\exp\{-(E_{new} - E_{old})\}
\end{aligned} \tag{7.13}$$

Here, E is the objective function.

7.6.6 Cooling Schedule

Cooling scheduling is the procedure which is followed to lower the temperature T (Stander and Silverman 1994). Natural annealing teaches us that the cooling rate should be adequately low for the probability distribution of the present state to be close to the thermodynamic equilibrium at all times during the simulation (Miki et al. 2003). The time taken for the equilibrium to be restored after a change in temperature is influenced by the shape of the objective function, the current temperature and the candidate generator. The best cooling rate should be experimentally attained for each problem. Thermodynamic, simulated annealing circumvents this problem by removing the cooling schedule and regulating the temperature at each step in the simulation based on the difference in energy between the two states, in accordance to the laws of thermodynamics (Weinberger 1990). The following cooling model is used (Salazar and Toral 1997; Marwala 2010):

$$T(i) = \frac{T(i-1)}{1+\sigma} \tag{7.14}$$

where $T(i)$ is the current temperature; $T\ (i-1)$ is the previous temperature and σ is the cooling rate. The implementation of SA is shown in Fig. 7.1 (Marwala 2010).

7.7 Experimental Investigations and Results

The methodology described below is used to analyze the manufacturing data from South Africa collected between 1992 and 2011. The variables identified for the modelling are (1) Domestic sales volumes; (2) Production volumes; (3) Number of factory workers; (4) Current stocks of raw materials in relation to planned production; (5) Business confidence; (6) Percentage rating shortage of skilled labour a constraint; and (7) Percentage rating shortage of semi-skilled labour a constraint. We then build an auto-associative network with seven input variables, four hidden nodes and seven outputs. The auto-associative network was based on the multi-layer perceptron architecture. It had a hyperbolic tangent activation function in the hidden nodes and a linear activation function in the outer layer. It assumes that the production volumes will be treated as a missing values to be estimated. The missing data estimation equation is optimized using a genetic algorithm, particle swarm optimization, and simulated annealing to identify the production volume.

When implementing a genetic algorithm, the population size was set to 20, the number of generations was set to 100, and the one point crossover with probability of crossover was set to be 0.65. The selection function which was used was the Roulette wheel and adaptive mutation with a mutation rate of 0.01. On implementing simulated annealing, the initial temperature was set to be 100. When implementing PSO, a population size of ten was set and the simulation was conducted for 500 generations. The results obtained when these optimization methods were implemented were an average of 5.3 % for GA, 5.1 % for simulated annealing, and 5.4 % for the PSO.

7.8 Conclusion

This chapter introduced auto-associative networks with genetic algorithms, particle swarm and simulated annealing optimization methods for modelling manufacturing data. The autoassociative network was created using the multi-layered perceptron. The results obtained gave marginally best results for simulated annealing, followed by genetic algorithm and the particle swarm optimization method.

References

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

Abdella M, Marwala T (2005) Treatment of missing data using neural networks. In: Proceedings of the IEEE international joint conference on neural networks, Montreal, 2005, pp 598–603

Abdella M, Marwala T (2006) The use of genetic algorithms and neural networks to approximate missing data in database. Comput Inf 24:1001–1013

Arya LD, Choube SC, Arya R, Tiwary A (2012) Evaluation of reliability indices accounting omission of random repair time for distribution systems using Monte Carlo simulation. Int J Electr Power Energy Syst 42:533–541

Banzhaf W, Nordin P, Keller R, Francone F (1998) Genetic programming-an introduction: on the automatic evolution of computer programs and its applications. Morgan Kaufmann, San Francisco

Botlani-Esfahani M, Toroghinejad MR (2012) Application of a Bayesian artificial neural network and the reversible jump Markov chain Monte Carlo method to predict the grain size of hot strip low carbon steels. J Serb Chem Soc 77:937–944

Černý V (1985) Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J Optim Theory Appl 45:41–51

Crossingham B, Marwala T (2008) Using genetic algorithms to optimise rough set partition sizes for HIV data analysis. Adv Intell Distrib Comput Stud Comput Intell 78:245–250

Dervilis N, Barthorpe R, Antoniadou I, Staszewski WJ, Worden K (2012) Damage detection in carbon composite material typical of wind turbine blades using auto-associative neural networks. In: Proceedings of SPIE – the international society for optical engineering, San Diego, 2012, art. no. 834806

Dindar ZA, Marwala T (2004) Option pricing using a committee of neural networks. Proc IEEE Int Conf Syst Man Cybern 1:434–438

Dong J, Yang J, Lei W, Shi R, Guo Y (2012) Antenna array design in MIMO radar using cyclic difference sets and simulated annealing. In: Proceedings of the international conference on microwave and millimeter wave technology, Shenzhen, China, pp 237–240

Engebrecht AP (2005) Fundamentals of computational swarm intelligence. Wiley, New York

Fang S-E, Ren W-X, Perera R (2012) A stochastic model updating method for parameter variability quantification based on response surface models and Monte Carlo simulation. Mech Syst Signal Process 33:83–96

Farzi S, Shavazi AR, Pandari AR (2013) Using quantum-behaved particle swarm optimization for portfolio selection problem. Int Arab J Inf Technol 10:art. no. 2/7-2761

Fishman GS (2012) Counting contingency tables via multistage Markov chain Monte Carlo. J Comput Graph Stat 21:713–738

Fonseca GHG, Brito SS, Santos HG (2012) A simulated annealing based approach to the high school timetabling problem. Lect Note Comput Sci 7435:540–549

Gholizadeh S, Fattahi F (2012) Design optimization of tall steel buildings by a modified particle swarm algorithm. Struct Design Tall Spec Build. doi:10.1002/tal.1042

Giraleas D, Emrouznejad A, Thanassoulis E (2012) Productivity change using growth accounting and frontier-based approaches – evidence from a Monte Carlo analysis. Eur J Oper Res 222:673–683

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading

Goldberg DE (2002) The design of innovation: lessons from and for competent genetic algorithms. Addison-Wesley, Reading

Granville V, Krivanek M, Rasson J-P (1994) Simulated annealing: a proof of convergence. IEEE Trans Pattern Anal Mach Intell 16:652–656

Hettiarachchi I, Mohamed S, Nahavandi S (2012) A marginalised Markov chain Monte Carlo approach for model based analysis of EEG data. In: Proceedings of the international symposium on biomedical imaging, Barcelona, 2012, pp 1539–1542

Hlalele N, Nelwamondo FV, Marwala T (2009) Imputation of missing data using PCA, neuro-fuzzy and genetic algorithms. Lect Note Comput Sci 5507:485–492

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor

Hulley G, Marwala T (2007) Genetic algorithm based incremental learning for optimal weight and classifier selection. Comput Model Life Sci Am Inst Phys Ser 952:258–267

Jeremiah E, Sisson SA, Sharma A, Marshall L (2012) Efficient hydrological model parameter optimization with sequential Monte Carlo sampling. Environ Model Softw 38:283–295

Kalatehjari R, Ali N, Hajihassani M, Kholghi Fard M (2012) The application of particle swarm optimization in slope stability analysis of homogeneous soil slopes. Int Rev Model Simul 5:458–465

Karabulut M, Ibrikci T (2012) A Bayesian scoring scheme based particle swarm optimization algorithm to identify transcription factor binding sites. Appl Soft Comput J 12:2846–2855

Kennedy J (1997) The particle swarm: social adaptation of knowledge. In: Proceedings of IEEE international conference on evolutionary computation, Piscataway, 1997, pp 303–308

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Piscataway, 1995, pp 1942–1948

Kennedy J, Eberhart RC (2001) Swarm intelligence. Morgan Kaufmann, San Francisco

Kim JK, Shin DW (2012) The factoring likelihood method for non-monotone missing data. J Korean Stat Soc 41:375–386

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Klopfer M, Wolowiec T, Satchouk V, Alivov Y, Molloi S (2012) Characterization and optimization of pyroelectric X-ray sources using Monte Carlo spectral models. Nucl Instrum Method Phys Res Sec A Accel Spectrom Detect Assoc Equip 689:47–51

Kramer MA (1992) Autoassociative neural networks. Comput Chem Eng 16:313–328

Laloy E, Linde N, Vrugt JA (2012) Mass conservative three-dimensional water tracer distribution from MCMC inversion of time-lapse GPR data. Water Resour Res. doi:10.1029/2011WR011238 (in press)

Lee C-H, Xu X, Eun DY (2012) Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. Perform Eval Rev 40:319–330

Leke B, Marwala T, Tim T, Lagazio M (2006) Using genetic algorithms versus line search optimization for HIV predictions. Trans Inf Sci Appl 4:684–690

Lemma TA, Hashim FM (2012) Wavelet analysis and auto-associative neural network based fault detection and diagnosis in an industrial gas turbine. In: Proceedings of the IEEE business, engineering and industrial applications colloquium, Kuala Lumpur, 2012, pp 103–108

Liu L, Mu H, Luo H, Li X (2012) A simulated annealing for multi-criteria network path problems. Comput Oper Res 39:3119–3135

Makki B, Hosseini MN (2012) Some refinements of the standard autoassociative neural network. Neural Comput Appl. doi:10.1007/s00521-012-0825-5 (in press)

Markowitz H (1952) Portfolio selection. J Finance 7:77–91

Martins TDC, De Camargo EDLB, Lima RG, Amato MBP, Tsuzuki MDSG (2012) Image reconstruction using interval simulated annealing in electrical impedance tomography. IEEE Trans Biomed Eng 59:1861–1870

Marwala T (2002) Finite element updating using wavelet data and genetic algorithm. Am Inst Aeronaut Astronaut J Aircraft 39:709–711

Marwala T (2004) Control of complex systems using Bayesian neural networks and genetic algorithm. Int J Eng Simul 5:28–37

Marwala T (2005) Finite element model updating using particle swarm optimization. Int J Eng Simul 6:25–30

Marwala T (2007) Bayesian training of neural network using genetic programming. Pattern Recognit Lett 28:452–1458

Marwala T (2009) Computational intelligence for missing data imputation, estimation and management: knowledge optimization techniques. IGI Global Publications, New York

Marwala T (2010) Finite element model updating using computational intelligence techniques. Springer, London

Marwala T (2012) Condition monitoring using computational intelligence methods. Springer, London

Marwala T, Chakraverty S (2006) Fault classification in structures with incomplete measured data using autoassociative neural networks and genetic algorithm. Curr Sci 90:542–548

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

Marwala T, de Wilde P, Correia L, Mariano P, Ribeiro R, Abramov V, Szirbik N, Goossenaerts J (2001) Scalability and optimisation of a committee of agents using genetic algorithm. In: Proceedings of the international symposium on soft computing and intelligent systems for industry, Florida, USA, arXiv 0705.1757

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087

Miki M, Hiroyasu T, Wako J, Yoshida T (2003) Adaptive temperature schedule determined by genetic algorithm for parallel simulated annealing. In: Proceedings of the congress on evolutionary computation, Canberra, Australia, pp 459–466

Milenkovic M, Bojovic N, Ribeiro RA, Glisovic N (2012) A fuzzy simulated annealing approach for project time–cost tradeoff. J Intell Fuzzy Syst 23:203–215

Miranda V, Castro ARG, Lima S (2012) Diagnosing faults in power transformers with autoassociative neural networks and mean shift. IEEE Trans Power Deliv 27:1350–1357

Mohamed S, Tettey T, Marwala T (2006) An extension neural network and genetic algorithm for bearing fault classification. In: Proceedings of the IEEE international joint conference on neural networks, Vancouver, Canada, pp 7673–7679

Moreau V, Bage G, Marcotte D, Samson R (2012) Statistical estimation of missing data in life cycle inventory: an application to hydroelectric power plants. J Clean Prod 37:335–341

Muthukaruppan S, Er MJ (2012) A hybrid particle swarm optimization based fuzzy expert system for the diagnosis of coronary artery disease. Expert Syst Appl 39:11657–11665

Nasir M, Das S, Maity D, Sengupta S, Halder U, Suganthan PN (2012) A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. Inf Sci 209:16–36

Nelwamondo FV (2008) Computational intelligence techniques for missing data imputation. Ph.D. thesis, University of the Witwatersrand, Johannesburg

Nelwamondo FV, Marwala T (2007) Rough set theory for the treatment of incomplete data. In: Proceedings of the IEEE conference on fuzzy systems, London, UK, pp 338–343

Palmé T, Breuhaus P, Assadi M, Klein A, Kim M (2011) Early warning of gas turbine failure by nonlinear feature extraction using an auto-associative neural network approach. Proc ASME Turbo Expo 3:293–304

Patel PB, Marwala T (2009) Genetic algorithms, neural networks, fuzzy inference system, support vector machines for call performance classification. In: Proceedings of the IEEE internationall conference on machine learning and applications, Florida, USA, pp 415–420

Ransome TM, Rubin DM, Marwala T, de Kok EA (2005) Optimising the verification of patient positioning in proton beam therapy. In: Proceedings of the IEEE 3rd international conference on computational cybernetics, Mauritius, 2005, pp 279–284

Rey-del-Castillo P, Cardeñosa J (2012) Fuzzy min-max neural networks for categorical data: application to missing data imputation. Neural Comput Appl 21:1349–1362

Salazar R, Toral R (1997) Simulated annealing using hybrid Monte Carlo. J Stat Phys 89(5/6):1047–1060

Shao W, Zuo Y (2012) Simulated annealing for higher dimensional projection depth. Comput Stat Data Anal 56:4026–4036

Shao W, Guo G, Meng F, Jia S (2012) An efficient proposal distribution for metropolis-hastings using a B-splines technique. Comput Stat Data Anal 57:465–478

Sheridan P, Yagahara Y, Shimodaira H (2012) Measuring preferential attachment in growing networks with missing-timelines using Markov chain Monte Carlo. Physica A Stat Mech Appl 391:5031–5040

Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings of IEEE international conference on evolutionary computation, Anchorage, Alaska, pp 69–73

Sivaram GSVS, Ganapathy S, Hermansky H (2010) Sparse auto-associative neural networks: theory and application to speech recognition. In: Proceedings of the 11th annual conference of the international speech communication association, Florence, Italy, pp 2270–2273

Stander J, Silverman BW (1994) Temperature schedules for simulated annealing. Stat Comput 4:21–32

Stošić BD, Santos Silva JR, Filho MC, Barros Cantalice JR (2012) Optimizing river discharge measurements using Monte Carlo Markov chain. J Hydrol 450–451:199–205

Tettey T, Marwala T (2006) Controlling interstate conflict using neuro-fuzzy modeling and genetic algorithms. In: Proceedings of the 10th IEEE international conference on intelligence engineering systems, London, UK, pp 30–44

Tian L, Chen H-G, Zhu J, Zhang L-S, Chen W-H (2012) A study of optimal sensor placement based on the improved adaptive simulated annealing genetic algorithms. J Vib Eng 25:238–243

Tsai LT, Yang C-C (2012) Improving measurement invariance assessments in survey research with missing data by novel artificial neural networks. Expert Syst Appl 39:10456–10464

Vihola M (2012) Robust adaptive Metropolis algorithm with coerced acceptance rate. Stat Comput 22:997–1008

Wan S, Yang K, Zhou H (2012) Efficient bitstream extraction for scalable video based on simulated annealing. Concurr Comput Pract Exp 24:1223–1230

Wang C, Yang Y (2011) Robust face recognition from single training image per person via auto-associative memory neural network. In: Proceedings of the international conference on electrical and control engineering, Beijing, China, pp 4947–4950

Weinberger E (1990) Correlated and uncorrelated fitness landscapes and how to tell the difference. Biol Cybern 63:325–336

Zhou L, Liang Y, Pan Q (2012) System error registration based on Metropolis-Hastings sampling. Syst Eng Electron 34:433–438