Zimbabwe Journal of Science & Technology pp. 54 - 68

Vol. 10 [2015]
e-ISSN 2409-0360
Zimbabwej.sci.technol

CLIMATE VARIABILITY FORECASTING USING BAT ALGORITHM OPTIMISED ARTIFICIAL NEURAL NETWORK

[1] **K. Mzelikahle,** [2]**N. Kokera,** [3]**K.R. Chilumani**
[1,3] National University of Science and Technology, P. O. Box AC 939, Ascot, Bulawayo
[2] Zimbabwe Open University, Bulawayo Regional Centre, Box 3550, Bulawayo
Correspondence: kernan.mzelikahle@nust.ac.zw
k.mzelikahle@gmail.com

Abstract

This paper presents a summary and results of a study that was conducted in an attempt to forecast climate variability in Zimbabwe using the BAT Algorithm optimised Artificial Neural Network (BAT-ANN) analysis technique. Forecasts of climate ahead of time can potentially allow governments, farmers and other players in private and/or public sectors to make decisions to reduce unwanted impacts or take advantage of expected favourable climate. However, potential benefits of climate forecasts vary considerably because of many physical, biological, economic, social, and political factors. In a developing country, like Zimbabwe where agriculture is the base of the national economy, climate conditions play leading role for progressive and sustainable development, therefore climate variability forecasts are very important. The BAT-ANN was adapted and tested using the Zimbabwean meteorological dataset and results confirm that our proposed model has the potential for reliable climate forecasting for a 25 year period. The mean percentage accuracy was used to evaluate the performance of the trained climate forecasting neural network and proved sufficient. Therefore, in this paper, we present a new technique to climate variability assessment namely; the BAT-ANN. In this study, the approach employed to achieve objectives was; collecting quantitative data, adapting a BAT-ANN for analysis, and developing a Java program that employs the BAT-ANN for forecasting. The objectives of the study were met. [1]

KEYWORDS — BAT Algorithm, Climate Variability, Artificial Neural Network, Network Optimisation, Forecasting.

1. INTRODUCTION

WEATHER is the fluctuating state of the atmosphere around us, characterised by the temperature, wind, precipitation, clouds and other atmospheric elements. Climate, on the other hand, refers to the average weather in terms of the mean and its variability over a certain timespan and a certain area. Climate varies from place to place, all depending on latitude, distance from the sea, vegetation, presence or absence of mountains and other geographical factors. Climate varies also in time; from season-to-season, year-to-year, decade-to-decade or on much longer time-scales (Aguado and Burt, 2010). In relation to climate, climate variability is the way climate fluctuates yearly above or below a known long-term average value. Climate has a large impact on human health and the well-being of communities throughout the world; therefore research in this field has a high priority in many countries, including Zimbabwe. The ability to understand, monitor and predict this climatic variability provides an opportunity to put historical experiences into perspective and to evaluate alternative strategies for making quality decisions. Due to the chaotic nature of the atmosphere, massive computational power is required to estimate the error involved in measuring the initial conditions of the atmosphere, given there is usually an incomplete understanding of atmospheric processes. This means that forecasts become less accurate as the difference in current time and the time for which the forecast is being made increases (Subhajini, 2011).

[1]This document is written in British English.

As a result, ensembles and models that help to narrow the weather forecasting error and pick the most likely outcome are being adopted in weather and climate forecasting. These are largely mathematical algorithmic models that imitate the neural network behaviour characteristics of the brain and they carry out parallel distributed information processing. The mathematical algorithmic models have emerged as excellent tools for deriving data oriented models because of their inherent characteristic of plasticity that permits the adaptation of the learning task when data is provided (Deng et al, 2008).

In this repute, a meta-heuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space (Saka and Dogan, 2012). Meta-heuristic algorithms are designed to produce solutions to complex optimisation problems where other optimisation methods have failed to be either effective or efficient. Furthermore, a good meta-heuristic implementation is likely to provide near optimal solutions in reasonable computation times (Glover and Kochenberger, 2003). Therefore, the Bat Algorithm (BA) is identified as a meta-heuristic algorithm of choice because it has superior performance as compared to other bio-inspired meta-heuristic algorithms. Table 1 shows a comparison of the Bat Algorithm against Genetic Algorithms (GA), and Particle Swarm Optimisation (PSO). From Table 1, it can be seen that PSO performs much better than genetic algorithms, while the Bat Algorithm is much superior to both algorithms in terms of accuracy and efficiency.

Table 1. Comparison of BA with GA, and PSO (Yang, 2010)

	GA	PSO	BA
Multiple peaks	$52124 \pm 3277(98\%)$	$3719 \pm 205(97\%)$	$1152 \pm 245(100\%)$
Michalewicz's (d=16)	$89325 \pm 7914(95\%)$	$6922 \pm 537(98\%)$	$4752 \pm 753(100\%)$
Rosenbrock's (d=16)	$55723 \pm 8901(90\%)$	$32756 \pm 5325(98\%)$	$7923 \pm 3293(100\%)$
De Jong's (d=256)	$25412 \pm 1237(100\%)$	$17040 \pm 1123(100\%)$	$5273 \pm 490(100\%)$
Schwefel's (d=128)	$227329 \pm 7572(95\%)$	$14522 \pm 1275(97\%)$	$8929 \pm 729(99\%)$
Ackley's (d=128)	$32720 \pm 3327(90\%)$	$23407 \pm 4325(92\%)$	$6933 \pm 2317(100\%)$
Rastrigin's	$110523 \pm 5199(77\%)$	$79491 \pm 3715(90\%)$	$12573 \pm 3372(100\%)$
Easom's	$19239 \pm 3307(92\%)$	$17273 \pm 2929(90\%)$	$7532 \pm 1702(99\%)$
Griewangk's	$70925 \pm 7652(90\%)$	$55970 \pm 4223(92\%)$	$9792 \pm 4732(100\%)$
Shuberts (18 minima)	$54077 \pm 4997(89\%)$	$23992 \pm 3755(92\%)$	$11925 \pm 4049(100\%)$

1.1 The Bat Algorithm

THE bat algorithm (BA) was developed by Xin-She Yang in 2010. It is a meta-heuristic algorithm for solving many optimisation problems. It is based on the echolocation behaviour of bats. Echolocation is an advanced hearing based navigation system used by bats and some other animals to detect objects in their surroundings by emitting a sound to the environment (Yang, 2010). While they are hunting for prey or navigating the atmosphere, bats produce a sound wave that travels across the canyon and eventually hits an object or a surface and return to them as an echo. The sound waves travel at a constant speed in zones where atmospheric air pressure is identical but may change if the atmospheric pressure changes depending on the density of the air (Yang, 2011). By following the time delay of the returning sound, bats can determine the precise distance to circumjacent objects. Further, the relative amplitudes of the sound waves received at each individual ear are used to identify shape and direction of the objects. The information collected in this way is synthesised and processed in the brain to depict a mental image of their surroundings (Yang, 2011).

Yang (2008) simulated echolocation behaviour of bats and its associated parameters in a numerical optimisation algorithm. They can locate the exact position of their prey or food and also they can determine the different types of insects within their surrounding even in a complete darkness. They emit a loud sound pulse and detect an echo that comes back from their surrounding objects and from that response of the atmospheric conditions, they can easily locate their prey. While searching for their prey, their loudness is loudest when they are far away from the prey and the loudness lowers when they are nearer to the prey. Now for emission and detection of echo which are generated by them, they use the concept of time delay. As a concept and practical hunting operation of

the bats, the time delay is measured as the distance between the two ears of the bats and the loudness variation of echoes is used to determine the distance to the targeted prey (Yang, 2010). Equations (1), (2) and (3) are used in Yangs's (2010) model when updating the frequency, velocity and solution bat on echolocation:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (1)$$

$$v_i{}^{t+1} = v_i{}^t + (x_i{}^t - x_*)f_i \quad (2)$$

$$x_i{}^{t+1} = x_i{}^t + v_i{}^t \quad (3)$$

where x_i is the position, v_i is the velocity, f_i is the frequency, and $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution. In this case, it is the current global best location (solution) which is located after comparing all the solutions among all bats. The frequency factor controls step size of a solution in the algorithm. This factor is assigned to a random value for each bat (solution) between upper and lower boundaries f_{min} and f_{max}. Velocity of a solution is proportional to frequency and the new solution depends on its new velocity. The loudness decreases when a bat has located its prey or food but the rate of pulse emission increases. For simplicity these parameters are regarded as binary operators in Yang's (2010) model. This means that a bat that has found their prey would stop producing any more sound. Below is a pseudo-code listing by Yang (2010).

Listing 1. Bat Algorithm Pseudo-Code (Yang, 2010)

```
Objective function f(x), x=(x_1,
   ..., x_d)^T
Initialise the bat population x_i(i
   =1,2,....,n) and v_i
Define pulse frequency, f_i
Initialise pulse rate and the
   loudness
While (t < Max number of iterations
   )
  Generate new solutions by
   adjusting frequency,
  and updating velocities and
   location/solutions
  [using eqn. 1 and 2]
  if (rand > r_i)
    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_i and f(x_i) < f(x_
   *))
    Accept the new solutions
    Increase r_i and reduce A_i
  end if
  Rank the bats and find the
   current best
end while
Postprocess results and
   visualisation
```

The standard bat algorithm has many advantages, and one of the key advantages is that it can provide very quick convergence at a very initial stage by switching from exploration to exploitation. This makes it an efficient algorithm for applications such as classifications and others when a quick solution is needed (Yang, 2011). However, if the algorithm is allowed to switch to exploitation stage too quickly by varying the amplitude and the pulse rate too quickly, it may lead to stagnation after some initial stage. In order to improve the performance, many methods and strategies have been attempted to increase the diversity of the solution and thus to enhance the performance. A few good variants of the bat algorithm have thus far been developed and some of the Bat Algorithm variants include the following:

- *Fuzzy Logic Bat Blgorithm (FLBA)* that was presented by Khan et al., in 2011. This algorithm is a combination of fuzzy logic and Bat algorithm, this was done by introducing the fuzzy logic into the bat algorithm.
- *The Multi-objective Bat algorithm (MOBA)* that was presented by Yang in 2011 as an extension of the Bat algorithm in order to deal with multi-objective optimisation. This algorithm demonstrated its effectiveness for solving design benchmarks in engineering.
- *The K-Means Bat algorithm (KMBA)* that was presented by Komarasamy and Wahi in 2013 is the combination of K-means and bat algorithm for efficient clustering.
- *The Binary Bat Algorithm (BBA)* that was presented by Nakamura et al., in 2012 for the purpose of solving classifications and feature selection problems.

- *The Improved Bat algorithm (IBA)* that was presented by Jamil et al., in 2013. This algorithm was extended with a good combination of Levy flights and subtle variations of loudness and pulse emission rates.

1.2 The Neural Network Model

A neural network is a powerful data modelling tool that is able to capture and represent complex input and output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform intelligent tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways:

- A neural network acquires knowledge through learning.
- A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

The true power and advantage of neural networks lies in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modelled. Traditional linear models are simply inadequate when it comes to modelling data that contains non-linear characteristics (Cheung and Cannons, 2002). An artificial neuron is a unit that performs a simple mathematical operation on its inputs and imitates the functions of biological neurons and their unique process of learning. An Artificial Neural Network is composed of a huge number of highly interconnected processing elements (neurons) working together to solve specific problems. In this research the focus is on a Multi-Layer Perceptron (MLP) network, which is the most popular and widely used Artificial Neural Network paradigm in many applications including forecasting. The MLP networks are used in a variety of problems especially in forecasting because of their inherent capability of arbitrary input-output mapping. An MLP is typically composed of several layers of nodes. The first or the lowest layer is an input layer where external information is received. The last or the highest layer is an output layer where the problem solution is obtained. The input layer and output layer are separated by one or more intermediate layers called hidden layers. The nodes in adjacent layers are usually fully connected by acyclic arcs from a lower layer to a higher layer (Deng et al., 2008). For a forecasting problem, the inputs to an ANN are usually independent variables. The functional relationship estimated by the ANN can be written as:

$$y = f(x_1, x_2, ..., x_p) \tag{4}$$

where x is an input vector, $x_1, x_2, ..., x_p$ are p independent variables and y is an output which is a dependent variable. In this sense, the neural network is functionally equivalent to a non-linear regression model. On the other hand, for an extrapolative or time series forecasting problem, the inputs are typically the past observations of the data series and the output is the future value. The Artificial Neural Network (ANN) performs the following function mapping:

$$y_{(t+1)} = f(y_1, y_{(t-1)}, ..., y_{(t-p)}) \tag{5}$$

where y_1 is the observation at time t (Deng et al., 2008). Figure 1 shows an example of a multilayer perceptron (MLP).

Before an ANN can be used to perform any desired task, it must be trained to do so. Basically, training is the process of determining the arc weights which are the key elements of an ANN. The knowledge learned by a network is stored in the arcs and nodes in the form of arc weights and node biases. It is through the linking arcs that an ANN can carry out complex non-linear mappings from its input nodes to its output nodes. In this research, the aim is to gather data sets consisting weather parameters namely; temperature, humidity, pressure and precipitation, and perform all required data pre-processing tasks. Thus the neural network is trained by updating all the weights and biases as per the obtained errors in each iteration. To improve the performance, the Bat Algorithm was used in training the neural network.

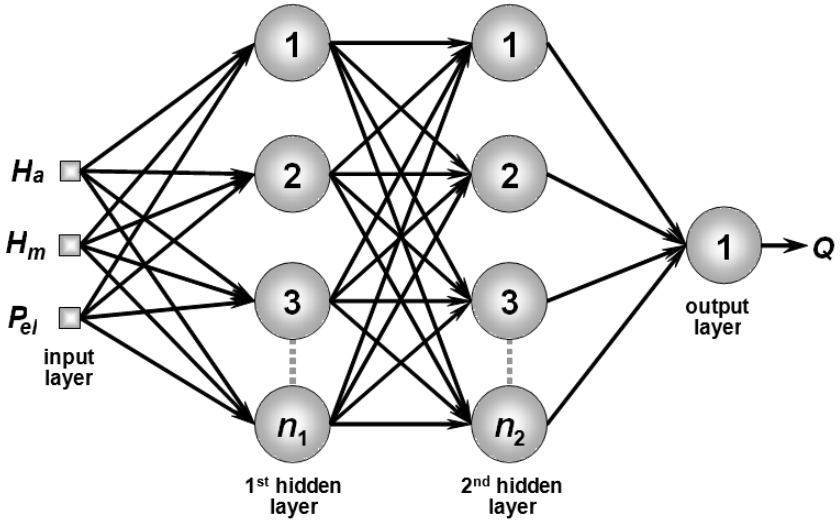

Figure 1. MLP neural network (Deng et al., 2008)

1.3 Weather Prediction using the Neural Network model

Maqsood et al., (2006), examined the applicability of Hop field Model (HFM) for weather forecasting in southern Saskatchewan in Canada. The model performance was contrasted with multi-layered perceptron network (MLPN). The data of temperature, wind speed and relative humidity were used to train and test the two models. With each model, 24 hour weather forecasts were made for winter, spring, summer and fall seasons and the results were discovered to be accurate. Moreover, ensembles of these models were generated by choosing the best values among the two predicted outputs that were closest to the actual values. Performance and reliabilities of the models were then evaluated by a number of statistical measures. The results indicate that the HFM was relatively less accurate for the weather forecasting problem. In comparison, the ensembles of neural networks produced the most accurate forecasts. The MLPN achieved useful weather forecasting results in an efficient way compared to the Hop Field modelling results, the MLPN exhibited lower errors. It was discovered that it is better capable of representing non-linear functions than the single layered perceptron. The back propagation algorithm was used to train the MLPN, however, the learning process of the MLPN is time-consuming and the algorithm's performance is heavily dependent on the network parameters such as the learning rate and momentum. Therefore, in this research the training is improved by optimising the ANN using the bat algorithm.

Devi et al., (2012) developed a neural network for predicting temperatures. They used the Back Propagation Neural Network (BPN) technique for training their network. The main advantage of the BPN neural network method was that it could fairly approximate a large class of functions and their model had potential to capture the complex relationships between many factors that contributed to certain temperature lurching. This proposed idea was tested using real time datasets and the results were compared with practical workings of meteorological department. These results confirmed that the trained neural network could predict the future temperature with less error. Their research model proved to be better than numerical models used but it could still produce errors and as a result inaccurate predictions could be obtained in the future. In this study the effectiveness of the training process is improved by using the BAT Algorithm in place of and as an optimisation of raw back-propagation.

Harold et al., (1998) developed a neural network, using input from the ETA Model and upper air soundings for the probability of precipitation (PoP) and quantitative precipitation forecast (QPF) for the DallasFort Worth, Texas, area. Forecasts from two years were verified against a network of 36 rain gauges. The resulting forecasts were remarkably sharp, of the 436 days with forecasts of less than 5% PoP, no rain occurred on 435 days. On the 111 days with forecasts of greater than 95% PoP, rain always occurred. The linear correlation between the forecast and observed precipitation amount was 0.95. As a result, the system indicated a potential for more ac-

curate precipitation forecasting. The neural network produced a remarkably good forecast of both the probability and amount of precipitation for the DallasFort Worth area. This research was not accurate for only one day of the 436 days where the forecasts predicted no rain.

Table 2. Sample Data, Bulawayo 2005

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
Temperature	29.5	30.2	28.8	27.7	26.5	24.8	22.3	28.3	29.8	31.9	30.6	25.8
Pressure	166.3	167.1	168.1	169.5	171.0	172.1	174.5	170.6	170.1	168.9	166.7	166.4
Humidity	60	57	60	59	49	48	53	39	32	37	55	81
Rainfall	101.0	21.1	29.7	0.8	3.9	0.0	0.0	0.0	0.0	0.0	79	237.3

2. METHODS

THE Design and Creation Research Methodology was used in this research. This was appropriate because a software artifact had to be developed. Further there was need for applicability in situations facing limited resources yet requiring both practical and technological contributions. The Design and Creation methodology was divided into two levels namely; Data Acquisition and Analysis, and Forecasting. Of these two levels, subtasks were identified and are discussed in brief below:

2.1 Data Acquisition

At this stage, the problem was articulated and the sources of relevant data identified. Data was collected largely from the Meteorological Service Department of Zimbabwe. The data was received in raw format and provided the Temperature, Pressure, Humidity, and Rainfall for all districts, grouped into provinces, in Zimbabwe for the period 1985 to 2013. Table 2 shows a sample of data that was acquired from the Zimbabwe Meteorological Service Department.

2.2 Data Analysis

For the climate variability problem, the chosen time series data was rainfall, temperature, humidity and pressure at particular regions. As is the case with many neural-network applications, pre-processing the inputs and the outputs improves the results significantly and was done. In this work, input and output pre-processing means extracting features from the inputs and transforming the target outputs in a way that makes it easier for the network to extract useful information from the inputs and associate it with the required outputs. This entails stationarising data, and statistical data cleaning for inputs.

2.3 ANN Optimisation

The training process applied in this research is as follows: First, input patterns of the training set are entered into the input nodes. The activation values of the input nodes are then weighted and accumulated at each node in the first hidden layer. The total is then transformed by an activation function into the node's activation value. It, in turn, becomes an input into the nodes in the next layer, until eventually the output activation values are found. The training algorithm is used to find the weights that minimise some overall error measure such as the sum of squared errors (SSE) and mean squared errors. In this study, the MLP training was supervised in that the desired response of the network (target value) for each input pattern was available. In addition to historical time series data, utilised as inputs, the values and forecasts of the neural network are correlated to the relationship with the series of known data. In this stage the development tools used were; Octave (high level interpreted language) and Java programming language on a Linux platform.

2.4 Forecasting

The neural network was evaluated under various criteria which include; functionality, completeness, consistency, accuracy, performance and reliability. In this research, the problem of forecasting was approached by adjusting ANN training methods using BAT Algorithm optimisation. The inputs to the multilayer network were chosen as the "previous" values and the output as the forecast. The network was trained and tested on sufficiently large training and testing sets that were extracted from meteorological historical time series data.

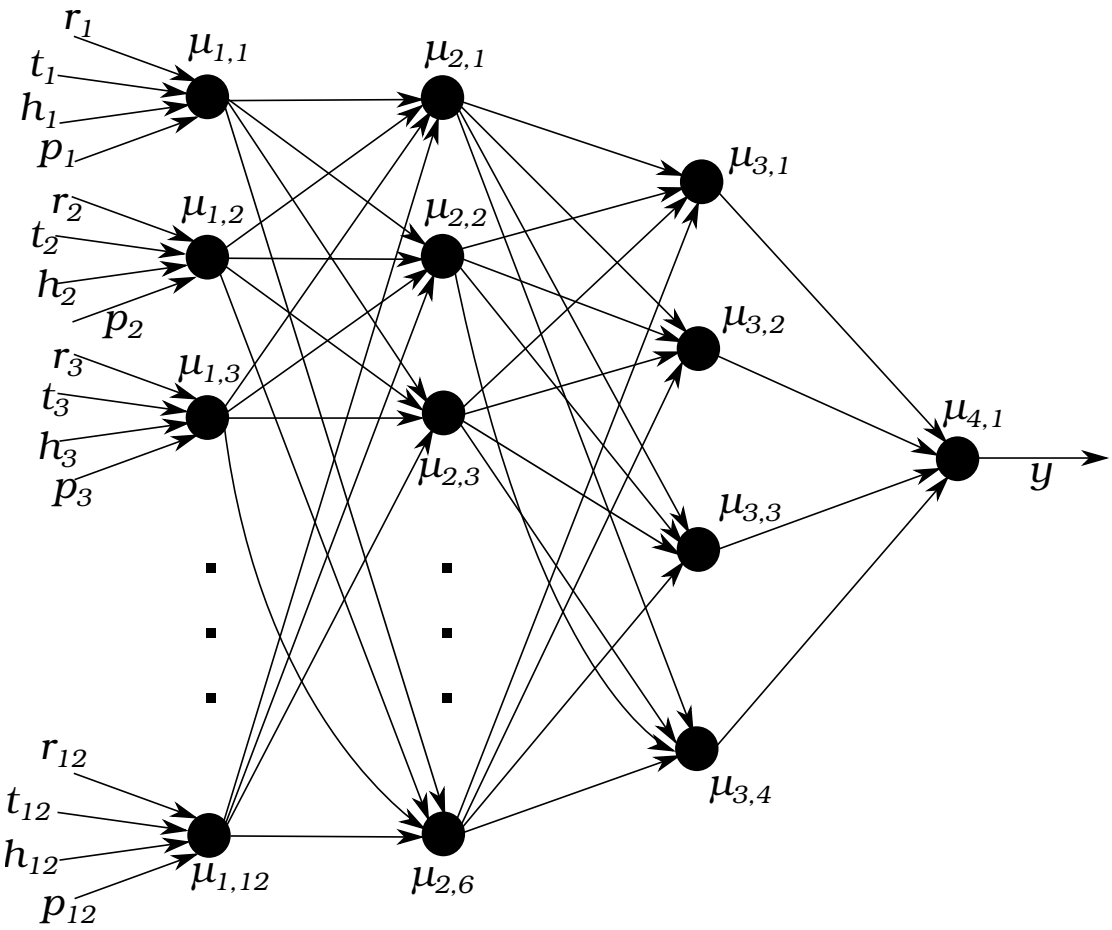

Figure 2. Bat Optimised ANN

2.5 BAT Optimised ANN

The objective of this stage was to adapt and optimise an Artificial Neural Network (ANN) using the Bat Algorithm. The approach used in this study was to adapt the Bat-ANN optimisation as was presented by Ramawan et al. in 2014. The structure of the adapted ANN is shown in Figure 2, and the steps taken were as follows:

- *Data Normalisation:* After collecting data, it was normalised so that it fell in range $[0, 1]$ in order to satisfy standard network training practices. Normalisation procedure before presenting the input data to the network is generally a good practice, since mixing variables with large magnitudes and small magnitudes will confuse the learning algorithm on the importance of each variable and may force it to finally reject the variable(s) with smaller magnitudes. In this research we used the neural network toolbox function in Octave called `prestd` which pre-processed the network training set by normalising the inputs and targets so that they have means of zero (0) and standard deviations of one (1) and the `poststd` in order to denormalise the pre-processed data to its original format, whenever the need arose. Data normalisation was performed at the beginning of the training process. The central idea behind such careful data normalisation was to remove the dependence on measurement units, which were directly relevant to weather forecasting since predictor variables were measured using different units.

- *Creating an Optimised ANN:* At this stage we began by adapting the Bat Algorithm into Octave and created a multilayer neural network, specifying the number of hidden layers, neurons in each layer, transfer function in each layer, training function and the input layer. The creation phase involved programming the ANN model. For this purpose GNU Octave 3.4.3 was used. Octave is an interactive programming language specifically suited for vectorisable numerical calculations. The Neural Network Toolbox contains the octave tools for

designing, implementing and simulating neural networks. In preparation for training the neural network, the input vectors and the target vectors were randomly divided into three sets as follows 70% for training set, 15% for validation set and 15% for testing set. Training of the neural network was based on two distinct sets, namely; the training set and the test set. The training set was used for computing the gradient and updating the neural network weights and biases. The test set consisted of patterns that are used to estimate how well the neural network performs.

- *Train Neural Network:* The neural network was trained with actual data of the past 22 years obtained from the meteorological department. The years range from 1985 to 2007. During the training process, the weights were adjusted in order to make the outputs (forecast values) close to the target outputs (measured) of the network. Supervised training was used, and involves a deliberate mechanism of providing the network with the desired outputs for given inputs. In supervised training, both the inputs and the outputs are provided. In this process, there was a training set that included the train-input (input) and the train-output (target) with 176 patterns each. This type of training required many samples which acted as examples for the neural network. For each input fed into the neural network the desired result or output of the network was specified. Each output that the network had obtained was compared to the desired output (target) and if an undesired output had been achieved the connection weights of the input had to be altered in order for the network to function at optimal efficiency. This was done to minimise errors within the results of the network. Errors were then transferred back through the system, causing the system to modify the weights which controlled the network. This process occurred over and over as the weights were adjusted to finally be optimal.

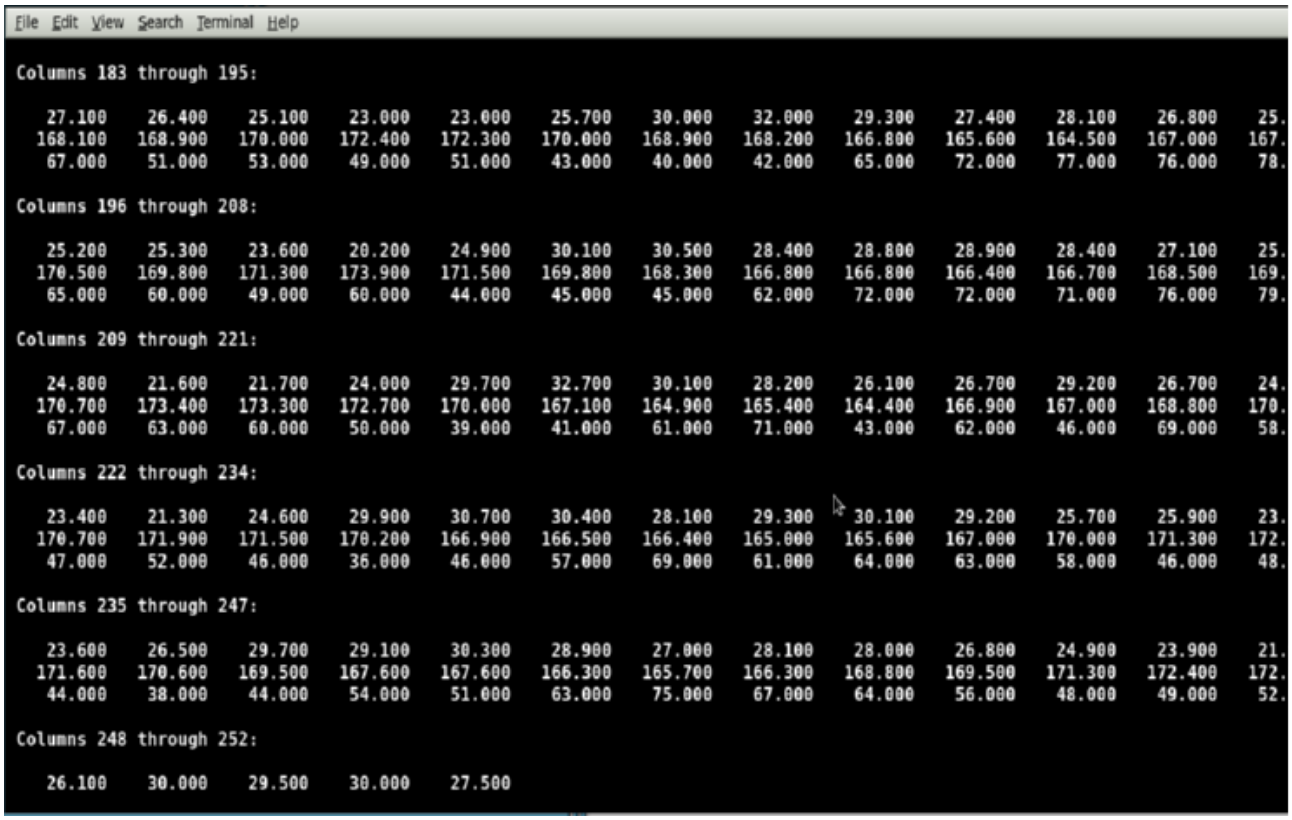

Figure 3. Input Matrix

2.6 ANN Implementation

The `newff` function in Octave creates a feed forward neural network. The following code snippet was used for neural network creation:

```
NET = newff(PR, SS, TRF, BTF, BLF,
                PF);
```

where `NET` is the created neural network created by `newff`; `PR - R x 2` matrices of min and max values for R input elements, `SS 1x` N_i row vector with size of i^{th} layer for N layers. `TRF` is the transfer function of the i^{th} layer, `BTF` is batch network training function, `BLF` being the batch weight/bias learning function and `PF` is the performance function. The input to the neural network has to be in a matrix form, so the first step in implementing the solution was to create a matrix of figures. Two integer matrices called Input (with 252 patterns) and Target (with 252 columns) were created to hold the weather parameters, Temperature, Pressure, Rainfall and Humidity for 22 years. These were used for the creation of the neural network. At this stage data was further subdivided into 3 sets; Training set,Validation set and Test set. Figure 3 shows the Input matrix.

A number of scaling methods were considered in preparing the data for training and testing. Since the neural network employed a sigmoidal activation function, the output of the network would be constrained to the unscaled range of $[0, 1]$. In order to compare this output with desired values, it was thus necessary to scale the target values to the same range. Hence the weather data was scaled to within a range of $[0, 1]$ to prevent the neural network weights from having to increase in magnitude to excessively large positive or negative values. As stated earlier, input patterns were normalised or preprocessed using the `prestd` function as shown below:

```
[pn,meanp,stdp,tn,meant,stdt] =
prestd(Traininput,Trainoutput)
```

where `Traininput` is the matrix of the input vectors (columns), `Trainoutput` is the matrix of the target vector, `pn` is the matrix of normalised input vectors, `tn` is the matrix of normalised target vectors, `stdp` is the vector containing standard deviations for each `input(p)`, `meanp` is the vector containing standard deviations for each `input(p)`, `stdt` is the vector containing standard deviations for each `target(t)` and `meant` is the vector containing standard deviations for each `target(t)`. The `prestd` function takes in two arguments, `Traininput(Input matrix)` and `Trainoutput (Target matrix)` which contains the data to be pre-processed. On the left hand side of `prestd`, `pn` is a matrix which holds the pre-processed inputs, `meanp` and `stdp` matrices holds the mean and standard deviation of the inputs, `tn` is a matrix which holds the pre-processed targets and `meant` and `stdt` matrices holds the mean and the standard deviation of the target values. The following code snippet was used to create the climate variability forecasting neural network in Octave:

```
net = newff(min-max(pn), (20.1),
 tansig, purelin, trainba, mse);
```

where `net` is the feed forward neural network which has been created earlier using the `newff` function. The function `min-max` is used to return the pre-processed input `pn` with the range of the matrix row. The neurons for the neural network's hidden layer are specified and for the output layer. The transfer functions used at each layer was also specified. In the system design the chosen transfer function is `tansig` on the hidden layer and the `purelin` transfer function on the output layer. The network training function was specified and is the `trainba` function, which was developed for the bat algorithm. The performance function used was the mean square error (`mse`). The neural network was trained, verified and tested on data for the period 1985 to 2007. The aim of this training process was to determine the neural network architecture that would yield the best prediction performance. The Bat algorithm was used in training the climate variability forecasting neural network. The following code snippet was used for training the neural network:

```
net = train(net,pn,tn);
```

where `net` is the climate forecasting neural network, `train` is the function that takes normalised Input and Target matrices, `pn` is the matrix of normalised input vectors, `tn` is the matrix of normalised target vectors. The training normally stops when one of these five conditions are reached:

i. Performance goal met,

ii. The maximum number of epochs reached,

iii. Minimum gradient reached,

iv. Maximum time exceeded, or

v. Performance has been minimised to the goal.

After training the neural network, its performance is obtained by testing it. Two matrices were used; Test-Input and Test-Output, which contains 38

columns each. These matrices were used to determine the accuracy rate of the neural network. This approach was successful and an appropriate architecture was identified. The neural network reached its performance goal with 2 hidden layers and 10 hidden neurons as shown in Figure 2. In the first layer the `tansig` functions calculate the layer's output as follows:

$$n = \frac{2}{1 + e^{-2n}} - 1 \tag{6}$$

In the output layer the linear function calculates the neural network's output. Further the following three training parameters were defined:

i. `net.trainParam.show = 100;` displays the results at every 100^{th} iteration (epoch),

ii. `net.trainParam.goal = 0.001;` is the error tolerance, and

iii. `net.trainParam.epochs = 1000;` is the maximum number of iterations.

During the training phase different combinations of transfer functions were tested to come up with the best combination for the neural network. The neural network performed better with a combination of `tansig` and `purelin`.

3. RESULTS

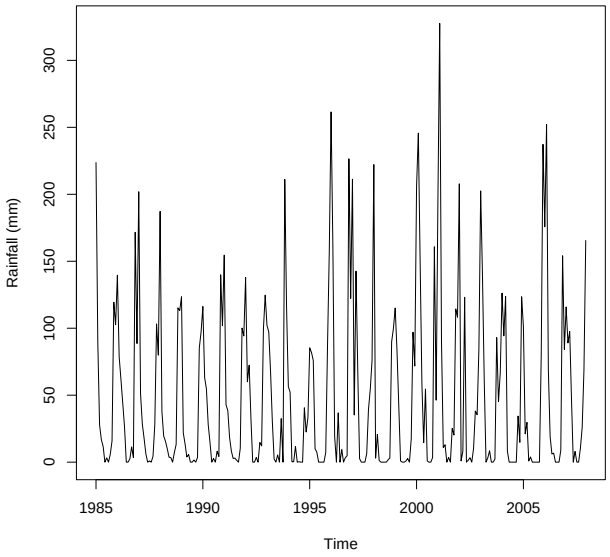

Figure 4: Structure of Training Data – Bulawayo Metropolitan Province

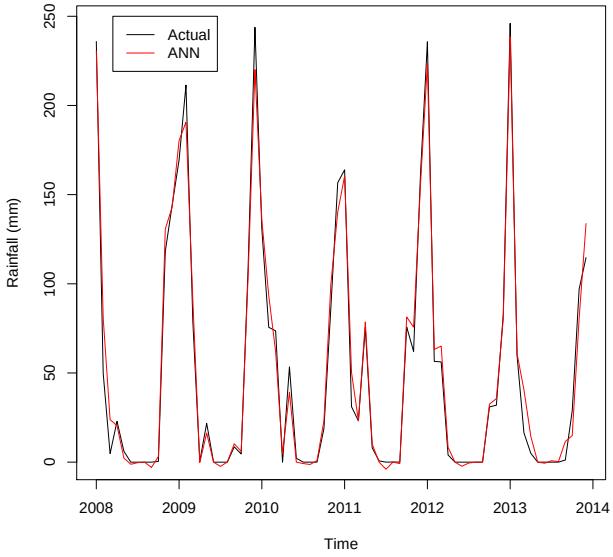

Figure 5: Comparison of the ANN Output against Target Data

PERFORMANCE testing was done using the Test-Set from 2008 to 2013. The results showed a slight marginal error from the target output. Software integration testing was done by checking that the Neural Network and the graphical user interface interacted without any errors. Simulation was done to test the performance of the neural network. The following code snippet was used for the simulation:

```
a = sim(net, Testinput);
```

where `a` is the output of the neural network which should be compared with the `Testoutput(Target)`, `net` is the neural network, `Testinput` is the Input vector for the Test-Set and `Testoutput` is the Target vector for the Test set. After simulating the neural network data was denormalised, that is to say post-processed using the function `poststd`. Figure 4 shows sample test dataset for the years 1985 – 2007, for the Bulawayo metropolitan province. The network was trained for monthly averages per province. Figure 5 shows results of testing the ANN forecasting for the years 2008 – 2013 which was compared against the real (target) data for the same period (black line shows the target data and the red line shows the ANN output). The error obtained from the ANN can be observed to be quite marginal. The mean percentage accuracy ψ was computed by:

$$\psi = \frac{Number\ of\ correctly\ predicted\ output}{Total\ number\ of\ outputs} \times 100 \tag{7}$$

and was found to be averaging at 93.2%.

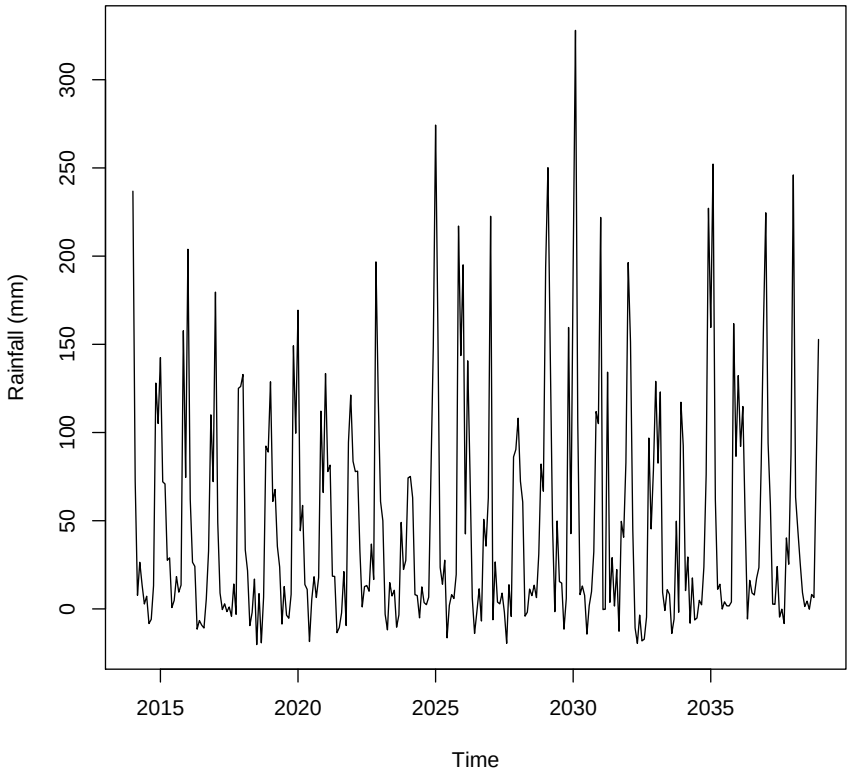

Figure 6. Prediction of a 25 year rainfall pattern

4. CONCLUSION

WHEN the ANN is run for all the data collected from all districts, grouped into provinces of Zimbabwe, for 25 years forecast, the neural network suggests (as seen in plotted sample output in Figure 6) that:

i. There is a decade of reducing average precipitation, from 2014 to 2024, followed by an increase in average precipitation, from 2025 to 2036, with exceptions in years 2028 and 2033. This seems to suggest that the rainfall pattern appears to follow a ten year cycle of low precipitation followed by a ten year cycle of improved precipitation, with few exceptions.

ii. In the years of low precipitation, the dry months appear to be receiving some minimal rainfall suggesting that they are relatively not so dry. However, in the years with improved precipitation, the dry months appear to be very dry, recording zero rainfall, and the rainy months with very high rainfall. This seems to suggest a possibility of flash flooding in the rainy months.

Summing up the 25 year period, there appears to be, on average, a 3% increase in average precipitation. This suggests that, the eastern, north-eastern and south-eastern districts of Zimbabwe are likely to be experiencing flash floods in the years after 2025, if the input parameters used are consistent with no drastic change to the weather. The objectives of the study were met, and are:

i. To model a climate forecasting neural network using data obtained from the Zimbabwe Meteorological Department.

ii. To apply the Bat Algorithm in optimising the climate forecasting neural network.

iii. To attempt to forecast climate variability.

The underlying ANN architecture that was adopted in this study is a Feed Forward ANN architecture. For further study, we recommend that the Bat algorithm be used with other neural network architectures, for example Recurrent Neural Network architecture. It is also possible to use other variants of the Bat algorithm, for example the Fuzzy Bat algorithm, so as to assess their performance compared to the standard Bat algorithm used in this study. More research needs to be done in order to improve the

ANN climate forecasting model by investigating different topologies and training algorithms. Since average monthly intervals for only four weather parameters namely temperature, pressure, rainfall and humidity were considered for rainfall prediction, seasonality was not a concern. However, it is possible to consider other weather parameters such as wind speed, wind direction, clouds, et cetera, in the prediction of rainfall and this may have a significant influence on the predictive performance.

5. NOTES

I The BAT Algorithm Implementation

Listing 2. Bat Algorithm as adapted into Octave

```
% ------------------------------------%
%            Bat Algorithm            %
%   Originally by Xin-She Yang, 2010   %
% Adapted into Octave for ANN training %
% ------------------------------------%
% Main programs starts here

function [best,fmin,N_iter]=bat_algorithm
   (para)
A=para(3);
r=para(4);
Qmin=0;
Qmax=2;
% Iteration parameters
% Initialize the population/solutions
for i=1:n,
  Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);
  Fitness(i)=Fun(Sol(i,:));
end
% Find the initial best solution
[fmin,I]=min(Fitness);
best=Sol(I,:);
% Start the iterations
for t=1:N_gen,
% Loop over all bats/solutions
for i=1:n,
 Q(i)=Qmin+(Qmin-Qmax)*rand;
 v(i,:)=v(i,:)+(Sol(i,:)-best)*Q(i);
 S(i,:)=Sol(i,:)+v(i,:);
 % Apply simple bounds/limits
 Sol(i,:)=simplebounds(Sol(i,:),Lb,Ub);
 % Pulse rate
 if rand>r
  % The factor 0.001 limits the
  % step sizes of random walks
  S(i,:)=best+0.001*randn(1,d);
 end if
 % Evaluate new solutions
 Fnew=Fun(S(i,:));
 % Update if the solution improves, or
   not too loud
 if (Fnew<=Fitness(i)) & (rand<A) ,
  Sol(i,:)=S(i,:);
  Fitness(i)=Fnew;
 end if
 % Update the current best solution
 if Fnew<=fmin,
  best=S(i,:);
  fmin=Fnew;
 end if
end for
N_iter=N_iter+n;
end %function
```

REFERENCES

ABDEL-RAHMAN E, Ahmad M, Akhtar A. R , (2012). *A metaheurisic bat-inspired algorithm for full body human pose estimation*, in: Ninth Conference on Computer and Robot Vision, pp. 369375.

ABHISHEK S, Thakur G. S. M, Gupta D, (2012), *Proposing Efficient Neural Network Training Model for Kidney Stone Diagnosis*, Department of Computer Science and Engineering Lovely Professional University, Jalandhar (Punjab), India, Vol. 3

AGUADO E, and Burt J. E, (2010) *Understanding weather and climate*, New York, Prentice Hall.Altringham J.D, (1996).

ANIL K, Mao J, and Mohiuddin K, (1996), *Artificial Neural Networks IBM Almaden Research Center*, IEEE Computer Special issue on Neural Computing March.

ANOCHI J. A, and DA Silva J. D. S, (2011), *Neural Network Models for Climate Forecasting based on Reanalysis Data*, Xcongresso Brasileio de Intelligencia Computacional.

BAEDE A. P, Ahlonsou M.E, Ding Y, and D Schimel, (2001), *The Climate System: an Overview*. Climate Change, Cambridge University Press.

BRUEN M, Krahe, P. and Zappa M, (2010). *Uncertainty in flood forecasts and end-users' perspectives*, 6th European Conference on Radar in Meteorology and Hydrology, ERAD 2010, Romania

BROWN D, Chanakira R, Chitiza K and Dhliwayo M, (2012), *Climate Change impacts vulnerability and adaptation in Zimbabwe*, December 2012.

BROWN D, Chanakira R, Chatiza K, Dhliwayo M, Dodman D, and Mugabe P, (2012), *Climate change impacts, vulnerability and adaptation in Zimbabwe*.

CHAGUTAH T, (2010). *Climate change vulnerability and preparedness in South Africa, Zimbabwe Country Report*. Hainrich Boell Stiftung, Capetown.

CHEN T. W, and Vassiliadis V. S,(2003), *Solution of general non linear optimization problems using the penalty modified barrier method with the use of exact hessians* ,Computer, Chem, Eng, vol 27, pp. 501-525.

CHEUNG V, and Cannons K, (2002), An Introduction to Neural Networks Signal and Data Compression Laboratory Electrical and Computer Engineering University of Manitoba Winnipeg, Manitoba, Canada.

CUI Z. H, and Cai X. J, (2009). *Integral particle swarm optimisation with dispersed accelerator information*, Fundam. Inform., Vol. 95, pp. 427447.

DENNIS A, Wixom B.H and R.M Roth, (2012), *Systems Analysis and Design*, 5th edition, p.54-57.

DENG W. J, Chen W. C and W Pei, (2008), *Back-propagation neural network based importance performance for determining critical service attributes*, Journal of Expert Systems and Applications, vol. (2), pp. 1-26.

DEVI C. J, Syam B, Reddy P. K, Kumar V, Reddy B. M, and Nayak N. R, (2012) *International Journal of Engineering Trends and Technology* Volume 3, Issue 1.

DE-VILLIERS M. R, (2005), *Three approaches as pillars for interpretive Information Systems research: development research, action research and grounded theory* In: Bishop.

DU Z. Y, Liu B, (2012). *Image matching using a bat algorithm with mutation*, Applied Mechanics and Materials, Vol. 203, No. 1, pp. 8893.

EATON J. W, (1997), *A high-level interactive language for numerical computations*, Edition 3 for Octave version 2.1.x, GNU Octave, Boston, USA, February 1997.

GANDOMI A. H, Yang X. S, Alavi A. H, and Talatahari S, (2013), *Bat algorithm for constrained optimization tasks*, Neural Computing and Applications, vol. 22, pp. 12391255.

GANDOMI A, Yang X. S, Alavi A, and S Talatahari, (2013) *Bat algorithm for constrained Optimization tasks*, Neural Computing and Applications, 221239-55.

GEETHA G, and Selvaraj R. S, *Prediction of monthly rainfall in Chennai using backpropagation neural network model*, Research Department of Physics, Presidency College, Chennai, Tamil Nadu, India.

GLOVER F, and Kochenberger G. A, (2003), *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston, MA.

GOYAL S, and Patterh M. S, (2013), *Performance of Bat Algorithm on Localization of wireless Sensor Network, International Journal of Computers and Technology*, May 25 2013, vol 6.

HALL T, Harold E. B, and C.A Doswell, (2008), *Weather and forecasting, Precipitation Forecasting Using a Neural Network*, NOAA/NWS West Gulf River Forecast Center, Fort Worth, Texas

HAVENS T. C, Spain C. J, Salmon N.G and J.M Keller, (2008) *Roach Infestation optimization IEEE swarm Intelligence Symposium*, IEEE Press, Piscataway, pp. 1-7.

HU M. J. C, (1964) *Application of ADALINE system to weather forecasting*, Technical Report, Stanford Electron.

JAMIL M, Zepernic H. J, and Yang X. S, (2013). *Improved bat algorithm for global optimization*, Applied Soft Computing, (2013, submitted).

KAVEH A, and P Zakion, (2013), *Enhanced Bat Algorithm for optimal Design of Skeletal Structures*, Asian Journal of Civil Engineering vol 15, 2 September 2013.

KAVEH A, Talatahari S and M.T. Alamica, (2012), *A new hybrid meta-heuristic for optimum design of frame structures*, Asian Journal of Civil Engineering.

KENNEDY J, and Eberhart R, (2001), *Swarm intelligence*, Morgan Kaufmann San Francisco.

KHAN K, and Sahai A,(2012), *A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context*, [online] Available from: http://www.mecs-press.org, Published June 2012 .

KHAN A. L, Qurashi R. J, and Khan U. A, (2011), *A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies*, International Journal of Computer Science Issues, Vol. 8, Issue 4, July 2011.

KOMARASAMY G, and Wahi A, (2013), *A New Algorithm for selection of better K value using modified hill climbing in K-means algorithm*, Journal of Theoretical and Applied Information Technology, Vol. 55, 30th September 2013.

KOURIE, D. (Eds.) Research for a Changing World: Proceedings of SAICSIT 2005. ACM International Conference Proceedings Series.

KRSE B, and Smagt P, (1996), *An Introduction to Neural Networks*, The University of Amsterdam, Amsterdam

LORENZ, E. N, (1982), *Atmospheric predictability experiments with a large numerical model*. Tellus, pp. 505-513.

MADZWAMUSE M, (2010), *Climate Governance in Africa Adaptation Strategies and Institutions*, Heinrich Boll Stiftung(HBS).

MAQSOOD P, Khan R M, and A Abraham, (2006), *Weather forecasting models using ensembles of neural networks*.

MARZBAN C, and G Stumpf, (1996), *A neural network for tornado prediction based on Doppler radar derived attributes*, Journal of applied Meteorology 35, pp. 617-626.

MCCONNELL S, (1996), *Rapid Development Taming Wild Software Schedules*, 1st Edition Microsoft Press

MICHAELIDES S. C, Neocleous C. C and C. N Schizas, (1995), *Artificial neural networks and multiple linear regression in estimating missing rainfall data*, In: Proceedings of the DSP95 International Conference on Digital Signal Processing, Limassol, Cyprus, pp. 668673

NAKAMURA R. Y. M, Pereira L. A. M, Costa K. A, Rodrigues D, Papa J. P and Yang X. S, (2012). *BBA: A binary bat algorithm for feature selection*, In: 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 22-25 Aug. 2012, IEEE Publication, pp. 291-297

OPENSHAW S, and Openshaw C, (1997) *Artificial Intelligence in Geography*, Chichester, John Wiley.

PACHECO A. B, (2012), *Introduction to GNU Octave, User Services Consultant LSU HPC and LONI*, HPC Training Series, Louisiana State University, Baton Rouge, April 18 2012

RAMAWAN M. K, Othman Z, Sulaiman S. I, Musirin I and N Othman, (2014). *A hybrid Bat Algorithm Artificial Neural Network for grid-connected photovoltaic system output prediction*, In: Proceedings of the Power Engineering and Optimization Conference (PEOCO), IEEE 8th International Conference, Langkawi, 24-25 March 2014, pp. 619-623.

RASHEDI E, Nezamabudi-Pour H and S Saryazdi, (2009), *GSA a gravitational search algorithm* Inf-Sci 179(13), pp. 2232-2248.

ROJAS R, (1996), *Neural Networks, A Systematic Introduction*, Verlag Berlin Heidelberg, Germany.

RIZWAN M, Qureshi J, and S.A Hussain, (2008), *An Adaptive Software Development Process Model, Advances in Engineering Software*, Elsevier Ltd Amsterdam, The Netherlands, pp. 654-658.

RUMELHART D. E, Hinton G.E, Williams R.J, (1986), *Learning internal representation by back- propagating errors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press.

SAKA M. P, and Dogan E, (2012), *Recent developments in meta-heuristic algorithms a review in BHV toping* (Ed) Computational Technology Reviews 5, pp. 31-78.

SOMMERVILLE I, (2004), *Software Engineering*, Addison Wesley, UK.

SUBHAJINI A. C, (2011), *An efficient weather Forecasting System using Radial Basis Function Neural Network*. Journal of Computer Science, pp. 962-966.

YANG X. S, (2008), *Nature inspired Metaheuristic Algorithms*, Luniver press, frome, uk.

YANG X. S, and Debs, (2009), *Cuckoo Search via levy flights*, In: World Congress on Nature and Biologically Inspired Computing, pp. 210-214.

YANG X. S, (2010), *A new meta-heuristic bat-inspired algorithm in Nature Inspired Cooperative Strategies for Optimization* (Eds. J. R. Gonzalez et al.), Studies in Computational Intelligence, Springer Berlin, 284, Springer, 284, pp. 65-74.

YANG X. S, (2011), *Bat Algorithm for Multi objective optimization*, International Journal of Bio-Inspired Computation, vol 3, pp. 267-274.

YONAS B, Dibike and P Coulibaly, (2006), *Temporal Neural Networks for downscaling climate variability and extremes*, pp. 136-144.

ZHANG G, Patuwo B. E, and Hu M. Y, (1997), *Forecasting with artificial neural networks*, International Journal of Forecasting 14, pp. 3562, USA