Neural Networks Training Based on Differential Evolution Algorithm Compared with Other Architectures for Weather Forecasting34

H. M. Abdul –Kader

Minufiya University, Faculty of Computers and Information, Shabin Elkom, Egypt,

Abstract
Accurate weather predictions are important for planning our day-to-day activities. In recent years, a large literature has evolved on the use of artificial neural networks (ANNs) in many forecasting applications. Neural networks are particularly appealing because of their ability to model an unspecified non-linear relationship between weather variables. This paper evaluates three neural networks architectures with different training techniques, in this context: the popular multilayer perceptron (MLP), the radial basis function network (RBF) and feed forward neural networks which were trained by differential evolution algorithm. Different testing and training scenarios are presented. Those scenarios are designed to obtain the most suitable one for weather predication at different neural network architectures. Simulation results for each scenario demonstrate the effectives of both neural network architectures and its associated training algorithm.

Key Words:
Weather Forecasting, Neural Networks, Radial basis functions, Differential Evolution

1. Introduction

Weather simply refer to the condition of the air on the earth at a given place and time. The application of science and technology to predict the state of the atmosphere for a future time and a given location is so important due to its effectiveness in human life [1], [2].

Today, weather forecasts are made by collecting quantitative data about the current state of the atmosphere and using scientific understanding of atmospheric processes to project how the atmosphere will evolve. The chaotic nature of the atmosphere implies the needs of the massive computational power required to solve the equations that describe the atmospheric conditions. This is resulted from incomplete understanding of atmospheric processes which mean that forecasts become less accurate as the difference in time between the present moment and the time for which the forecast is being made increases.

Weather is a continuous, data-intensive, multidimensional, dynamic and chaotic process, and these properties make weather forecasting a big challenge. Generally, two methods are used to forecast weather (a) the empirical approach and (b) the dynamical approach. The first approach is based upon the occurrence of analogs and is often referred to by meteorologists as analog forecasting. This approach is useful for predicting local-scale weather if recorded cases are plentiful. The second approach is based upon equations and forward simulations of the atmosphere, and is often referred to as computer modeling. The dynamical approach is only useful for modeling large-scale weather phenomena and may not predict short-term weather efficiently. Most weather prediction systems use a combination of empirical and dynamical techniques [2], [3].

ANNs provide a methodology for solving many types of nonlinear problems that are difficult to solve by traditional techniques. Most meteorological processes often exhibit temporal and spatial variability, and are suffered by issues of nonlinearity of physical processes, conflicting spatial and temporal scale and uncertainty in parameter estimates. The ANNs have capability to extract the relationship between the inputs and outputs of a process, without the physics being explicitly provided [4]. Thus, these properties of ANNs are well suited to the problem of weather forecasting under consideration [3].

The main purpose is to develop the most suitable ANNs architecture and its associated training technique for weather prediction. The data which are used in training and testing are typical previous measurement of ambient temperature for selected Egyptian cities. This development will be based on using three different neural networks architecture to demonstrate the suitable one for this application. Backpropagation (BP) feedforward network , radial basis function network and feedforward neural network which were trained by differential evolution algorithm are the selected architectures in this paper

The rest of this paper is organized as follows. Section 2 reviews the basic structure of weather forecasting system. The basic architecture of the both radial basis functions (RBF) neural network, multilayer feedforward neural networks and differential evolution algorithm are given in section 3. Those neural network architectures are used as a prediction tools for the weather. In section 4 simulation results for both neural network architectures are introduced. Discussions of the obtained results are given in section 5. Finally the conclusions are drawn in section 6.

Manuscript received March 5, 2009
Manuscript revised March 20, 2009

2. Weather Forecasting System Structure

Components of a modern weather forecasting system include the following modules: data collection, data assimilation and numerical weather prediction. A brief review for each component is presented in this section.

2.1 Data collection

Observations of atmospheric pressure, temperature, wind speed, wind direction, humidity, and precipitation are made near the earth's surface by trained observers, automatic weather stations. The World Meteorological Organization acts to standardize the instrumentation, observing practices and timing of these observations worldwide [1].

2.2 Data Assimilation

During the data assimilation process, information gained from the observations is used in conjunction with a numerical model's most recent forecast for the time that observations were made to produce the meteorological analysis. This is the best estimate of the current state of the atmosphere. It is a three dimensional representation of the distribution of temperature, moisture and wind [1]

2.3 Numerical Weather Prediction

Numerical Weather Prediction (NWP) uses the power of computers to make a forecast. Complex computer programs, also known as forecast models, run on supercomputers and provide predictions on many atmospheric variables such as temperature, pressure, wind, and rainfall. A forecaster examines how the features predicted by the computer will interact to produce the day's weather [1].

3. Weather Prediction Using ANN

ANN is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) to solve specific problems. ANNs, learn by example. An ANN is configured for a specific application, such as pattern recognition, data classification or prediction, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include: Adaptive learning, fault Tolerance. Neural networks have broad applicability to real world problems. In fact, they have already been successfully applied in many industries [4].

The configuration of the neural network depends highly on the problem. Therefore, it is left with the designer to choose an appropriate number of hidden layers and hidden layer nodes based on experience. Thus, an appropriate architecture is determined for each application using the trial and error method. The learning rate parameter and momentum term were adjusted intermittently to speed up the convergence [4].

3.1 Feedforward Backpropagation ANN

A typical neural network consists of layers. In a single layered network there is an input layer of source nodes and an output layer of neurons. A multi-layer network has in addition one or more hidden layers of hidden neurons. Both types of networks are displayed in Fig. 1. Extra hidden neurons raise the network's ability to extract higher-order statistics from (input) data. This is a crucial quality, especially if there is a large input layer. Furthermore a network is said to be fully connected if every node in each layer of the network is connected to every other node in the adjacent forward layer [4].

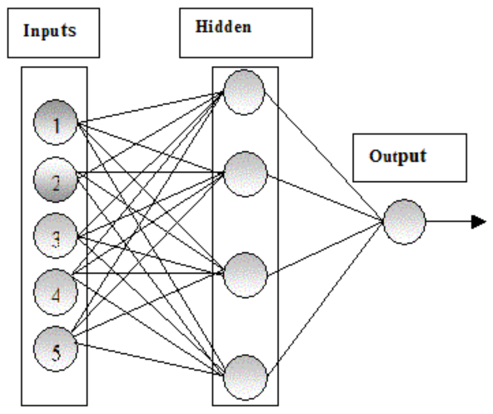

Fig. 1 Simple Neural Network Architecture

The network "learns" by adjusting the interconnections (called weights) between layers. When the network is adequately trained, it is able to generalize relevant output for a set of input data. A valuable property of neural networks is that of generalization, whereby a trained neural network is able to provide a correct matching in the form of output data for a set of previously unseen input data. Learning typically occurs by example through training, where the training algorithm iteratively adjusts the connection weights (synapses). Backpropagation is one of the most famous training algorithms for multilayer perceptrons. BP is a gradient descent technique to minimize the error E for a particular training pattern. For adjusting the weight w_{ij} from the i-th input unit to the j-th output, in the batched mode variant the descent is based on the gradient $\nabla E(\frac{\delta E}{\delta w_{ij}})$ for the total training set:

$$\Delta w_{ij} = -\varepsilon^* \frac{\delta E}{\delta w_{ij}} + \alpha^* \Delta w_{ij}(n-1) \quad (1)$$

The gradient gives the direction of error E. The parameters ε and α are the learning rate and momentum respectively [3].Normally, the learning rate is held constant throughout the training. If the learning rate is too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm is slow to converge. The performance of the steepest descent algorithm can be improved by using an adaptive learning rate, which will keep the learning step size as large as possible while keeping learning stable. The learning rate is made adaptive to the complexity of the local error surface. If the new error exceeds the old error by more than a predefined ratio, the new weights are discarded. In addition, the learning rate is decreased. Otherwise the new weights are kept. If the new error is less than the old error, the learning rate is increased. This architecture was used in the field of forecasting in many fields [5], [6].

3.2 Basic Architecture of RBF

A RBF network with m outputs and n_h hidden nodes can be expressed as:

$$y_i(t) = w_{i0} + \sum_{j=1}^{n_h} w_{ij}\phi\left(\left\| v(t) - c_j(t) \right\|\right); \quad i = 1,...,m \quad (2)$$

where w_{ij}, w_{i0} and $c_j(t)$ are the connection weights, bias connection weights and RBF centres respectively, $v(t)$ is the input vector to the RBF network composed of lagged input, lagged output and lagged prediction error and $\phi(\bullet)$ is a non-linear basis function. The symbol $\|\bullet\|$ denotes a distance measure that is normally taken to be the Euclidean norm.

Since neural networks are highly non-linear, even a linear system has to be approximated using the non-linear neural network model. However, modelling a linear system using a non-linear model can never be better than using a linear model. Considering this argument, the RBF network with additional linear input connections is used. The proposed network allows the network inputs to be connected directly to the output node via weighted connections to form a linear model in parallel with the non-linear standard RBF model as shown in Figure 2.

The new RBF network with m outputs, n inputs, *n-h* hidden nodes and n_l linear input connections can be expressed as:

$$y_i(t) = w_{i0} + \sum_{j=1}^{n_l} \lambda_{ij} vl(t) + \sum_{j=1}^{n_h} w_{ij}\phi\left(\left\| v(t) - c_j(t) \right\|\right); \ i = 1,2,\cdots,m \quad (3)$$

where the λ‘s and vl’s are the weights and the input vector for the linear connections respectively. The input vector for the linear connections may consist of past inputs, outputs and noise lags. Since λ's appear to be linear within the network, the λ's can be estimated using the same algorithm as for the w’s. As the additional linear connections only introduce a linear model, no significant computational load is added to the standard RBF network training. Furthermore, the numbers of required linear connections are normally much smaller than the number of hidden nodes in the RBF network. In the present study, Givens least squares algorithm with additional linear input connection features is used to estimate w’s and λ‘s [7-10].

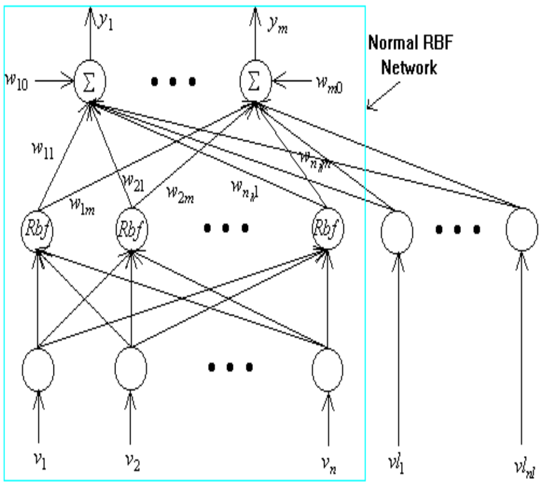

Fig. 2 RBF Network Architecture

The radial basis function (RBF) network is one approach which has shown a great promise in function approximation and prediction of problems because of its faster learning capacity. Compared to the feedforward network, the radial basis function (RBF) network is the next-most-used network model. As the name implies, this network makes use of radial functions.

3.3 Differential Evolution Algorithms

Simulated Annealing (SA), Genetic Algorithms (GA), and Differential Evolution (DE). In the recent past, these algorithms have been successfully applied for solving complex engineering optimization problems and are widely known to diffuse very close to the global optimum solution. A genetic algorithm that is well adapted to solving a combinatorial task like the traveling salesman problem may fail miserably when used to minimize functions with real variables and many local optima. The concept of binary coding used by simple GA limits the resolution with which an optimum can be located to the precision set by the number of bits in the integer. So, the simple GA was later modified to work with the real variables as well. Just as floating point numbers are more appropriate than integers for representing points in continuous space, addition is more appropriate than random bit flipping for searching the continuum [11]. Consider, for example to change a binary 16(10000) into a binary 15(01111) with bit-wise flipping would require inverting all the five bits. In most bit flipping schemes a mutation of this magnitude would be rare. Alternately under addition, 16 become 15 simply by adding -1. Adopting addition as the mutation operator to restore the adjacency of nearby points is not, however, a panacea. Then the fundamental question concerning addition would be how much to add. The simple adaptive scheme used by DE ensures that these mutation increments are automatically scaled to the correct magnitude. Similarly DE uses a non-uniform crossover in that the parameter values of the child vector are inherited in unequal proportions from the parent vectors. For reproduction, DE uses a tournament selection where the child vector competes against one of its parents. The overall structure of the DE algorithm resembles that of most other population based searches. The parallel version of DE maintains two arrays, each of which holds a population of NP, D-dimensional, real valued vectors. The primary array holds the current vector population, while the secondary array accumulates vectors that are selected for the next generation. In each generation, NP competitions are held to determine the composition of the next generation. Every pair of vectors (Xa, Xb) defines a vector differential: Xa - Xb. When Xa and Xb are chosen randomly, their weighted differential is used to perturb another randomly chosen vector Xc. This process can be mathematically written as X.c = Xc + F (Xa - Xb). The scaling factor F is a user supplied constant in the range ($0 < F \leq 1.2$). The optimal value of F for most of the functions lies in the range of 0.4 to 1.0 [11,[12]. Then in every generation, each primary array vector, Xi is targeted for crossover with a vector like X' c to produce a trial vector Xt. Thus the trial vector is the child of two parents, a noisy random vector and the target vector against which it must complete. The non-uniform crossover is used with a crossover constant CR, in the range $0 \leq CR \leq 1$. CR actually represents the probability that the child vector inherits the parameter values from the noisy random vector. The architecture DE algorithm is shown in fig.3.

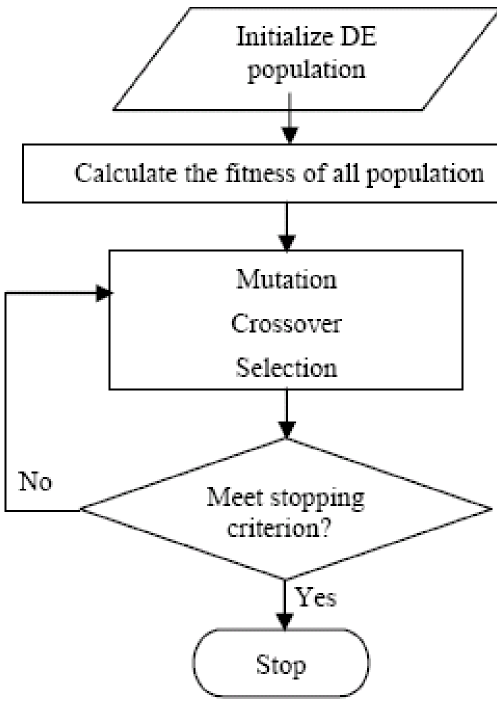

Figure 3 the architecture DE algorithm

In this paper the steps of applying DE in neural network training which was described in [11] will be used in our simulation results.

4- Neural Network Weather Forecasting Implementation

In this section an implantation for weather forecasting is introduced using three different neural network architectures, namely, RBF and backpropagation neural networks. For this purpose different case studies are implemented for both neural networks architectures

4.1 Radial Bases Function Neural Network

The simulation procedure using RBF network is divided into two steps, the first one first consider the selection of RBF network setting parameters. The second one considers the using of this network in forecasting at different training and testing scenarios. The details of those two steps are given in the following subsections.

- RBF Networks Setting

In this simulation, the implemented radial basis function network has a single hidden layer with 10 nodes. Each node in the hidden layer use Gaussian function as basis functions [8]. The basis centers are selected from the input data which are used to train the network. The weighting vectors are initialized with random values. The learning of the network is based on the backpropagation algorithm [5]. The radial basis network using is built Matlab neural network toolbox, so the following architecture parameters must be specified.

- Auto center at output layer: this parameter controls whether the first layer weights are learned or set. It must be an integer value A value of 1 indicates the center weights are determined using self-organizing learning. A value of 0 indicates the center weights must be explicitly set.
- Number of inputs: this parameter sets the number of units allocated for the input layer. It must be an integer value greater than or equal to 1.
- Number of basis units in the hidden layer: this parameter sets the number of basis units allocated for the hidden layer. It must be an integer value greater than or equal to zero.
- Number of outputs: this parameter sets the number of units allocated for the output layer. It must be an integer value greater than or equal to 1.

There are many disadvantages of RBF networks setting. First it is difficult to use Gaussian functions to approximate constant values if a function has nearly constant values in some intervals. Second, when the training patterns include a large error, the network will interpolate these training patterns incorrectly.

- RBF network Forecasting Steps

Daily data sets of 2006 (i.e. maximum, minimum and average temperature) of Cairo city were used as training set, and first two months of 2007 were used as test set of our neural network.

- The first case study scenario is described as follows:

We use the average temperature of Cairo city using only one year as at a training set namely 2006 temperatures, in addition to the previous two days of the day we want to predict. Then we use the obtained network architecture and weights to test the ability of this network to predict the average temperatures for the first two months in the next year (2007). The result of this scenario is shown in Fig. 4. This figure shows that the obtained result is inaccurate enough.

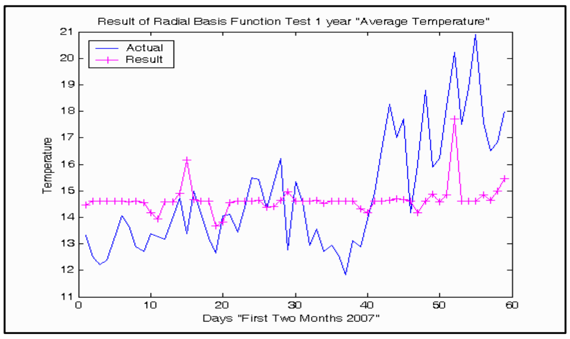

Fig.4 RBF Test output using training set 2006

- The second case study scenario is described as follows:

In this case, we tried to increase the training data set via using the temperature records of three previous years instead of one year , so the records of use 2004, 2005 , 2006 yeas are used as a training set. Then the obtained network architecture and weights to test the ability of this network are used to predict the average temperatures for the first two months in the next year (2007). The result of this scenario is shown in Fig. 5. This figure shows that the obtained result is little bit more accurate than scenario one.

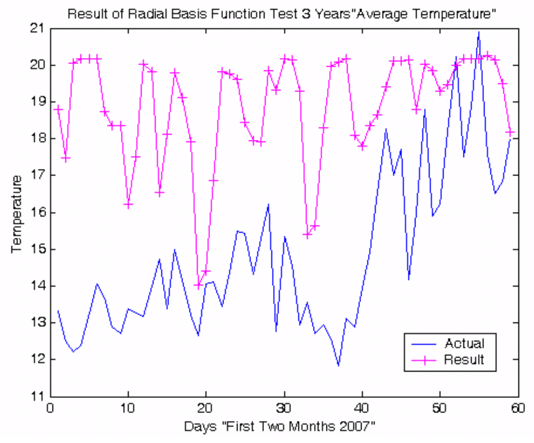

Fig.5 RBF Test output using training set 2006, 2005, and 2004.

- The third case study scenario is described as follows:

In this case, another arrangement of training data in addition changes some parameters of the network such as number of maximum trained epochs and spread constant in order. The trained data set in this case uses the previous 5 days of the same year. The result of this scenario is shown in Fig. 6. This figure shows that the obtained result is little bit more accurate than scenario two.

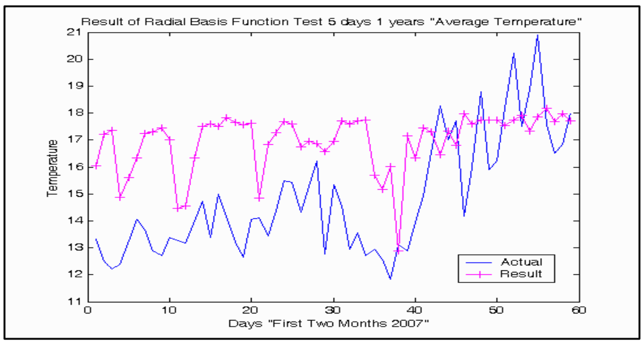

Fig.6.a One year training set

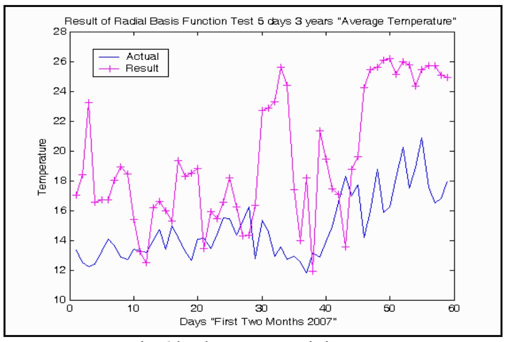

Fig.6.b Three year training set
Fig.6 RBF NN using 5 days Training set.

- The fourth case study scenario is described as follows:
In all the previous case, the average temperature is used for training and testing, while in this case we built two similar networks one for maximum temperature and the other for minimum temperature. The same network architecture and parameters of case studies on and two are used for this case. The trained data set for this case is the previous year daily minimum and maximum temperature. The obtained results for training and test outputs are illustrated in Fig. 7.

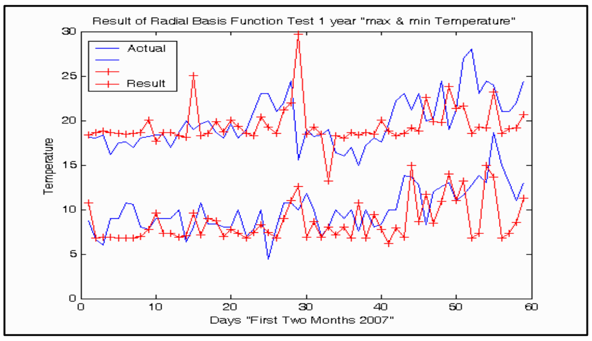

Fig. 7 RBF Training output of maximum and minimum temperature

The obtained network architecture from the above testing scenario is used to predict the next two months temperatures (minimum temperature and maximum temperature). The results of this prediction are shown in Fig.8.

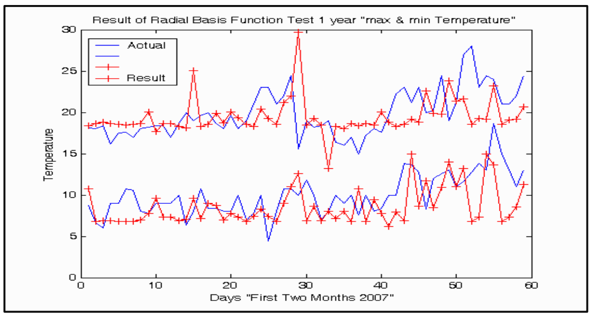

Fig.8 RBF Test output of maximum and minimum temperature

We can conclude that the RBF networks gives good training results, since it has natural unsupervised learning characteristics and a modular network structure, these properties make it more effective for fast and robust weather forecasting. The output of the neural network highly depends on the available trained data, so the manner of selection of trained data set to obtain more accurate results for forecasting should be considers.

4.2 Backpropagation Neural Networks

The backpropagation method is a popular technique to train multi-layer feed-forward neural networks in a supervised manner. In this section a feedforward neural network is used. In this case we built two similar networks one for maximum temperature and the other for minimum temperature. The trained data set for this case is the previous year daily minimum and maximum temperature of year 2006 but we use in this case 24 reading each day, one reading for each hour. The obtain results for training and test outputs are as illustrated in Fig. 9.

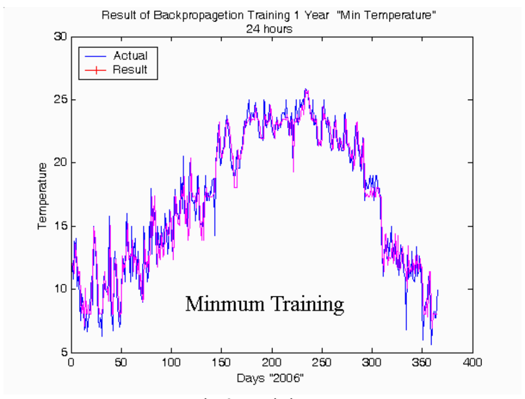

Fig.9 a Minimum

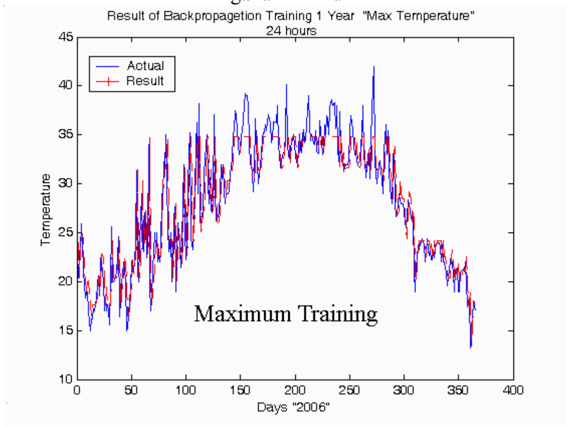

Fig.9 b Maximum
Fig.9 Backpropagation NN max and min Training results

The obtained network architecture from the above testing scenario is used to predict the next two months temperatures of year 2007 (minimum and maximum temperature). The results of this prediction are shown in Fig.10.

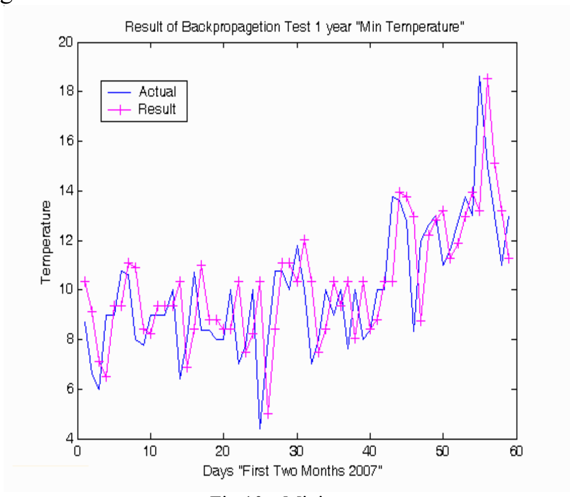

Fig 10.a Minimum

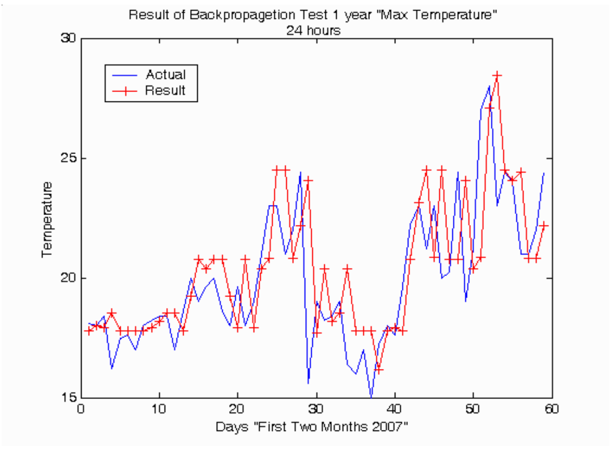

Fig 10.b Maximum
Fig.10 Backpropagation NN max and min Test results

4.3 Differential Evolution Neural Network

The proposed methodology is to train multilayer feedforward neural network which proposed in [11] is used in this paper. In this section a feedforward neural network which trained differential evolution algorithm is used. In this case we built two similar networks one for maximum temperature and the other for minimum temperature. The trained data set for this case is the previous year daily minimum and maximum temperature of years 2006 and 2007 but we use in this case 24 reading each day, one reading for each hour. The obtain results testing the trained DE network are shown in Fig. 11.

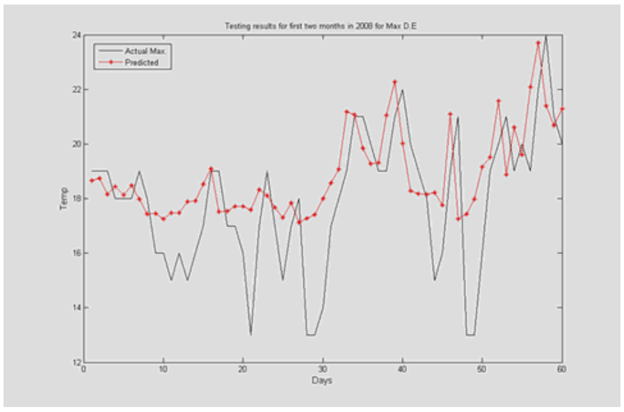

Fig.11 a Maximum

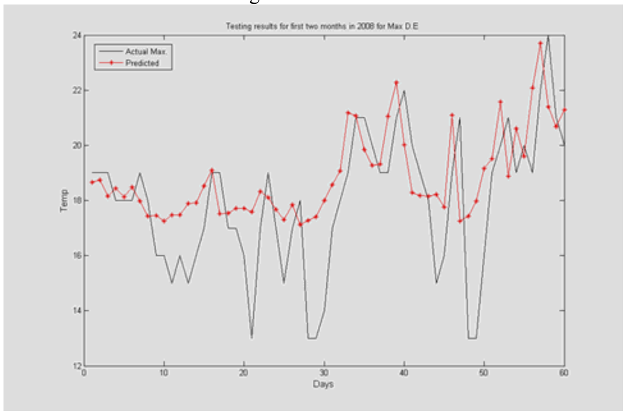

Fig.11 b Minimum
Fig. 11 feedforward neural network trained by DE

5- Results Discussions

In this section the performance of the selected neural network architectures is evaluated. The obtained results from simulations results can be summarized in Table. 1. and Table 2. The first table summarizes the simulated scenarios and the data used for training and testing. In the same table the training algorithm for each neural network architecture is shown. The second table shows the neural network performance for each scenario. From all scenarios and simulation results, we can conclude that scenario 6 that uses feedforward network trained with DE are the most suitable architecture for such application. This is results from the ability of DE to reach a global minimum for the backprogration trained network. Also, this scenario indicates means that for temperate prediction it is better to use two separate neural networks one for maximum and one for minimum temperate with feedforward NN trained by DE.

Table 1. Simulation results summary

Scenario	Training data year	Testing data year	Temperature used	ANN type	Training Algorithm
Scenario 1	2006	2007	Average	RBF	BP
Scenario 2	2004,2005 2006	2007	Average	RBF	BP
Scenario 3	Previous 5day of 2007	2007	Average	RBF	BP
Scenario 4	2006	2007	Maximum , Minimum	RBF	BP
Scenario 5	2006	Two months of 2007	Maximum , Minimum	MLP	BP
Scenario 6	2006 and 2007	Two months of 2008	Maximum , Minimum	MLP	DE training

Table 2. Simulation results summary

Scenario	Mean Square error	Training time in sec.
Scenario 1	10.0716	14
Scenario 2	6.08	13
Scenario 3	7.32	14
Scenario 4	5.456	16
Scenario 5	5.328 5.345	23
Scenario 6	4.3443 3.6884	10

6- Conclusions

Weather data all over the world are true time series data. This makes neural networks a very suitable tool to be used for weather forecasting. In fact a software application has been developed in this work to demonstrate the capabilities of accurate forecasting for neural network. This paper has evaluated the use of three different artificial neural network models to forecast temperature in some Egyptian towns. The gained simulated results shows that the popular feedforward neural network which trained by DE is most accurate model to use as a temperature predicator. Especially in the uniform temperature distribution (minimum or maximum temperature) which can be considered the most suitable technique for temperature forecasting as shown in scenario 6.

References

[1] Kuligowski RJ, Barros AP (1998) Localized precipitation forecasts from a numerical weather prediction model using artificial neural networks. Weather and Forecasting 13:1194–1205

[2] Maqsood I, Khan MR, Abraham A (2002a) Intelligent weather monitoring systems using connectionist models. International Journal of Neural, Parallel and Scientific Computations 10:157–178

[3] Maqsood I, Khan MR, Abraham A (2002b) Neuro-computing based Canadian weather analysis. In: The 2nd International Workshop on Intelligent Systems Design and Applications. Comput Intell Applic, Dynamic Publishers, Atanta, Georgia. pp 39–44

[4] Haykin S. (2004). Neural Networks, A Comprehensive Foundation. International Edition, Second Edition. Prentice Hall International, Inc, USA. 842 s.

[5] Innamaa S. (2000). Short-Term Prediction of Traffic situation using MLP neural networks (http://www.vtt.fi/rte/projects/tetra/ITS2000.pdf))

[6] Moro QI, Alonso L, Vivaracho CE (1994) Application of neural networks to weather forecasting with local data. In: 12 IASTED Internat Conf Applied Informat 12, 68–70

[7] Park, J. and Sandberg, I. (1991) Universal approximation using radial basis function networks. *Neural Computation*, 3(2), 246-257.

[8] Powell, M. J. D. (1987) Radial basis functions for multivariable interpolation: a review. In J. C. Mason, and M. G. Cox (Eds.), *Algorithms for Approximation*, pp. 143- 167. Oxford: Clarendon Press.

[9] Tan Y, Wang J, Zurada JM (2001) Nonlinear blind source separation using a radial basis function network. IEEE Transactions on Neural Networks 12:124–134

[10] Lundstedt, H.: Progress in space weather predictions and applications, Adv. Space Res., **36**, 2516-2523 (2005).

[11]Jarmo Ilonen,Jon I-Kristian Kamarainen and Jouni Lampinen: *Differential Evolution Training Algorithm for Feed-Forward Neural Networks,* Neural Processing Letters 17: 93–105,2003.

[12] Karin Zielinski, Dagmar Peters, and Rainer Laur , " Run Time Analysis Regarding Stopping Criteria For Differential Evolution and Particle Swarm Optimization " , 1st International Conference on Experiments/Process/System Modelling/ Simulation/ Optimization , Athens, 6-9 July, 2005 .

H. M. Abdul-kader obtained his B.S. and M.SC. (by research) both in Electrical Engineering from the Alexandria University , Faculty of Engineering , Egypt in 1990 and 1995 respectively. He obtained his Ph.D. degree in Electrical Engineering also from Alexandria University, Faculty of Engineering, and Egypt in 2001 specializing in neural networks and applications. He is currently a Lecturer in Information systems department, Faculty of Computers and Information, Menoufya University, Egypt since 2004. He has worked on a number of research topics and consulted for a number of organizations. He has contributed more than 30+ technical papers in the areas of Neural networks, Database applications, Information security and Internet applications.