Soft Computing Models for Weather Forecasting

Ajith Abraham, Ninan Sajeeth Philip[*] and P.K. Mahanti[+]

Department of Computer Science, Oklahoma State University, USA, Email: ajith.abraham@ieee.org

[*]Department of Physics, Cochin University of Science and Technology, India, Email: nsp@cusat.ac.in

[+]University of New Brunswick, New Brunswick, Canada

Keywords

Soft computing, neural network, neuro-fuzzy, weather forecast

Abstract

Long-term rainfall prediction is very important to countries thriving on agro-based economy. In general, climate and rainfall are highly non-linear phenomena in nature giving rise to what is known as "butterfly effect". The parameters that are required to predict the rainfall are enormously complex and subtle so that uncertainty in a prediction using all these parameters is enormous even for a short period. Soft computing is an innovative approach to construct computationally intelligent systems that are supposed to possess humanlike expertise within a specific domain, adapt themselves and learn to do better in changing environments, and explain how they make decisions. Unlike conventional artificial intelligence techniques the guiding principle of soft computing is to exploit tolerance for imprecision, uncertainty, robustness, partial truth to achieve tractability, and better rapport with reality. In this paper, we analysed 87 years of rainfall data in Kerala state, the southern part of Indian Peninsula situated at latitude-longitude pairs ($8^o29'$ N - $76^o57'$ E). We attempted to train 5 soft computing based prediction models with 40 years of rainfall data. For performance evaluation, network predicted outputs were compared with the actual rainfall data. Simulation results reveal that soft computing techniques are promising and efficient.

1. Introduction

Rain is one of the nature's greatest gifts and in developing countries like India; the entire agriculture depends upon rain. It is thus a major concern to identify any trends for rainfall to deviate from its periodicity, which would disrupt the economy of the country. This fear has been aggravated due to threat by the global warming and green house effect. The geographical configuration of India with the three oceans, namely Indian Ocean, Bay of Bengal and the Arabian Sea bordering the peninsula gives her a climate system with two monsoon seasons and two cyclones interspersed with hot and cold weather seasons. The parameters that are required to predict the rainfall are enormously complex and subtle so that the uncertainty in a prediction using all these parameters even for a short period. The period over which a prediction may be made is generally termed the event horizon and in best results, this is not more than a week's time. Thus it is generally said that the fluttering wings of a butterfly at one corner of the globe may cause it to produce a tornado at another place geographically far away. Edward Lorenz (meteorologist at MIT) discovered this phenomenon in 1961 and is popularly known as the butterfly effect [15]. In our research, we aim to find out how well the proposed soft computing models are able to understand the periodicity in these patterns so that long-term predictions can be made. This would help one to anticipate with some degree of confidence the general pattern of rainfall to be expected in the coming years. In pace with the global interest in climatology, there has been a rapid updating of resources in India also to access and process climatological database. There are various data acquisition centres in the country that record daily rainfall along with other measures such as sea surface pressure, temperature etc. that are of interest to climatological processing. These centres are also associated to the World Meteorological Organization (WMO).

We used an artificial neural network using backpropagation (variable learning rate), adaptive basis function neural network [12], neural network using scaled conjugate gradient algorithm and an Evolving Fuzzy Neural Network (EFuNN) [8] for predicting the rainfall time series [1] [11]. The soft computing models described above were trained on the rainfall data corresponding to a certain period in the past and cross validate the prediction made by the network over some other period. In section 2 and 3, we present some theoretical background about neural networks and neuro-fuzzy systems. In section 4, the experimental set up is explained followed by the discussions and simulation results. Conclusions are also provided towards the end.

2. Artificial Neural Networks

Artificial neural networks (ANNs) were designed to mimic the characteristics of the biological neurons in the human brain and nervous system [14]. 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 generalisation, 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 (BP) 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 ∇E ($\frac{\delta E}{\delta w_{ij}}$) for the total training set:

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

The gradient gives the direction of error E. The parameters ε and α are the learning rate and momentum respectively [3].

With standard steepest descent, 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 will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface [7]. 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 (typically 1.04), the new weights are discarded. In addition, the learning rate is decreased (typically by 70%). Otherwise the new weights are kept. If the new error is less than the old error, the learning rate is increased (typically by 5%). Thus a near optimal learning rate is obtained for the local terrain. When a larger learning rate could result in stable learning, the learning rate is also increased. When the learning rate is too high to guarantee a decrease in error, it gets decreased until stable learning resumes.

Adaptive Basis Function Neural Network (ABFNN) performs better than the standard BP networks in complex problems [12]. The ABFNN works on the principle that the neural network always attempt to map the target space in terms of its basis functions or node functions. In standard BP networks, this function is a fixed sigmoid function that can map between zero and plus one (or between minus one and plus one) the input applied to it from minus infinity to plus infinity. It has many attractive properties that made the BP an efficient tool in a wide verity of applications. However some studies conducted on the BP algorithm have shown that in spite of its wide spread acceptance, they systematically outperform other classification procedures only when the targeted space has a sigmoidal shape. This implies that one should choose a basis function such that the network may represent the target space as a nested sum of products of the input parameters in terms of the basis function. The ABFNN thus starts with the standard sigmoid basis function and alters its non-linearity by an algorithm similar to the weight update algorithm used in BP. Instead of the standard sigmoid function, ABFNN uses a variable sigmoid function defined as:

$$O_f = \frac{a + tanh(x)}{1 + a} \tag{2}$$

where a is the control parameter that is initially set to unity and is modified along with the connection weights along the negative gradient of the error function. Such a modification could improve the speed of convergence and accuracy with which the network could approximate the target space.

In the Conjugate Gradient Algorithm (CGA) a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. A search is made along the conjugate gradient direction to determine the step size, which will minimize the performance function along that line. A line search is performed to determine the optimal distance to move along the current search direction. Then the next search direction is determined so that it is conjugate to previous search direction. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction. An important feature of the CGA is that the minimization performed in one step is not partially undone by the next, as it is the case with gradient descent methods. An important drawback of CGA is the requirement of a line search, which is computationally expensive. The Scaled Conjugate Gradient Algorithm (SCGA) [10] is basically designed to avoid the time-consuming line search at each iteration. SCGA combine the model-trust region approach, which is used in the Levenberg-Marquardt algorithm with the CGA.

3. Neuro-Fuzzy Systems

A neuro-fuzzy system [2] is a combination of ANN and Fuzzy Inference System (FIS) [5] [13] in such a way that neural network learning algorithms are used to determine the parameters of FIS. An even more important aspect is that the system should always be interpretable in terms of fuzzy *if-then* rules, because it is based on the fuzzy system reflecting vague knowledge. We used evolving fuzzy neural network [8] implementing a Mamdani type FIS [9] as illustrated in Figure 1. EFuNN have a five-layer structure as shown in Figure 2. The input layer represents input variables. The second layer of nodes represents fuzzy quantification of each input variable space. Each input variable is represented here by a group of spatially arranged neurons to represent a fuzzy quantization of this variable. Different membership functions can be attached to these neurons (triangular, Gaussian, etc.). The nodes representing membership functions can be modified during learning. New neurons can evolve in this layer if, for a given input vector, the corresponding variable value does not belong to any of the existing MF to a degree greater than a membership threshold. The third layer contains rule nodes that evolve through hybrid supervised/unsupervised learning. The rule nodes represent prototypes of input-output data associations, graphically represented as an association of hyper-spheres from the fuzzy input and fuzzy output spaces. Each rule node *r* is defined by two vectors of connection weights – $W_1(r)$ and $W_2(r)$, the latter being adjusted through supervised learning based on the output error, and the former being adjusted through unsupervised learning based on similarity measure within a local area of the input problem space. The fourth layer of neurons represents fuzzy quantification for the output variables. The fifth layer represents the real values for the output variables. EFuNN evolving algorithm used in our experimentation was adapted from [8].

4. Experimentation Setup for Training and Performance Evaluation

Although the rainfall data from 1842 AD were recorded, there were many missing values in between and hence we had to restrict to periods for which a continuous time series was available. This was obtained for the period from 1893 to 1980. The rainfall data was standardised and we divided the data from 1893-1933 as training set and data from 1934-1980 as test set. While the proposed neuro-fuzzy system is capable of adapting the architecture according to the problem we had to perform some initial experiments to decide the architecture of the neural network. Since rainfall has a yearly periodicity, we started with a network having 12 input nodes. Further experimentation showed that it was not necessary to include information corresponding to the whole year, but 3-month information centred over the predicted month of the fifth year in each of the 4 previous years would give good generalization properties.

Thus, based on the information from the four previous years, the network would predict the amount of rain to be expected in each month of the fifth year. We used the same architecture for all the three neural network learning algorithms. To have a performance comparison of the different learning techniques, the training was terminated after 1000 epochs. Experiments were carried out on a Pentium II 450MHz machine and the codes were executed using MATLAB and C++. The training was repeated three times after re-initialising the networks. Test data was presented to the network and the output from the network was compared with the actual data in the time series. The worst observed errors are reported. Following are the details of network training:

EFuNN Training

We used 5 membership functions for each input variable and the following evolving parameters: sensitivity threshold *Sthr*=0.999, error threshold *Errthr*=0.001. EFuNN uses a one pass training approach. Figure 3 illustrates the EFuNN training details and we obtained an RMSE of 0.0006.

ANN Training

For neural networks using BP, backpropagation with variable learning rate (BP-VLR) and SCGA, we used 1 input layer, 2 hidden layers and an output layer [12-12-12-1]. Input layer consists of 12 neurons corresponding to the input variables. The first and second hidden layer consists of 12 neurons. For the ABFNN network, we used only I hidden layer with 7 neurons. Training errors (RMSE) achieved are reported in Table 1. To have a performance evaluation between the 4 learning algorithms, we also trained a neural network (12-7-1) with one hidden layer containing 7 neurons and the training was terminated after 1000 epochs for all the four learning methods. Figure 4 shows the training performance and convergence of the four neural network algorithms.

Test Results

Table 1 summarizes the comparative performance of EFuNN and ANN learning algorithms. Figure 5 depicts the test results given by EFuNN algorithm. Lowest RMSE was obtained using EFuNN (0.090) and it was 0.095, 0.094, 0.092 and 0.093 for BP, BP-VLR and SCG and ABF neural networks respectively. Figure 6 depicts the comparative performance between the different soft computing models.

EFuNN outperformed neurocomputing techniques with the lowest RMSE test error and performance time. EFuNN adopts a one-pass (one epoch) training technique, which is highly suitable for online learning. Hence online training can incorporate further knowledge very easily. Compared to pure BP and BP-VLR, ABFNN and SCGA converged much faster. Alternatively, BP training needs more epochs (longer training time), to achieve better performance. Compared to ANN, an important advantage of neuro-fuzzy model is its reasoning ability (*if-then* rules) of any particular state. ABFNN has also given promising results with the smallest network architecture among neural networks.

5. Conclusions

In this paper, we attempted to forecast the rainfall (one month ahead) based on soft computing techniques. As the RMSE values on test data are comparatively less, the prediction models are reliable. As evident from Figure 5, there have been few deviations of the predicted rainfall value from the actual. In some cases it is due to delay in the actual commencement of monsoon, EI-Nino Southern Oscillations (ENSO) resulting from the pressure oscillations between the tropical Indian Ocean and the tropical Pacific Ocean and their quasi periodic oscillations [4]. As climate and rainfall prediction involves tremendous amount of imprecision and uncertainty, neuro-fuzzy technique might warrant the ideal prediction model. The proposed prediction models based on soft computing on the other hand are easy to implement and produces desirable mapping function by training on the given data set. A network requires information only on the input variables for generating forecasts. In our experiments, we used only 40 years training data to evaluate the learning capability. Network performance could have been further improved by providing more training data. Moreover, the considered connectionist models are very robust, capable of handling the noisy and approximate data that are typical in weather data, and therefore should be more reliable in worst situations. Choosing suitable parameters for the soft computing models is more or less a trial and error approach. Optimal results will depend on the selection of parameters. Selection of optimal parameters may be formulated as a evolutionary search [6] to make SC models fully adaptable and optimal according to the requirement.

References

[1] Abraham A, Philip N S and Joseph K B (2001), "Will We Have a Wet Summer? Soft Computing Models for Long Term Rainfall Forecasting", 15th European Simulation Multiconference (ESM 2001), Modelling and Simulation 2001, Kerckhoffs E J H and Snorek M (Eds), Prague, Czech Republic, pp. 1044-1048.

[2] Abraham A and Nath B. (2000), "Designing Optimal Neuro-Fuzzy Systems for Intelligent Control", The Sixth International Conference on Control, Automation, Robotics and Vision, ICARCV 2000, Wang J L (Editor), (CD ROM Proceeding, Paper reference 429 - FP7.3 (I)

[3] Abraham A and Nath B. (2000), "Optimal Design of Neural Nets Using Hybrid Algorithms", In proceedings of 6th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000), Springer Verlag, Germany, Lecture Notes in Artificial Intelligence (LNAI 1886), pp. 510-520.

[4] Chowdhury A and Mhasawade S V (1991), "Variations in Meteorological Floods during Summer Monsoon Over India", Mausam, 42, 2, pp. 167-170,

[5] Cherkassky V. (1998), "Fuzzy Inference Systems: A Critical Review", Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, Kayak O, Zadeh LA et al (Eds.), Springer, pp.177-197.

[6] Fogel D. (1999), "Evolutionary Computation: Towards a New Philosophy of Machine Intelligence", 2nd Edition, IEEE press.

[7] Hagan M T, Demuth H B and Beale M H. (1996), "Neural Network Design", Boston, MA: PWS Publishing.

[8] Kasabov N. (1998), "Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation", in Yamakawa T and Matsumoto G (Eds), Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, pp. 271-274.

[9] Mamdani E H and Assilian S. (1975), "An experiment in Linguistic Synthesis with a Fuzzy Logic Controller", International Journal of Man-Machine Studies, Vol. 7, No.1, pp. 1-13.

[10] Moller A F. (1993), "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", Neural Networks, Volume (6), pp. 525-533.

[11] Philip N S and Joseph K B (2001), On the Predictability of Rainfall in Kerala: An Application of ABF Neural Network, Computational Sciences (ICCS 2001), Lecture Notes in Computer Science LNCS 2074, Springer Verlag, Germany, pp. 400-408.

[12] Philip N S and Joseph K B. (2001), "Adaptive Basis Function for Artificial Neural Networks", Neurocomputing Journal, (Accepted for publication).

[13] Zadeh L A. (1998), "Roles of Soft Computing and Fuzzy Logic in the Conception, Design and Deployment of Information/Intelligent Systems", Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, O Kaynak, LA Zadeh, B Turksen, IJ Rudas (Eds.), pp1-9.

[14] Zurada J M. (1992), "Introduction to Artificial Neural Systems", West Publishing.

[15] Lorenz E N., (1963), Deterministic Nonperiodic Flow, Journal of the Atmospheric Sciences 20, 130-141

List of tables

Table 1. Test results and performance comparison of rainfall forecasting

	EFuNN	ANN (BP)	ANN (VLR)	ANN (SCGA)	ANN (ABF)
Learning epochs	1	10000	10000	600	1000
Training error (RMSE)	0.0006	0.0954	0.0875	0.0780	0.0800
Testing error (RMSE)	0.0901	0.0948	0.0936	0.0923	0.0930
Computational load (in billion flops)	0.065	8.82	8.75	1.26	-

List of Figures

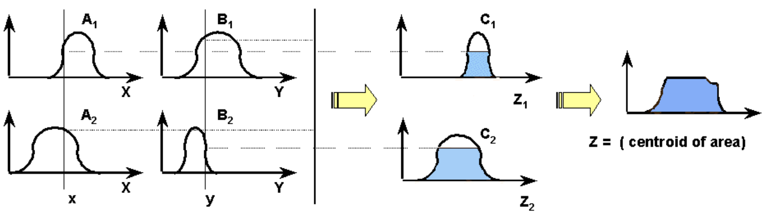

Figure 1. Mamdani fuzzy inference system

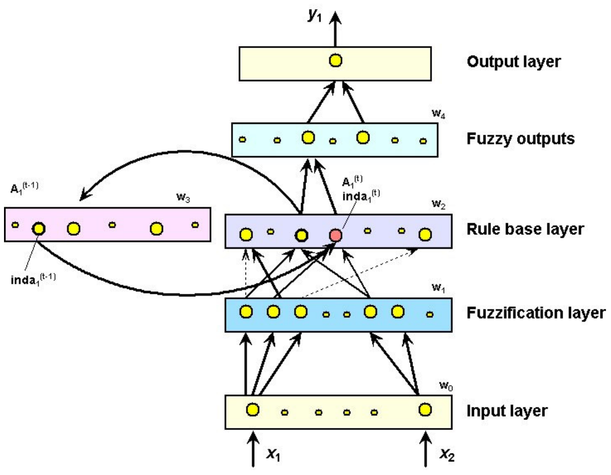

Figure 2. Architecture of EFuNN

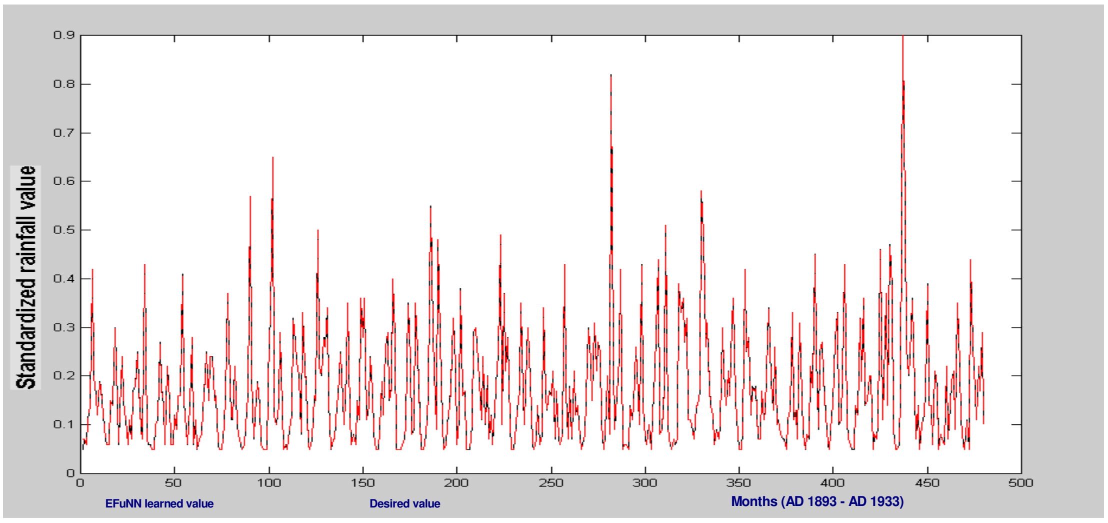

Figure 3. Training results for EFuNN

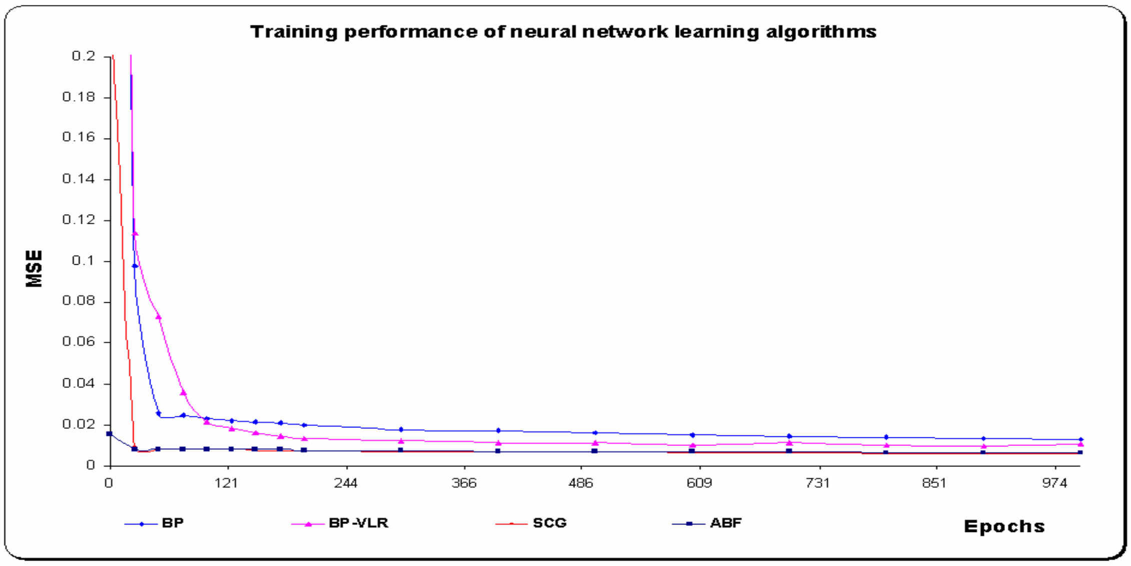

Figure 4. Convergence of neural network learning algorithms (for 1000 epochs).

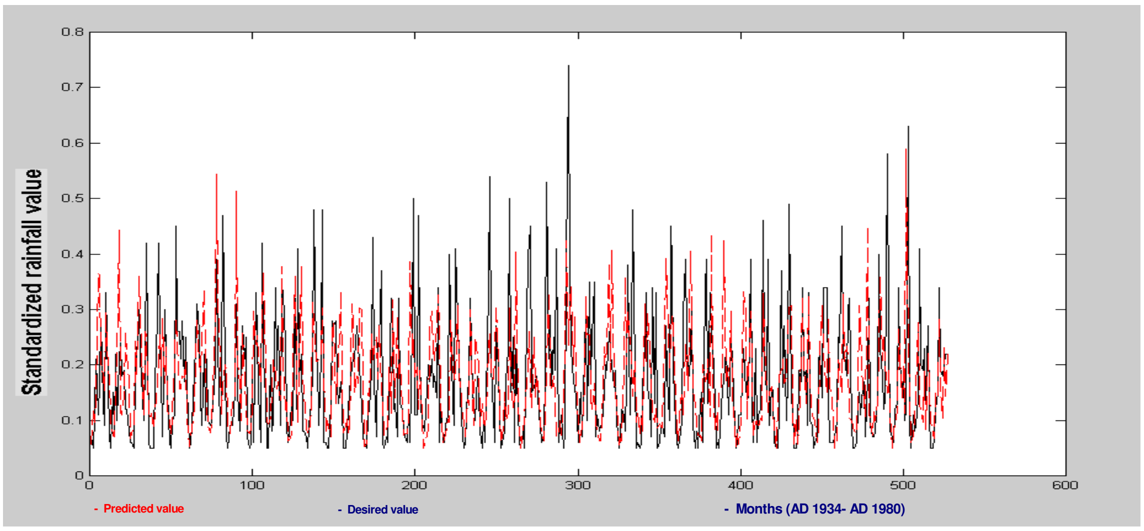

Figure 5. Test results showing the forecast and desired values using EFuNN

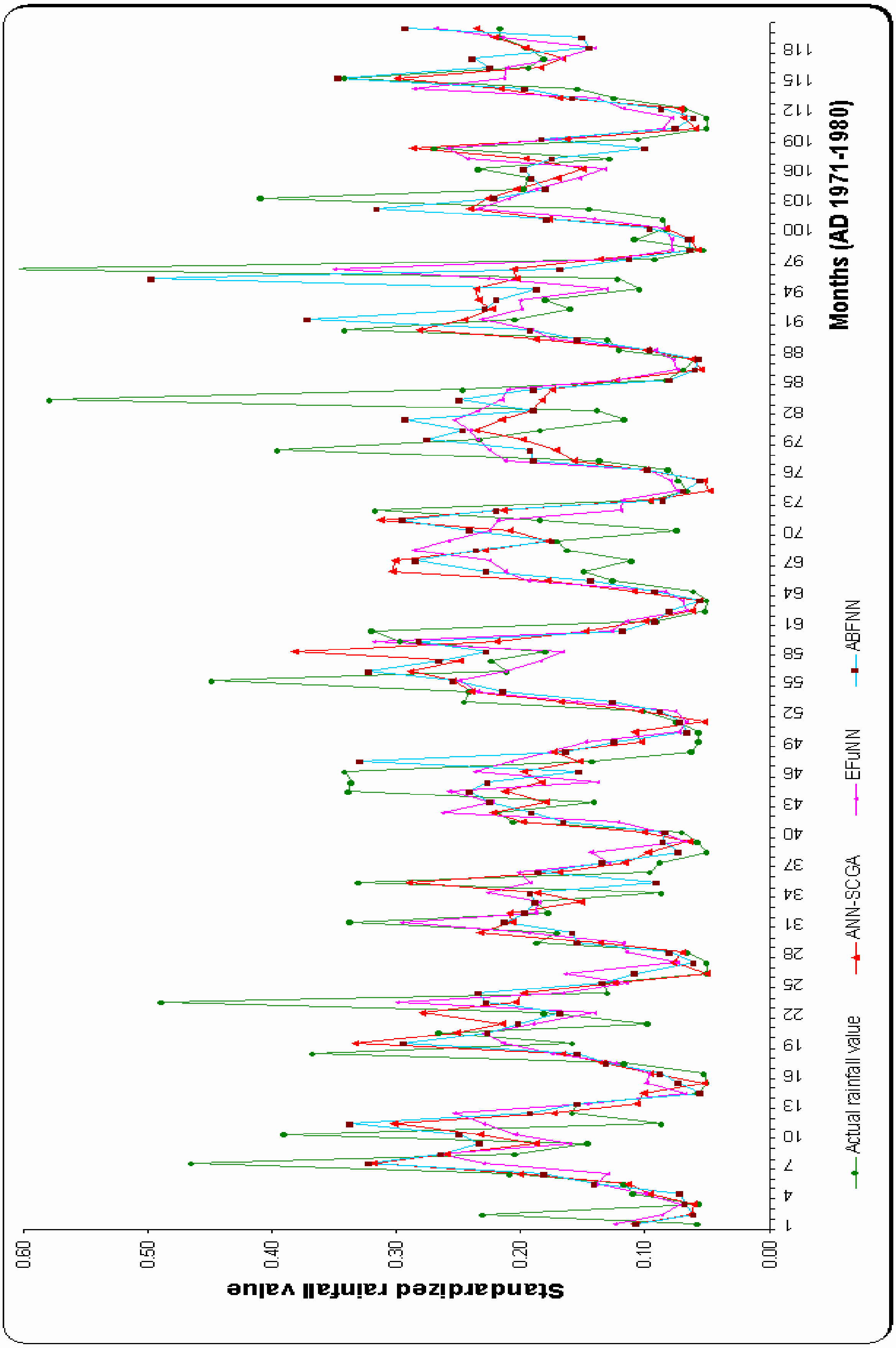

Figure 6. Test results showing monthly prediction of rainfall for 10 years using the different SC models