Jordan Pi-Sigma Neural Network for Temperature Prediction

Noor Aida Husaini[1], Rozaida Ghazali[1],
Nazri Mohd Nawi[1], and Lokman Hakim Ismail[2]

[1] Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johore, Malaysia
[2] Faculty of Environmental and Civil Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johore, Malaysia
`gi090003@siswa.uthm.edu.my`,
`{rozaida,nazri,lokman}@uthm.edu.my`

Abstract. This study examines and analyses the use of a new recurrent neural network model: Jordan Pi-Sigma Network (JPSN) as a forecasting tool. JPSN's ability to predict future trends of temperature was tested and compared to that of Multilayer Perceptron (MLP) and the standard Pi-Sigma Neural Network (PSNN); trained with the standard gradient descent algorithm. A set of historical temperature measurement for five years from Malaysian Meteorological Department was used as input data to train the networks for the next-day prediction. Simulation results show that JPSN forecast comparatively superior to MLP and PSNN models, with lower prediction error, thus revealing a great potential in predicting the temperature measurement.

Keywords: Jordan Neural Network, Recurrent Neural Network, Pi-Sigma Neural Network, Higher Order Neural Network, Temperature Forecasting.

1 Background

Forecasts are made in many disciplines, among them are economics [1], [2], weather forecast [3], [4], disease forecasting [5], [6], and technology forecasting [7], [8]. Accordingly, temperature forecasting, the subject of this study, is the essence of traceability for weather forecasting. Certainly, temperature can be defined qualitatively as a measure during the day, by considering corresponding hotness or coldness [9] that lies on a predetermined set of values. The temperature can have a greater influence in daily life than any other single element on a routine basis. Therefore, precise temperature measurement is needed to obtain the accuracies [10]. Being located in the equatorial region, the annual temperature range in Malaysia is relatively small [11] which is around 26.5ºC. It varies over the lowland area. The lowest temperature recorded was in Kuala Krai Meteorological Station with the reading of 18.0°C. Meanwhile, the highest temperature recorded was in Sri Aman Meteorological Station with the reading of 34.6°C [11].

Temperature forecasting is mainly issued in qualitative terms with the use of conventional methods, assisted by the data projected images taken by meteorological satellites to assess future trends [4], [12]. It is noted that numerical models products are later use as input to obtain a temperature forecast. There are many different scientific efforts in the domain of meteorological parameters forecasting (temperature, solar radiation, humidity, etc.) [11] has been applied; however, those techniques involve sophisticated mathematical models to justify the use of empirical rules and require a prior knowledge of the characteristics of the input time-series to predict future events. The potential of achieving outstanding performance by utilising neural network (NN) has captured a great attention from forecasters, researchers, scientists and engineers. NN theory grew out of Artificial Intelligence (AI) research, which naturally adapts special characteristics from human intelligence, by learning in a manner similar to the human brain. Haykin [13] describes NN as an adaptive machine that has a natural tendency for storing experiential knowledge by constructing a nonlinear mapping between input-output data.

Various studies in temperature forecasting have been applied and discussed by many authors. Bhardwaj *et al.* [12] used the Direct Model Output (DMO) which forecast values for each location of interest that were obtained using several atmospheric parameters. On the other hand, Smith *et al.* [14] noted that ward-style NN can be used for predicting the temperature. Meanwhile, Radhika and Shashi [15] used Support Vector Machine (SVM) for one-step-ahead prediction. They found that SVM could be used for weather forecasting. Finally, Wang and Chen [16] found that automatic clustering techniques and two-factor higher-order fuzzy can be used to cluster the historical numerical data into intervals of different lengths.

Until recently, the widely used NN, Multilayer Perceptron (MLP) also received a great attention in many fields such as prediction [14], [15], pattern recognition [17], signal processing [18], [19], and classification [20], [21]. However, MLP adopts computationally intensive training algorithms and relatively slow when the hidden layer increases [2], [22]. It also introduces many local minima in the error surface [23]. This is then; the development of Higher Order Neural Network (HONN) has captured researchers' attention. Pi-Sigma Neural Network (PSNN) which lies within this area, has the ability to converge faster and maintain the high learning capabilities of HONN [1], [2]. The uses of PSNN itself for temperature forecasting are preferably acceptable. However, in this study we disposed towards an idea to develop a new network model: Jordan Pi-Sigma Network (JPSN) to overcome such drawbacks in MLP and takes the advantages of PSNN. Presently, JPSN is use to learn the historical temperature data of Batu Pahat, and to predict the temperature measurements for the next-day.

The remaining parts of the paper are broken up into the following sections: Neural Networks, Learning Algorithm of the JPSN, Research Design of Neural Network Models, Results and Discussions, and Conclusions. The Neural Networks review the network models that being used in this study. The learning algorithm of the JPSN describes the step-by-step process to train the network. Results and discussions compile pertinent tables and graphical review of the information presented. The paper is ended with the conclusions.

2 Neural Networks

NN is essentially a parametric model that can be one of the very useful tools for solving many practical forecasting problems. This section describes the NN models used in this study.

2.1 Multilayer Perceptron

MLP is formed by a set of sensory units that represent the input layers, one or more hidden layers, and an output layer. The MLP, which is typically used in supervised learning problem, have the ability to learn from input-output pairs [24]. The standard MLP is usually trained with the standard gradient descent algorithm [25]. The input signal propagates through the network layer-by-layer [13] through weighted-sum [26] and sigmoid activation function $1/\left(1+e^{-x}\right)$ of the hidden and output units. The sigmoid function is often chosen because it is mathematically convenient and close to linear near origin which prevents accelerating growth throughout the network [27], thus allowing the networks to model well on nonlinear problems [28]. The computation of the network error is done at the output layer, and the error is propagated back to the input layer by updating weights consequently, right after each input is presented to the network. This training process is iterated until a stepwise change of the weights have converged [25]. It is noted the MLP has been successfully applied in many applications involving pattern recognition [17], signal processing [18], [19], and classification [20], [21]. However, the ordinary MLP is extremely slow due to its multilayer architecture especially when the hidden layer increases [22]. It also converges very slow in typical situations principally when dealing with complex and nonlinear problems; and does not scale well with problem size [22].

2.2 Pi-Sigma Neural Network

PSNN is a type of HONN which comprises of a single layer of learnable weights [22]. PSNN consists of two layers of units; the product unit and the summing unit layers. The weighted inputs are fed to a linear summing unit, and to the output layer. The number of the summing units in PSNN reflects the network order. By using an additional summing unit, it will increase the network's order by 1. In PSNN, weights from hidden layer to the output layer are fixed to 1. This property drastically reduces the training time of PSNN. Sigmoid and linear functions are adopted at the output layer and summing layer, respectively. The use of linear summing units at the hidden layer makes the convergence analysis of the learning rules for the PSNN more accurate and tractable [2]. The applicability of this network was successfully applied for function approximation [2], classification [22], pattern recognition [29], cryptography scheme [30], and so forth. Compared to other HONN models, PSNN has a more simple structure [2] and can converge faster whilst maintain the high learning capabilities of HONN [1], [2]. Moreover, PSNN is superior to the MLP with at least two orders of magnitude less number of computations when compared to the ordinary MLP for similar performance levels, and over a broad class of problems [1].

2.3 Jordan Neural Network

The Jordan Neural Network (JNN), which was proposed by Jordan [31], is a type of recurrent neural network (RNN). The motivation for exploring JNN is their potential of maintaining internal state to generate temporal sequences. The output of JNN which is clocked back to the input layers are called context unit. Each unit receives a copy from the output neurons and from themselves [31]. The recurrent connections from the output layer to the context neurons have an associated parameter, m which lies between $[0,1]$. The feedback connection concatenates the input-output layer, therefore, allows the network to posses short term memory for temporal pattern recognition over a time series. The activation at time t is given by the following expression:

$$c(t) = mc(t-1) + x(t-1) \tag{1}$$

where $x(t-1)$ is the output network at time $t-1$. The activation function for the rest nodes is calculated similar to that of MLP. By considering the expression of the context neuron activation, it is probable to write:

$$c(t) = \sum_{j=1}^{t-1} \mu^{j-1} x(t-1) \tag{2}$$

Consequently, the parameter m endows to the JNN with certain inertia of the context neurons. To note, the parameter m resolves the sensitivity of the context neurons to preserve this information [31]. Thus, the recurrence on the context units provides a decaying history of the output over the most recent time steps. The pertinency of JNN has been tested through [32], [33], [34] for different kind of problems.

2.4 Jordan Pi-Sigma Network

A new type of recurrent HONN, called the Jordan Pi-Sigma Network (JPSN) is proposed in this research work. The JPSN will be used for temperature prediction, and the network will be trained with the standard gradient descent algorithm [25]. The architecture of JPSN is constructed by having a recurrent link from output layer back to the input layer which gives the temporal dynamics of the time series process [35]. The network combines the properties of both HONN and RNN so that better performance can be achieved. Additionally, the unique architecture of JPSN may also avoid from the combinatorial explosion of higher-order terms as the network order increases.

The structure of JPSN is quite similar to the ordinary PSNN. The main difference is the integration of a recurrent or feedback link from the output to the input layer thus allowing for computational structure in a more parsimonious way [35]. Weights between the input layers and summing layers are trainable, while weights between the summing layers and the output layer are fixed to unity. In this work, the number of hidden nodes for MLP, and the number of higher order terms for PSNN and JPSN, are set between 2 and 5. The architecture of the proposed JPSN is shown in Fig. 1.

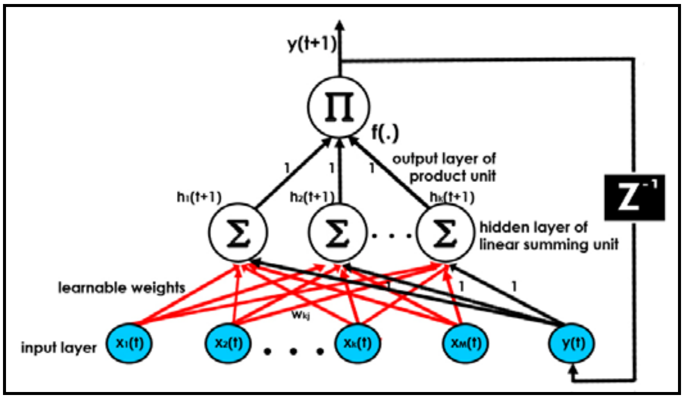


Fig. 1. The JPSN (z^{-1} denotes the time delay operator)

3 Learning Algorithm of the JPSN

The learning algorithm for JPSN is calculated as follows:

$$h_L(t+1)=\sum_{m=1}^{m} w_{Lm}x_m(t)+w_{L(M+1)}+w_{L(M+2)}y(t)=\sum_{m=1}^{m+2} w_{Lm}z_m(t) \tag{3}$$

where $h_L(t+1)$ represents the activation of the L unit at time $t+1$, and $y(t)$ is the previous network output. The unit's transfer function f denotes sigmoid function, which bounded of output range of $[0,1]$ is used in this network.

```
For each training example, do:
    1. Calculate the output.
```

$$y(t+1)=f\left(\prod_{L=1}^{k} h_L(t+1)\right) \tag{4}$$

```
        where $y(t+1)$ is the network output.
    2. Compute the error $e$ at output node:
```

$$e(t+1)=d(t+1)-y(t+1) \tag{5}$$

```
    3. Compute the weight changes:
```

$$\Delta w_{ij}(t+1)=-\eta\frac{\partial J(t+1)}{\partial w_{ij}}+\alpha\Delta w_{ij}(t) \tag{6}$$

```
        where $\eta$ is the learning rate and $\alpha$ is the momentum term.
```

```
4.  Update the weight:
                         $W_i = W_i + \Delta W_i$                    (7)

5.  If current epoch > maximum epoch
            stop the training
    else
            go to step 1
```

4 Research Design of Neural Network Models

4.1 Data Description

We are considering the univariate input data; the 5-years daily measurement of temperature in Batu Pahat region, ranging from 2005 to 2009. The data was obtained from Central Forecast Office, Malaysian Meteorological Department (MMD).

4.2 Experimental Design

All networks were built considering 5 different number of input nodes ranging from 4 to 8. A single neuron was considered for the output layer. The number of hidden layer (for MLP), and higher order terms (for PSNN and JPSN) was initially started with 2 nodes, and increased by one until a maximum of 5 nodes. The combination of 4 to 8 input nodes and 2 to 5 nodes for hidden layer/higher order terms of the three network models yields a total of 1215 NN architectures for each in-sample training dataset. Since the forecasting horizon is one-step-ahead, the output variable represents the temperature measurement of one-day ahead.

Each of data series is segregated in time order and is divided into three sets; the training, validation and the out-of-sample data. To avoid computational problems, the data is normalised between the upper and lower bounds of the network transfer function, which often to be the monotonically increasing function, $1/\left(1+e^{-x}\right)$ [27]. The interconnections weights in each neuron are adjusted to minimise the error by using standard gradient descent algorithm [25]. To be more certain of getting the true global minima, it is practical to initialise the weights with a small random number. The reason to initialise weights with small values between $[0,1]$ is to prevent saturation for all patterns and the insensitivity to the training process. On the other hand, if the initial weights are set too small, training will tend to start very slow. The training set serves the model for training purpose and the testing set is used to evaluate the network performances [28]. Training involves the adjustment of the weights so that the NN is capable to predict the value assigned to each member of the training set. The network is validated and tested in order to preserve some features for calibration purpose [28], which means producing appropriate outputs for those input samples which were not encountered during training [4].

4.3 Assessment Criteria

There is no consensus on the most appropriate measure to assess the performance of a forecasting technique. Network models performance were evaluated using Mean Squared Error (MSE), Signal to Noise Ratio (SNR) [1], Mean Absolute Error (MAE) [36] and Normalised Mean Squared Error (NMSE) [1], which are expressed by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(P_i - P_i^*\right)^2 \tag{8}$$

$$SNR = 10 * \lg(sigma)$$

$$sigma = \frac{m^2 * n}{SSE}$$

$$SSE = \sum_{i=1}^{n}\left(P_i - P_i^*\right)$$

$$m = \max(P) \tag{9}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|P_i - P_i^*\right| \tag{10}$$

$$NMSE = \frac{1}{\sigma^2 n}\sum_{i=1}^{n}\left(P_i - P_i^*\right)^2$$

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(P_i - P_i^*\right)^2$$

$$P^* = \sum_{i=1}^{n} P_i \tag{11}$$

where n is the total number of data patterns, P_i and P_i^* represent the actual and predicted output value [1], [36]. In addition, we also consider the number of iterations during the training process.

5 Results and Discussions

All networks training and testing have been implemented using MATLAB 7.10.0 (R2010a) on Pentium® Core ™^{2} Quad CPU. Pattern selection was set randomly and stopping criteria was set to 3000 events. All results presented were based on the average of 10 simulations/runs. The prediction performances of JPSN are instantiated with PSNN and MLP based on the aforementioned performance metrics. Table 1 shows the performance of JPSN, PSNN and MLP. Over all the training patterns, the JPSN obtained the lowest MAE, which is 0.063458; while the MAE for PSNN and MLP were 0.063471 and 0.063646, respectively. JPSN outperformed PSNN by ratio 1.95×10^{-4}, and 2.9×10^{-3} for the MLP. Moreover, it can be seen that JPSN reached higher value of SNR. Therefore, it can be concluded that the networks can track the signal better than PSNN and MLP.

Table 1. Comparison of JPSN, PSNN and MLP for all performance metrics

Network Model	JPSN	PSNN	MLP
MAE	0.063458	0.063471	0.063646
SNR	18.7557	18.71039	18.69706
NMSE	0.771034	0.779118	0.781514
MSE Training	0.006203	0.006205	0.00623
MSE Testing	0.006462	0.006529	0.006549
Epoch	1460.9	1211.8	2849.9

The models' performances were also evaluated by comparing their NMSE. Fig. 2 illustrates the NMSE for the three network models. It shows that JPSN fashionably gives better NMSE when compared to both PSNN and MLP.

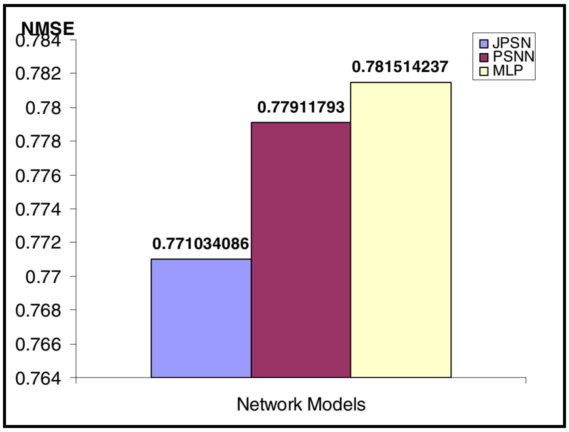
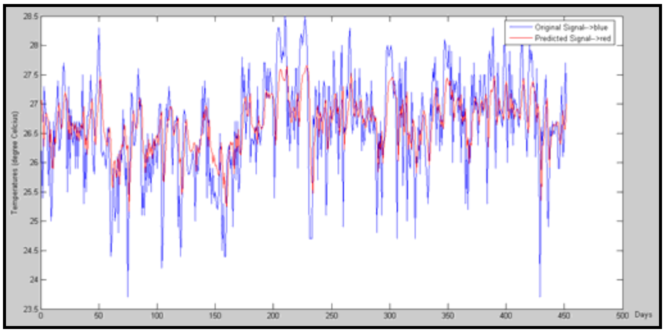
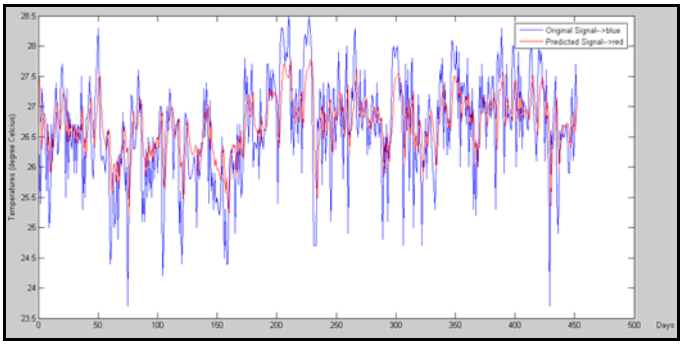


Fig. 2. NMSE for JPSN, PSNN and MLP

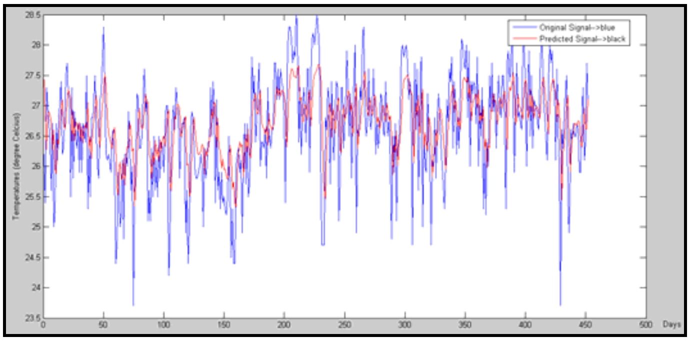
The graphs depicted in Fig. 3 present the prediction errors on the testing data set for all network models. As expected, differences between the predicted and the actual data for JPSN practically beat out PSNN and MLP by 1.038% and 1.341%, respectively. Evaluations on different performance metrics over the temperature data demonstrated that JPSN were merely improved the performance level compared to the two benchmarked network models, PSNN and MLP.



a) Temperature Forecast made by JPSN



b) Temperature Forecast made by PSNN



c) Temperature Forecast made by MLP

Fig. 3. Temperature Forecast made by JPSN, PSNN and MLP on the Testing Set

6 Conclusions

This study focuses on using JPSN as an alternative mechanism to predict the temperature, which is less memory required during the network training. The network combines the properties of both PSNN and RNN. On the whole, JPSN is able to show a better performance level compared to the ordinary PSNN and MLP. It is concluded that JPSN has the capability to forecast temperature and, if properly trained, the meteorological station could benefit from the use of this superior forecasting tool. Looking for the future, we are delighted to consider the hybridisation of more than one meteorological parameters in order to improve the prediction of future temperature events.

Acknowledgments. The authors would like to thank Universiti Tun Hussein Onn Malaysia for supporting this research under the Postgraduate Incentive Research Grant.

References

1. Ghazali, R., Hussain, A., El-Dereby, W.: Application of Ridge Polynomial Neural Networks to Financial Time Series Prediction. In: International Joint Conference on Neural Networks (IJCNN 2006), Vancouver, BC, pp. 913–920 (2006)
2. Ghazali, R., Al-Jumeily, D.: Application of Pi-Sigma Neural Networks and Ridge Polynomial Neural Networks to Financial Time Series Prediction. In: Zhang, M. (ed.) Artificial Higher Order Neural Networks for Economics and Business. Information Science Reference, pp. 271–293 (2009)
3. Lorenc, A.C.: Analysis Methods for Numerical Weather Prediction. Quarterly Journal of the Royal Meteorological Society 112, 1177–1194 (1986)
4. Paras, Mathur, S., Kumar, A., Chandra, M.: A Feature Based Neural Network Model for Weather Forecasting. In: Proceedings of World Academy of Science, Engineering and Technology. pp. 67–73 (2007)
5. Cooke, B.M., Jones, D.G., Kaye, B., Hardwick, N.V.: Disease Forecasting. In: The Epidemiology of Plant Diseases, pp. 239–267. Springer, Netherlands (2006)
6. Coombs, A.: Climate Change Concerns Prompt Improved Disease Forecasting. Nature Medicine 14, 3 (2008)
7. Byungun, Y., Yongtae, P.: Development of New Technology Forecasting Algorithm: Hybrid Approach for Morphology Analysis and Conjoint Analysis of Patent Information. IEEE Transactions on Engineering Management 54, 588–599 (2007)
8. Cheng, A.-C., Chen, C.-J., Chen, C.-Y.: A Fuzzy Multiple Criteria Comparison of Technology Forecasting Methods for Predicting the New Materials Development. Technological Forecasting and Social Change 75, 131–141 (2008)
9. Childs, P.R.N.: Temperature. In: Practical Temperature Measurement., pp. 1–15. Butterworth-Heinemann, Butterworths (2001)
10. Ibrahim, D.: Temperature and its Measurement. In: Microcontroller Based Temperature Monitoring & Control, Newnes, pp. 55–61 (2002)
11. Malaysian Meteorological Department, `http://www.met.gov.my`

12. Bhardwaj, R., Kumar, A., Maini, P., Kar, S.C., Rathore, L.S.: Bias-free Rainfall Forecast and Temperature Trend-based Temperature Forecast using T-170 Model Output during the Monsoon Season. Meteorology & Atmospheric Sciences 14, 351–360 (2007)
13. Haykin, S.: Neural Networks - A Comprehensive Foundation. Prentice-Hall, Englewood Cliffs (1998)
14. Smith, B.A., Hoogenboom, G., McClendon, R.W.: Artificial Neural Networks for Automated Year-Round Temperature Prediction. Computers and Electronics in Agriculture 68, 52–61 (2009)
15. Radhika, Y., Shashi, M.: Atmospheric Temperature Prediction using Support Vector Machines. International Journal of Computer Theory and Engineering 1, 55–58 (2009)
16. Wang, N.-Y., Chen, S.-M.: Temperature Prediction and TAIFEX Forecasting based on Automatic Clustering Techniques and Two-Factors High-Order Fuzzy Time Series. Expert Systems with Applications 36, 2143–2154 (2009)
17. Isa, N.A.M., Mamat, W.M.F.W.: Clustered-Hybrid Multilayer Perceptron Network for Pattern Recognition Application. Applied Soft Computing 11, 1457–1466 (2011)
18. Nielsen, J.L.G., Holmgaard, S., Ning, J., Englehart, K., Farina, D., Parker, P.: Enhanced EMG Signal Processing for Simultaneous and Proportional Myoelectric Control. In: Annual International Conference of the IEEE, Engineering in Medicine and Biology Society, EMBC 2009, pp. 4335–4338 (2009)
19. Li, H., Adali, T.: Complex-Valued Adaptive Signal Processing using Nonlinear Functions. EURASIP J. Adv. Signal Process, 1–9 (2008)
20. Yuan-Pin, L., Chi-Hong, W., Tien-Lin, W., Shyh-Kang, J., Jyh-Horng, C.: Multilayer Perceptron for EEG Signal Classification during Listening to Emotional Music. In: TENCON 2007 - 2007 IEEE Region 10 Conference, pp. 1–3 (2007)
21. Belacel, N., Boulassel, M.R.: Multicriteria Fuzzy Classification Procedure PROCFTN: Methodology and Medical Application (2003)
22. Shin, Y., Ghosh, J.: The Pi-Sigma Networks: An Efficient Higher-Order Neural Network for Pattern Classification and Function Approximation. In: Proceedings of International Joint Conference on Neural Networks, Seattle, WA, USA, pp. 13–18 (1991)
23. Yu, W.: Back Propagation Algorithm. Psychology/University of Newcastle (2005)
24. Güldal, V., Tongal, H.: Comparison of Recurrent Neural Network, Adaptive Neuro-Fuzzy Inference System and Stochastic Models in Eğirdir Lake Level Forecasting. Water Resources Management 24, 105–128 (2010)
25. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
26. Rumbayan, M., Nagasaka, K.: Estimation of Daily Global Solar Radiation in Indonesia with Artificial Neural Network (ANN) Method. In: Proceedings of International Conference on Advanced Science, Engineering and Information Technology (ISC 2011), pp. 190–103. Malaysia National University (2011)
27. Cybenko, G.: Approximation by Superpositions of a Sigmoidal Function. Signals Systems 2, 14 (1989)
28. Shrestha, R.R., Theobald, S., Nestmann, F.: Simulation of Flood Flow in a River System using Artificial Neural Networks. Hydrology and Earth System Sciences 9, 313–321 (2005)
29. Shin, Y., Ghosh, J., Samani, D.: Computationally Efficient Invariant Pattern Recognition with Higher Order Pi-Sigma Networks. In: Intelligent Engineering Systems through Artificial Neural Networks-II, pp. 379–384 (1992)
30. Song, G.: Visual Cryptography Scheme Using Pi-Sigma Neural Networks. In: International Symposium on Information Science and Engineering, pp. 679–682 (2008)

31. Jordan, M.I.: Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In: Proceedings of the Eighth Conference of the Cognitive Science Society, pp. 531–546. IEEE Press, Piscataway (1986)
32. Saha, S., Raghava, G.P.S.: Prediction of Continuous B-cell Epitopes in an Antigen using Recurrent Neural Network. Proteins: Structure, Function, and Bioinformatics 65, 40–48 (2006)
33. Carcano, E.C., Bartolini, P., Muselli, M., Piroddi, L.: Jordan Recurrent Neural Network versus IHACRES in Modelling Daily Streamflows. Journal of Hydrology 362, 291–307 (2008)
34. West, D., Dellana, S.: An Empirical Analysis of Neural Network Memory Structures for Basin Water Quality Forecasting. International Journal of Forecasting (2010) (in press) (corrected proof)
35. Hussain, A.J., Liatsis, P.: Recurrent Pi-Sigma Networks for DPCM Image Coding. Neurocomputing 55, 363–382 (2002)
36. Valverde Ramírez, M.C., de Campos Velho, H.F., Ferreira, N.J.: Artificial Neural Network Technique for Rainfall Forecasting Applied to the São Paulo Region. Journal of Hydrology 301, 146–162 (2005)