A Novel Genetic-Algorithm-Based Neural Network for Short-Term Load Forecasting

S.H. Ling, F.H.F. Leung, H.K. Lam, Y.S. Lee and P.K.S. Tam

Centre of Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong.

Abstract – This paper presents a neural network with a novel neuron model. In this model, the neuron has two activation functions and exhibits a node-to-node relationship in the hidden layer. This neural network provides a better performance than a traditional feed-forward neural network and fewer hidden nodes are needed. The parameters of the proposed neural network are tuned by genetic algorithm (GA) with arithmetic crossover and non-uniform mutation. Some applications are given to show the merits of the proposed neural network.

***Index Terms*-Neural network, genetic algorithm, short-term load forecasting**

I. INTRODUCTION

Neural networks are widely applied in areas such as prediction [19], system modeling and control [18]. Owing to its particular structure, a neural network is good in learning [2] using some algorithms such as genetic algorithm (GA) [1] and back propagation [2]. Traditionally, a feed-forward neural network [24] has 3 layers (input, hidden and output layers) of nodes connected in a layer-to-layer manner.

GA is widely applied in optimization problems [1-7] where the number of parameters is large and the analytical global solutions are difficult to obtain. It has been applied in different areas such as fuzzy control [20], path planning [21], greenhouse climate control [22], modeling and classification [23] etc.

A novel neural network model is proposed in this paper. Two activation functions are used in the neuron and a node-to-node relationship is proposed in the hidden layer. This

network model is found to be able to give a better performance than the traditional feed-forward neural network [1-3]. GA with arithmetic crossover and non-uniform mutation can help tuning the parameter of the proposed network. Numerical examples (3-inputs XOR problem and sunspot number forecasting) are used to test the proposed network and good results are obtained. Two applications are also given, which are short-term daily load forecasting and pattern recognition.

This paper is organized as follows. Section II introduces the proposed neural network. Training of the neural network with GA is presented in section III. Numerical examples will be given in section IV. Applications on short-term daily load forecasting and pattern recognition realized by the proposed GA-based neural network will be presented in section V. A conclusion will be drawn in section VI.

II. Neural Network Model

Fig. 1 shows the proposed neuron. It has two activation functions: static activation functions (SAF) and dynamic activation functions (DAF) that govern the input-output relationships of the neuron. For the SAF, the parameters are fixed and its output depends on the input of the neuron. For the DAF, the parameters depend on the outputs of other neurons and its SAF. With this proposed neuron, the connection of the proposed neural network is shown in Fig. 2, which is a three-layer neural network. A node-to-node relationship is introduced in the hidden layer. Comparing with the traditional feed-forward neural network [24], the proposed neural network can offer a better performance. The merits of the proposed neural network will be shown in the later section.

A. Proposed neuron model

We consider the SAF first. Let v_{ij} be the synaptic connection weight from the *i*-th input node x_i to the *j*-th neuron, the output κ_j of the *j*-th neuron's SAF is given by,

$$\kappa_j = net_s^j(\sum_{i=1}^{n_{in}} x_i v_{ij}),\ i = 1, 2, \ldots, n_{in},\ j = 1, 2, \ldots, n_h \tag{1}$$

where n_{in} denotes the number of input and $net_s^j(\cdot)$ is the *j*-th static activation function.

$$net_s^j(\sum_{i=1}^{n_{in}} x_i v_{ij}) = \begin{cases} e^{\frac{-\left(\sum_{i=1}^{n_{in}} x_i v_{ij} - m_s^j\right)^2}{2\sigma_s^{j2}}} - 1 & \text{if } \sum_{i=1}^{n_{in}} x_i v_{ij} \le m_s^j \\ 1 - e^{\frac{-\left(\sum_{i=1}^{n_{in}} x_i v_{ij} - m_s^j\right)^2}{2\sigma_s^{j2}}} & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, n_h \tag{2}$$

where m_s^j and σ_s^j are the static mean and static standard deviation for the *j*-th SAF respectively. The parameters (m_s^j and σ_s^j) are fixed after the training processing. By using the proposed activation function in (2), the output value is ranged from –1 to 1. The shape of the proposed activation function is shown in Fig. 3. In Fig. 3a, the effect of the mean value to the activation function is shown. The standard deviation σ_s of the function is fixed at 0.2, and the mean value *m* is chosen from –0.4 to 0.4. In Fig. 3b, the effect of the standard deviation to the activation function is shown. The mean value m_s is fixed at 0, and the standard deviation σ_s is chosen from 0.1 to 0.5. It can be observed from these two figures that $net_s(f) \to 1$ as $f \to \infty$, and $net_s(f) \to -1$ as $f \to -\infty$.

For the DAF, the neuron output z_j of the *j*-th neuron is defined as,

$$z_j = net_d^j(\kappa_j),\ j = 1, 2, \ldots, n_h \tag{3}$$

where $net_d^j(\cdot)$ is the *j*-th DAF, which is given by,

$$net_d^j(\kappa_j, m_d^j, \sigma_d^j) = \begin{cases} e^{\frac{-(\kappa_j - m_d^j)^2}{2\sigma_d^{j2}}} - 1 & \text{if } \kappa_j \le m_d^j \\ 1 - e^{\frac{-(\kappa_j - m_d^j)^2}{2\sigma_d^{j2}}} & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, n_h \tag{4}$$

where

$$m_d^j = p_{j+1,j} \times \kappa_{j+1} \tag{5}$$

$$\sigma_d^j = p_{j-1,j} \times \kappa_{j-1} \tag{6}$$

m_d^j and σ_d^j are the dynamic mean and dynamic standard deviation for the *j*-th DAF; κ_{j-1} and κ_{j+1} represent the SAF's output of the *j*–1-th and *j*+1-th neurons respectively; $p_{j+1,j}$ denotes the weight of the link between the *j*+1-th node and the *j*-th node, and $p_{j-1,j}$ denotes the weight of the link between the $j-1$-th node and the *j*-th node. It should be noted from Fig. 1 that if $j=1$, $p_{j-1,j}$ is equal to $p_{n_h,j}$, and if $j=n_h$, $p_{j+1,j}$ is equal to $p_{1,j}$. Unlike the SAF, the DAF is dynamic as its parameters depend on the outputs of the *j*–1-th and *j*+1-th neurons. Referring to (1-6), the input-output relationship of the proposed neuron is as follows:

$$z_j = net_d^j(net_s^j(\sum_{i=1}^{n_{in}} x_i v_{ij})),\ j = 1, 2, \ldots, n_h \tag{7}$$

B. Connection of the proposed neural network

The proposed neural network has three layers with n_{in} nodes in the input layer, n_h nodes in the hidden layer, and n_{out} nodes in the output layer. In the hidden layer, the neuron model presented in the previous section is employed. A node-to-node relationship is introduced in the hidden layer. In the output layer, a static activation function is employed, which is given by,

$$y_l = net_o^l(\sum_{j=1}^{n_h} z_j w_{jl}),\ l = 1, 2, \ldots, n_{out}. \tag{8}$$

From (7), we have, $y_l = net_o^l(\sum_{j=1}^{n_h} net_d^j(net_s^j(\sum_{i=1}^{n_{in}} x_i v_{ij}))w_{jl})$ (9)

where w_{jl}, j = 1, 2, …, n_h; l = 1,2, … n_{out}, denotes the weight of the link between the *j*-th hidden and the *l*-th output nodes; $net_o^l(\cdot)$ denotes the activation function of the output neuron:

$$net_o^l\left(\sum_{j=1}^{n_h} z_j w_{jl}\right) = \begin{cases} e^{\frac{-\left(\sum_{j=1}^{n_h} z_j w_{jl} - m_o^l\right)^2}{2\sigma_o^{l\,2}}} - 1 & \text{if } \sum_{j=1}^{n_h} z_j w_{jl} \le m_o^l \\ 1 - e^{\frac{-\left(\sum_{j=1}^{n_h} z_j w_{jl} - m_o^l\right)^2}{2\sigma_o^{l\,2}}} & \text{otherwise} \end{cases} \tag{10}$$

where m_o^l and σ_o^l are the mean and the standard deviation of the output node activation function respectively. The parameters of the proposed neural network can be trained by GA.

III. Training With Genetic Algorithm

In this section, the proposed neural network is employed to learn the input-output relationship of an application using GA. This GA is implemented with arithmetic crossover and non-uniform mutation [3]. A population of chromosomes *P* is initialized and then evolves. First, two parents are selected from *P* by the method of spinning the roulette wheel [3]. Then a new offspring is generated from these parents using crossover and mutation operations, which are governed by the probabilities of crossover and mutation respectively. These probabilities are chosen by trial and error through experiments for good performance. The new population thus generated replaces the current population. The above procedures are repeated until a certain termination condition is satisfied. The termination condition may be that the algorithm stops when a predefined number of generations have been processed.

Let the input-output relationship of an application be described by,

$$\mathbf{y}^d(t) = \mathbf{g}\left(\mathbf{x}^d(t)\right),\ t = 1, 2, \ldots, n_d \tag{11}$$

where $\mathbf{y}^d(t) = \left[y_1^d(t) \quad y_2^d(t) \quad \cdots \quad y_{n_{out}}^d(t)\right]$ is the desired output corresponding to the input $\mathbf{x}^d(t) = \left[x_1^d(t) \quad x_2^d(t) \quad \cdots \quad x_{n_{in}}^d(t)\right]$ of an unknown nonlinear function $\mathbf{g}(\cdot)$; n_d denotes the number of input-output data pairs. The fitness function is defined as,

$$fitness = \frac{1}{1 + err} \tag{12}$$

$$err = \frac{\sum_{k=1}^{n_{out}} \sum_{t=1}^{n_d} \left| y_k^d(t) - y_k(t) \right|}{n_d \times n_{out}} \tag{13}$$

The objective is to maximize the fitness value of (12) using GA by setting the chromosome to be $[v_{ij} \quad m_s^j \quad \sigma_s^j \quad p_{j+1,j} \quad p_{j-1,j} \quad w_{jl} \quad m_o^l \quad \sigma_o^l]$ for all *i*, *j*, *l*. In this paper, $v_{ij}, p_{j+1,j}, p_{j-1,j}, w_{jl} \in [-1 \quad 1]$, $m_s^j, m_o^l \in [-0.5 \quad 0.5]$ and $\sigma_s^j, \sigma_o^l \in [0.01 \quad 0.5]$. The range of the fitness function of (12) is [0, 1]. By using the proposed GA-based neural network, a well-trained neural network with respect to the fitness value can be obtained.

IV. EXAMPLES

In this section, two examples are given: the XOR problem and the forecasting of sunspot number.

A. XOR Problem

The three-input XOR function, which is not linearly separable, has the following input-output relationship:

$$\begin{cases} (-1,-1,-1),(-1,+1,+1),(+1,-1,+1),(+1,+1,-1) \rightarrow -1 \\ (-1,-1,+1),(-1,+1,-1),(+1,-1,-1),(+1,+1,+1) \rightarrow +1 \end{cases} \tag{14}$$

The three inputs of the proposed neural network are defined as $x_i(t)$, i = 1, 2, 3 and $y(t)$ is the network output. The number of hidden nodes (n_h) is set at 3. Referring to (9), the proposed neural network used for the three-inputs XOR classification problem is governed by,

$$y(t) = net_o\left(\sum_{j=1}^{3} net_d^j\left(net_s^j\left(\sum_{i=1}^{3} x_i v_{ij}\right)\right) w_j\right) \tag{15}$$

The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \tag{16}$$

$$err = \frac{\sum_{t=1}^{8} \left| y^d(t) - y(t) \right|}{8} \tag{17}$$

GA is employed to tune the parameters of the proposed neural network of (15). The objective is to maximize the fitness function of (16). The chromosomes used for the GA are $[v_{ij} \quad m_s^j \quad \sigma_s^j \quad p_{j+1,j} \quad p_{j-1,j} \quad w_j \quad m_o \quad \sigma_o]$ for all i and j. The initial values of the parameters of the neural network are randomly generated. For comparison purpose, a traditional feed-forward neural network (3-input-single-output) trained by GA is also used to solve the three-input XOR classification problem. The number of hidden node of the traditional neural network is 5. (By using 5 hidden nodes for the traditional neural network, the number of parameters to be tuned is the same as that of the proposed neural network, which is 26). The population size used for the GA is 10 and the number of iterations to train the neural network is 1000 for both networks. The probabilities of crossover and mutation for GA are set at 0.8 and 0.35 respectively, which are chosen by trial and error. The results of the proposed and traditional neural networks are tabulated in Table I and shown in Fig. 4. It can be seen that the performance of the proposed neural network is better.

B. Forecasting the Sunspot Number

The sunspot numbers from 1700 to 1980 exhibit non-linear, non-stationary, and non-Gaussian cycles that are difficult to model and predict. We use the proposed neural network (3-input-single-output) for the sunspot number forecasting. The inputs, x_i, of the proposed neural network are defined as $x_1(t) = y_1^d(t-1)$, $x_2(t) = y_1^d(t-2)$ and $x_3(t) = y_1^d(t-3)$, where t denotes the year and $y_1^d(t)$ is the sunspot numbers at the year t. The sunspot numbers of the first 180 years (i.e. $1705 \le t \le 1884$) are used to train the proposed neural network. Referring to (9), the proposed neural network used for the sunspot forecasting is governed by,

$$y_1(t) = net_o^1\left(\sum_{j=1}^{3} net_d^j\left(net_s^j\left(\sum_{i=1}^{3} x_i v_{ij}\right)\right) w_{j1}\right) \tag{18}$$

The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \tag{19}$$

$$err = \sum_{t=1705}^{1884} \frac{\left| y_1^d(t) - y_1(t) \right|}{180} \tag{20}$$

GA is employed to tune the parameters of the proposed neural network of (18). The result is again compared with that from a traditional 3-input-single-output neural network. The numbers of hidden nodes of the proposed network and the traditional network are 3 and 9 respectively, which are chosen by trial and error through experiments for good performance. The population size used for the GA is 10 and the number of iterations to train the neural network is 1000. The probability of crossover is set at 0.8 for both networks. The probability of mutation is set at 0.3 and 0.25 for the proposed and the traditional networks respectively. The tuned neural networks are used to forecast the sunspot number of the years 1885-1979. The fitness value, training error (governed by (20)) and the forecasting error (governed by $\sum_{t=1885}^{1980} \frac{\left| y_1^d(t) - y_1(t) \right|}{96}$) are tabulated in Table II. The proposed neural network once again gives a better performance.

V. Applications

In this section, two application examples are given, which are short-term daily home electric load forecasting and pattern recognition.

A. Short-Term Daily Home Electric Load Forecasting

In the intelligent home system [8], the AC power line network is used not only for supplying power, but also serving as the data communication channel for electrical appliances. With this AC power line data network, a short-term load forecasting can be realized. An accurate load forecasting can bring the following benefits: 1) increasing the reliability [4] of the AC power line data network, and 2) optimizing electric load scheduling. Artificial neural network have been considered as a very promising tool to short-term electric

load forecasting [9-17]. However, the gradient-descent (GD) algorithms for parameters training of the feed-forward neural networks suffer from the common problems of convergence to local minima and sensitivity to initial values of the parameters. Global search techniques such as GA may solve these problems.

The idea of the daily electric load forecasting is to construct seven multi-input multi-output neural network, one for each day in a week. Each network has 24 outputs representing the expected hourly load for a day. One important job in designing the forecasting system is the selection of the input variables. In this electric load forecasting system, we have three main kinds of input variables: 1) Historical data of loads: hourly loads of the previous day were collected and used as historical load inputs. These data reflect the family habit of consuming power. 2) Temperature inputs: the average temperature of the previous day and the present day are used as two inputs in this forecasting system. 3) Rainfall index inputs: the average rainfall indexes of the previous day and the present day are used as two inputs in this forecasting system. The range of the rainfall index is from 0 (no rain) to 1 (heavy rain).

One network serves one day-type (from Monday to Sunday). Each neural network has 28 inputs and 24 outputs. The first 24 input nodes (x_1, …, x_{24}) represent the previous 24 hourly loads and are denoted by $x_i = L_i^d(t-1)$,where i = 1,2,…,24. Node 25 (x_{25}) and node 26 (x_{26}) represent the average temperatures of the previous day and present day respectively. Node 27 (x_{27}) and node 28 (x_{28}) represent the average rainfall indexes of the previous day and present day respectively. The output layer consists of 24 output nodes that represent the forecasted 24 hourly loads of a day and are denoted by $y_l(t) = L_l(t)$, l = 1,2,…,24. Data of 12 weeks (week 1 to week 12) for learning and data of 2 weeks (week 13 to week 14) for testing are prepared. Referring to (9), the proposed neural network used for the daily electric load forecasting is governed by,

$$y_l(t) = net_o^l(\sum_{j=1}^{n_h} net_d^j(net_s^j(\sum_{i=1}^{24} x_i v_{ij}))w_{jl}),\ l = 1, 2, \ldots, 24 \tag{21}$$

The number of hidden node (n_h) is changed from 3 to 9 in order to test the learning performance. GA is employed to tune the parameters of the neural network of (21). The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \tag{22}$$

$$err = \frac{1}{12}\sum_{t=1}^{12}\frac{1}{24}\sum_{k=1}^{24}\frac{\left|y_k^d(t) - y_k(t)\right|}{y_k^d(t)} \tag{23}$$

The value of *err* indicates the mean absolute percentage error (MAPE) of the forecasting result. The population size is 10. The number of the iterations to train the proposed neural network is 2000. The results are tabulated in Table III and Table IV for Wednesday and Sunday respectively. The fitness value and the number of parameters of the network are shown. We can observe that the performance of the proposed neural network is better than the traditional one. Table V and Table VI show the average training error from week 1 to week 12 in term of MAPE and the average forecasting error from week 13 to week 14 in term of MAPE on Wednesday and Sunday respectively. The best training errors are 1.7829 and 2.0776 for Wednesday and Sunday respectively. These imply 24.0% and 32% improvements respectively over the traditional neural network. The best forecasting errors are 1.9365 and 1.9120 for Wednesday and Sunday respectively. These imply 31.6% and 30.9% improvements. Fig. 5 shows the results of the daily electric load forecasting on Sunday (week 13). We can see that the forecasting result using the proposed neural network is better and a good forecasting is obtained.

B. Pattern Recognition

An application on pattern recognition by the proposed neural network will be presented in this section. Every point on a two-dimensional plane is characterized by a

number. A 10-input-3-output neural network is used. The ten inputs nodes, x_i, i = 1, 2, … ,10, are the numbers corresponding to 10 uniformly sampled points of the input pattern. Three patterns are to be recognized: rectangle, triangle and straight line. We use 300 sets of 10 samples points for each pattern to train the neural network. Hence, we have 900 (300×3) sets of data for training. The three outputs, $y_l(t)$, l = 1, 2, 3 are the output value of each pattern. A larger value of $y_l(t)$ implies that the input pattern matches more closely to the defined class. For example, a larger $y_1(t)$ implies that the input pattern more likely to be a rectangle. Referring to (9), the proposed neural network used for the pattern recognition is governed by,

$$y_l(t) = net_o^l\left(\sum_{j=1}^{n_h} net_d^j\left(net_s^j\left(\sum_{i=1}^{10} x_i v_{ij}\right)\right) w_{jl}\right),\ l = 1, 2, 3. \tag{24}$$

GA is employed to tune the parameters of the proposed neural network of (24). The fitness function is defined as follows,

$$fitness = \frac{1}{1+err} \tag{25}$$

$$err = \sum_{k=1}^{10} \frac{\sum_{t=1}^{300}\left(\left(\frac{y_k(t)}{\|\mathbf{y}(t)\|}\right)^2 - \left(\frac{y_k^d(t)}{\|\mathbf{y}^d(t)\|}\right)^2\right)}{300\times10} \tag{26}$$

The value of *err* indicates the mean square error (MSE) of the recognition system. The initial values of the neural network parameters are randomly generated. For comparison, a traditional feed-forward neural network (10-inputs-3-outputs) trained by GA is also used to recognize the patterns. The number of hidden node of the proposed network (n_h) and the traditional network are 7 and 16 respectively, which are chosen by trial and error through experiments for good performance. The population size is 10 and the number of iterations to train the neural network is 2000. The probability of crossover for GA is set at 0.8 for both

networks. The probabilities of mutation are set at 0.1 and 0.06 for the proposed and the traditional neural networks respectively. After training, we use 600 (200×3) sets of data for testing. The results are tabulate in Table VII. From this Table, it can be seen that the training error and forecasting error of the proposed neural network are smaller. The recognition accuracy of the proposed network is also better. When 7 hidden nodes for the proposed neural network and 16 hidden nodes for the traditional neural network are used, the numbers of parameters to be tuned are 125 and 227 respectively. The proposed neural network has only 55% of the number of parameters of the traditional neural network. Hence, the performance of proposed neural network is better and fewer hidden nodes are needed.

VI. CONCLUSION

A novel GA-based neural network has been proposed. Its parameters can be tuned by GA with arithmetic crossover and non-uniform mutation. A novel neuron model with two activation functions has been introduced. By employing this neuron model in the hidden layer, the performance of the neural network is found to be better than that of the traditional feed forward neural network. Examples of multi-input XOR problem, sunspot forecasting, short-term daily home electric load forecasting and pattern recognition have been given. The performance of the proposed neural network in these examples is good.

ACKNOWLEDGEMENTS

The work described in this paper is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administration Region, China (Project No. PolyU 5127/00E), and is partially supported by a research grant from The Hong Kong Polytechnic University (Project No. G-V954).

REFERENCES

[1] J.H. Holland, *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[2] D.T. Pham and D. Karaboga, *Intelligent optimization techniques, genetic algorithms, tabu search, simulated annealing and neural networks*. Springer, 2000.

[3] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs* (2nd ed.). Springer-Verlag, 1994.

[4] Y. Hanaki, T. Hashiyama, and S. Okuma, "Accelerated evolutionary computation using fitness estimation," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 1, 1999, pp. 643-648.

[5] K.A. De Jong, *Ph.D. Thesis*: *An analysis of the behavior of a class of genetic adaptive systems.* Ann Arbor, MI: University of Michigan, 1975.

[6] G.X. Yao and Y. Liu "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, July 1999.

[7] S. Amin and J.L. Fernandez-Villacanas, "Dynamic Local Search," in *Proc. 2nd Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1997, pp. 129-132.

[8] L.K. Wong, S.H. Ling, F.H.F. Leung, Y.S. Lee, S.H. Wong, T.H. Lee, H.K. Lam, K.H. Ho, D.P.K. Lun, and T.C. Hsung, "An intelligent home," in *Proc. Workshop on Service Automation and Robotics*, Hong Kong, 19-20 June 2000, pp. 111-119.

[9] K.Y. Lee, Y.T. Cha, and J.H. Park, "Short-term load forecasting using an artificial neural network," *IEEE Trans. Power Systems*, vol. 7, no. 1, pp. 124-132, 1992.

[10] Y.Y. Hsu and C.C. Yang, "Design of artificial neural networks for short-term load forecasting. Part 1: self-organizing feature maps for day type identification," *IEE Proceedings-C*, vol. 138, no. 5, pp. 407-418, 1991.

[11] Drezga and D.S. Rahman, "Short-term load forecasting with local ANN predictors," *IEEE Trans. Power System*, vol. 14, no. 3, pp. 844-850, Aug. 1999.

[12] J.A. Momoh, Y. Wang, and M. Elfayoumy, "Artificial neural network based load

forecasting," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 4, 1997, pp. 3443-3451.

[13] D. Part, M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg, "Electric load forecasting using an artificial neural network," *IEEE Trans. Power System*, vol. 6, no. 2, pp. 442-449, 1991.

[14] C.N. Lu, H.T. Wu, and S. Vemuri, "Neural network based short-term load forecasting," *IEEE Trans. Power Systems*, vol. 8, no. 1, pp. 336-342, Feb.1993.

[15] A.P. Rewagad and V.L. Soanawane, "Artificial neural network based short term load forecasting," in *Proc. 1998 IEEE Region 10 Int. Conf. Global Connectivity in Energy, Computer, Communication and Control*, vol. 2, 1998, pp. 588-595.

[16] A.G. Bakirtzls, V. Petridls, S.J. Klartzis, M.C. Alexladls, and A.H. Malssls, "A neural network short term load forecasting model for Greed power system," *IEEE Trans. Power Systems*, vol. 11, no. 2, pp. 858-863, May 1996.

[17] O. Mohammed, D. Part, R. Merchant, T. Dinh, C. Tong, and A. Azeem, "Practical experience with an adaptive neural network short-term load forecasting system," *IEEE Trans. Power System*, vol. 10, no. 1, pp. 254-265, Feb 1995.

[18] M. Brown and C. Harris, *Neuralfuzzy adaptive modeling and control*. Prentice Hall, 1994.

[19] M. Li, K. Mechrotra, C. Mohan, and S. Ranka, "Sunspot numbers forecasting using neural network," in *Proc. 5th IEEE International Symposium on Intelligent Control*, 1990, pp. 524-528.

[20] B.D. Liu, C.Y. Chen and J. Y. Tsao, "Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics, Part B*, vol. 31 no. 1, pp. 32-53, Feb. 2001.

[21] H. Juidette and H. Youlal, "Fuzzy dynamic path planning using genetic algorithms," *Electronics Letters*, vol. 36, no. 4, pp. 374-376, Feb. 2000.

[22] R. Caponetto, L. Fortuna, G. Nunnari, L. Occhipinti, and M. G. Xibilia, "Soft computing for greenhouse climate control," *IEEE Trans. Fuzzy Systems*, vol. 8, no. 6, pp. 753-760, Dec. 2000.

[23] M. Setnes and H. Roubos, "GA-fuzzy modeling and classification: complexity and performance," *IEEE. Trans. Fuzzy Systems*, vol. 8, no. 5, pp. 509–522, Oct. 2000.

[24] A.E. Bryson and Y.C. Ho, *Applied Optimal Control*. Blaisdell, 1969.

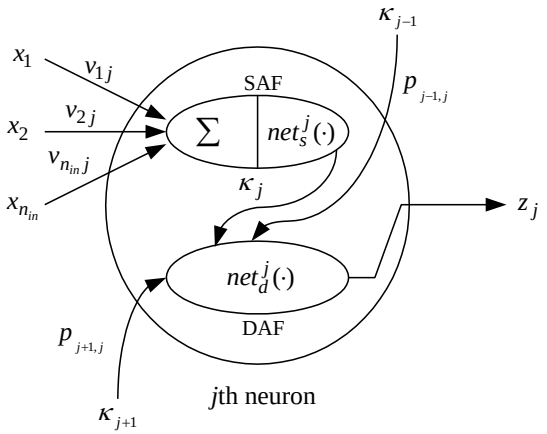


Fig. 1. Model of the proposed neuron.

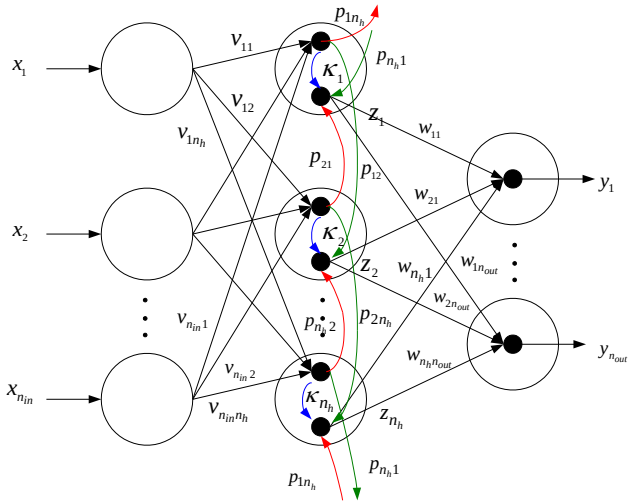
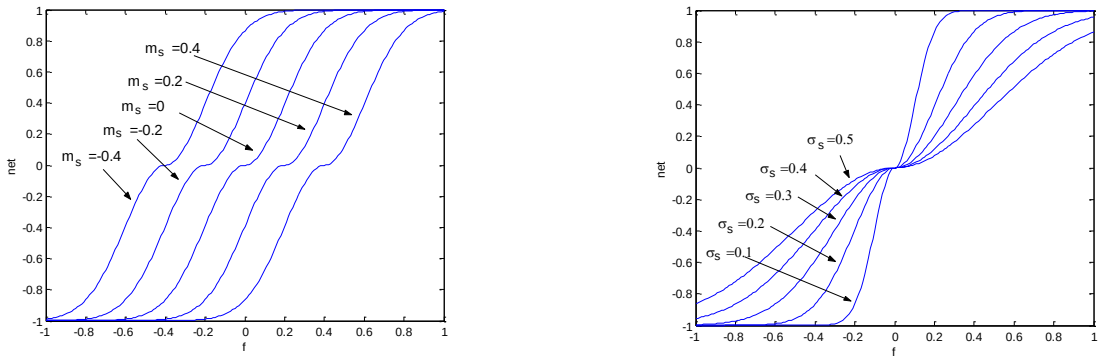
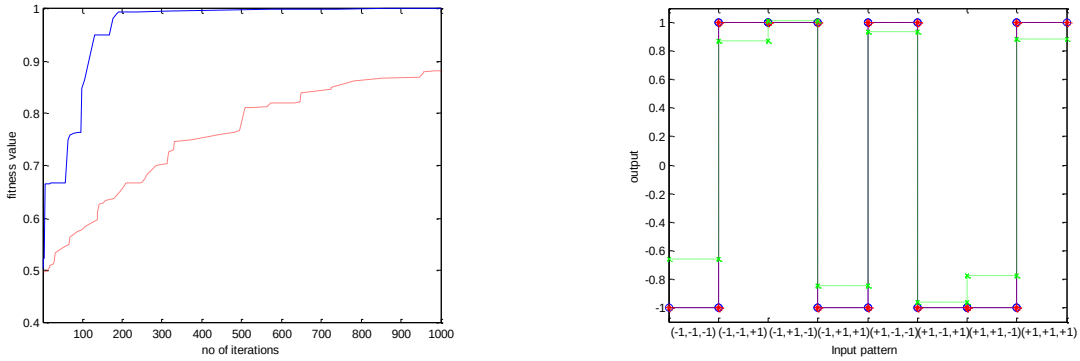


Fig. 2. Connection of the proposed neural network



(a) (b)

Fig. 3. Sample activation functions of the proposed neuron: (a) $\sigma_s = 0.2$, (b) $m_s = 0$.



(a) (b)

Fig. 4. Results of the XOR problem: a) Fitness values obtained from the proposed (solid line) and the traditional neural networks (dotted line) for 1000 iterations; b) The output patterns obtained by the proposed (dotted line with '✳' marks) and the traditional neural networks (dotted line with '×' marks), as compared with the desired output (solid line with '○' marks).

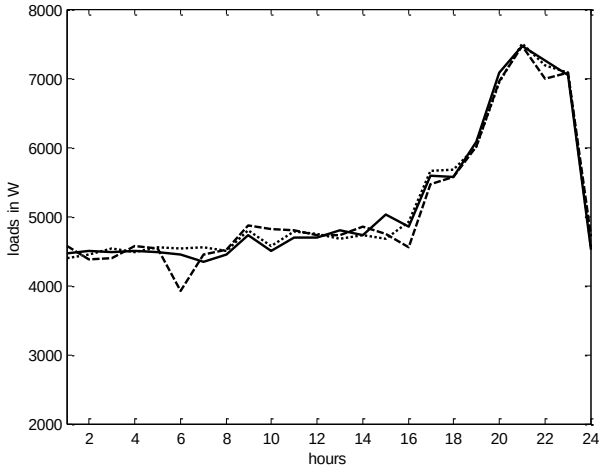


Fig. 5. Daily electric load forecast results on Sunday (Week13) obtained by the proposed neural network (dotted line) and the traditional neural network (dashed line), as compared with the actual load (solid line).

	Fitness Value	MAE
Proposed Neural Network	0.999988	0.002533
Traditional Neural Network	0.880901	0.135201

Table I. Results of the proposed neural network and the traditional neural network for 3-inputs XOR classification problem after 1000 iterations.

	Fitness Value	Training error (MAE)	Forecasting (MAE)
Proposed Neural Network	0.9546	9.51	13.18
Traditional Neural Network	0.9429	12.12	14.31

Table II. Results of the proposed neural network and the traditional neural network for forecasting of the sunspot number after 1000 iterations.

	Proposed Neural Network		Traditional Neural Network	
n_h	Fitness Value	Number of parameters	Fitness Value	Number of parameters
3	0.977358	216	0.944286	183
4	0.978133	272	0.974208	236
5	0.977819	328	0.977089	289
6	0.981248	384	0.967182	342
7	0.977091	440	0.972413	395
8	0.982483	496	0.967235	448
9	0.981188	552	0.950723	501

Table III. Results of the proposed neural network and the traditional neural network for daily electric load forecasting for Wednesday.

	Proposed Neural Network		Traditional Neural Network	
n_h	Fitness Value	Number of parameters	Fitness Value	Number of parameters
3	0.979647	216	0.945921	183
4	0.979552	272	0.965002	236
5	0.977991	328	0.967021	289
6	0.977371	384	0.968254	342
7	0.975546	440	0.970349	395
8	0.976570	496	0.964480	448
9	0.977074	552	0.966030	501

Table IV. Results of the proposed neural network and the traditional neural network for daily electric load forecasting for Sunday.

	Proposed Neural Network		Traditional Neural Network	
n_h	Training error (MAPE)	Forecasting error (MAPE)	Training error (MAPE)	Forecasting error (MAPE)
3	2.3167	2.8466	5.9002	5.4427
4	2.2356	2.4853	2.6474	3.6954
5	2.2684	2.8813	2.3448	2.8316
6	1.9110	2.0443	3.3932	3.8580
7	2.3446	2.7791	2.8370	2.8964
8	1.7829	1.9365	3.3874	3.4152
9	1.9173	1.9898	5.1831	4.3619

Table V. Training and forecasting errors in term of MAPE for daily electric load forecasting for Wednesday

	Proposed Neural Network		Traditional Neural Network	
n_h	Training error (MAPE)	Forecasting error (MAPE)	Training error (MAPE)	Forecasting error (MAPE)
3	2.0776	2.3054	5.7170	5.2094
4	2.0875	1.9120	3.6268	4.0758
5	2.2504	2.5362	3.4104	3.3927
6	2.3153	2.5324	3.2786	3.2126
7	2.5067	2.6807	3.0556	2.7665
8	2.3992	2.3987	3.6828	4.1690
9	2.3464	2.2005	3.5164	4.1102

Table VI. Training and forecasting errors in term of MAPE for daily electric load forecasting for Sunday.

	Fitness value	No. of parameters (n_h)	Training error (MSE)	Forecasting error (MSE)	Recognition accuracy
Proposed neural network	0.9887	125	0.0114	0.0238	96.00%
Traditional neural network	0.9717	227	0.0291	0.0411	93.33%

Table VII. Results of the proposed neural network and the traditional neural network for pattern recognition.